PMSM FOC motor control using AURIX™ TC3xx

32-bit TriCore™ AURIX™ TC3xx microcontroller

About this document

Scope and purpose

This document describes the implementation of Permanent Magnet Synchronous Motors (PMSM) Field Oriented Control (FOC) software for a 3-phase motor using the Infineon AURIX™ TC3xx microcontroller.

Intended audience

This document is intended for customers who would be using AURIX™ TC3xx Family microcontrollers for PMSM FOC applications.

Note: A basic knowledge of motor drive applications and FOC are mandatory.

Table of contents

About this document **1**
Table of contents **1**
1 Introduction **3**
1.1 Key features 4
1.2 Abbreviations and acronyms 4
1.3 AURIX™ resource allocation 5
1.4 Software overview 6
1.5 Limitations of use for PMSM FOC software 8
2 PMSM FOC software components **9**
2.1 Control schemes 9
2.1.1 Speed control scheme 9
2.1.2 Current control scheme 9
2.2 Motor start / speed change / motor stop operations control 10
2.3 Ramp generator 10
2.4 Stationary and rotating reference frame 10
2.5 Current sensing and calculation 12
2.5.1 Three phase current sensing 12
2.5.2 High-side DC-link current sensing 14
2.6 Space vector modulation 14
2.7 PWM Generation 19
2.8 Motor speed and position feedback 20
2.9 DC-link voltage sensing 20
2.10 Phase voltage sensing 21
2.11 Interrupts 21
2.12 Motor control state machine 22
2.13 Gate driver support 24
3 Configuration **26**
3.1 User configuration 26
3.1.1 General 26
3.1.2 Custom Kit configuration 27

3.1.3 Advanced user configuration 28
3.2 Hardware configuration 28
3.2.1 Controller card 28
3.2.1.1 Interrupt priorities 28
3.2.1.2 PWM outputs (GTM configuration) 29
3.2.1.3 Incremental encoder (GPT12 configuration) 29
3.2.1.4 TLE9180 configuration (GPIO and QSPI) 30
3.2.1.5 TLF35584 configuration (GPIO and QSPI) 31
3.2.2 Inverter board configuration 31
3.2.2.1 General 31
3.2.2.2 Supply voltage and supply voltage sensing 31
3.2.2.3 Output, Bridge Driver and B6 Bridge 32
3.2.2.4 Current sensing 32
3.2.2.5 Phase voltage sensing 33
3.2.3 Motor specific configuration 34
3.2.3.1 Motor parameters 34
3.2.3.2 Incremental encoder configuration 34
3.2.3.3 Speed limits 34
3.2.3.4 Current PI controllers settings 35
3.2.3.5 Speed PI controllers settings 36
3.2.3.6 Speed and dq-axis current references ramp settings 37
4 PMSM FOC software data structure 38
5 PMSM FOC software API functions 39
6 References 43
Revision history 44

1 Introduction

The intention of this software is to offer functionality to drive Permanent Magnet Synchronous Motors (PMSM) in sensor mode using AURIX™ TC3xx devices. It contains all the common modules necessary for the modes as generic drives, and provides a high level of configurability and modularity to address different segments.

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

Feedback on rotor position and rotor speed is required in FOC motor control. The feedback can come from sensorless mechanism or from sensors:

- 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 sensed in the motor phase, in the leg shunt or DC-Link shunt at the low-side MOSFET. In this software, phase current sensing is expected from the leg shunts.

In the Figure 1 one can see the typical block diagram for the PMSM FOC, where single shunt and three shunt low-side current sensing are supported.

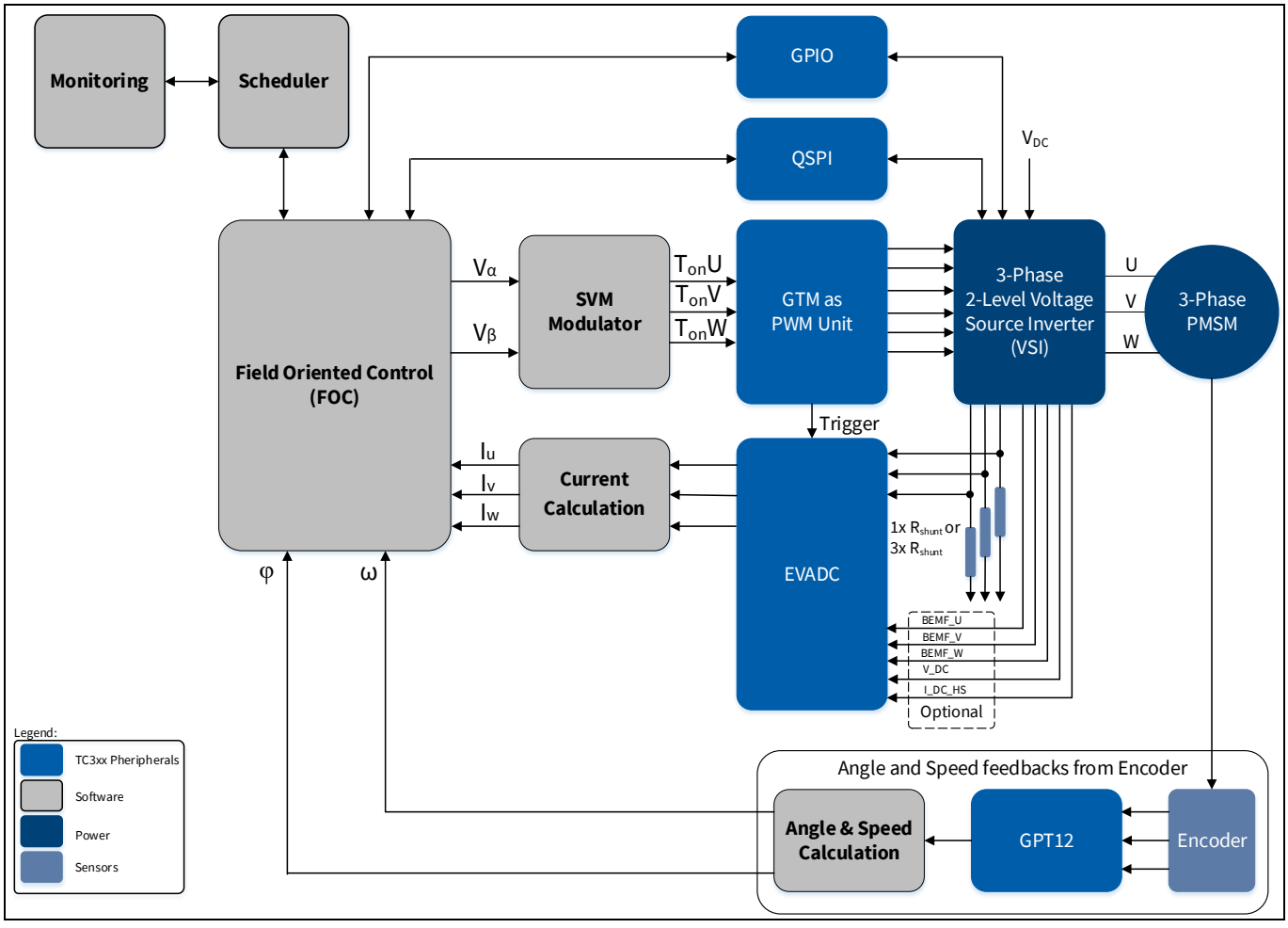

Figure 1 Block diagram of PMSM FOC motor control

1.1 Key features

The key features supported are listed in the following table:

Table 1 Key Software features supported

Feature	
Framework & Handling	Task scheduler
	Motor control state machine
Peripheral driver	Microcontroller specific low level driver (iLLD)
Math Control Blocks	Clarke transformations
	Park transformations
	I_d and I_q current PI controllers
	Speed PI controller
	sine, cosine, tan, atan, sqrt
	Ramp function
Control Scheme	Speed control
	Current control
Space Vector Modulation (SVM)	7-segment SVM
Device Feature	ADC synchronous conversion: motor phase current sensing (2 or 3 shunts) ADC synchronous conversion: motor phase voltage sensing
Gate driver support	TLE9180D
Rotor Speed and Angle Calculation	Encoder as position sensor
	Automatic encoder calibration
Others	Linear ramp generator for speed and current references
	Low pass filter

Note: The example software project does not include resolver or hall functionality, neither sensorless operation.

1.2 Abbreviations and acronyms

Term	Definition
API	Application Programming Interface
AURIX™	AURIX™ microcontroller family
CPU	Central Processing Unit
CSA	Current Sense Amplifier
EVADC	Enhanced Versatile Analog-to-Digital Converter
FOC	Field Oriented Control
GPIO	General Purpose Input / Output
GPT12	General Purpose Timer Unit
GTM	Generic Timer Module
iLLD	Infineon Low Level Driver

Term	Definition
ISR	Interrupt Service Routine
MCU	Microcontroller Unit
MRST	Master Receive Slave Transmit
MTSR	Master Transmit Slave Receive
PI	Proportional Integral Controller
PMSM	Permanent Magnet Synchronous Motors
PWM	Pulse Width Modulation
SW	Software
QSPI	Queued Synchronous Peripheral Interface
SVM	Space Vector Modulation
TC387	An AURIX™ TC3xx derivate

1.3 AURIX™ resource allocation

The AURIX™ TC3xx microcontroller family is ideal for PMSM FOC motor control systems. They have dedicated motor control peripherals, GPT12, GTM, EVADC, QSPI and EDSADC [1]. In this PMSM FOC motor control software, used hardware peripherals are listed in the Table 2.

Note: *The default resource allocations are for AURIX™ TC3xx Motor Control Application Kit, which contains of AURIX ™ TC387 Application Kit with TFT Display.*

Note: *Peripherals that are used for additional features such as display and touch, graphical user interface and operating system are not described.*

Table 2 AURIX™ TC387 Peripherals used for sensored PMSM FOC with three shunt current sensing

AURIX™ peripherals	Usage	Default resource allocation in AURIX™ TC387
GTM	Base timer	GTM TOM1 Channel 0
	PWM generation for phase U – low side	GTM TOM1 Channel 1
	PWM generation for phase U – high side	GTM TOM1 Channel 2
	PWM generation for phase V – low side	GTM TOM1 Channel 3
	PWM generation for phase V – high side	GTM TOM1 Channel 4
	PWM generation for phase W – low side	GTM TOM1 Channel 5
	PWM generation for phase W – high side	GTM TOM1 Channel 6
	EVADC Trigger	GTM TOM1 Channel 7
EVADC	Phase U current sensing (VO1)	EVADC Group 0 Channel 0 Queue 0
	Phase V current sensing (VO2)	EVADC Group 3 Channel 0 Queue 0
	Phase W current sensing (VO3)	EVADC Group 2 Channel 0 Queue 0
	Output of reference voltage of differential amplifier (VRO)	EVADC Group 1 Channel 0 Queue 0
	Phase U voltage sensing (BEMF U)	EVADC Group 1 Channel 5 Queue 2
	Phase V voltage sensing (BEMF V)	EVADC Group 2 Channel 3 Queue 2

AURIX™ peripherals	Usage	Default resource allocation in AURIX™ TC387
	Phase W voltage sensing (BEMF W)	EVADC Group 3 Channel 1 Queue 2
	DC-link voltage sensing (VDC)	EVADC Group 1 Channel 3 Queue 1
	High-side DC-link current sensing	EVADC Group 2 Channel 2 Queue 1
QSPI	TLE9180 Chip select	QSPI4 Slave select signal SLSO 3
	TLE9180 SPI clock	QSPI4 Serial clock SCLK
	TLE9180 SPI MOSI	QSPI4 MTSR
	TLE9180 SPI MISO	QSPI4 MRST B
	TLF35584 Chip select	QSPI2 Slave select signal SLSO 1
	TLF35584 SPI clock	QSPI2 Serial clock SCLK
	TLF35584 SPI MOSI	QSPI2 MTSR
	TLF35584 SPI MISO	QSPI2 MRST B
GPT12	Encoder A	GPT120 T3INA
	Encoder B	GPT120 T3EUDA
	Encoder Top Zero (Index)	GPT120 T4INA

1.4 Software overview

The project structure is shown in Figure 2. The project is divided into application software (AppSw), base software (BaseSw) and operating system software (OS). The application software consists of main program (Main), display and touch application (Display), GUI application (OneEye), operating system tasks (OSTasks) and PMSM FOC software (PmsmFoc). Base software consists of AURIX™ TC38A low level drivers (iLLD), infrastructure, service and external device driver libraries.

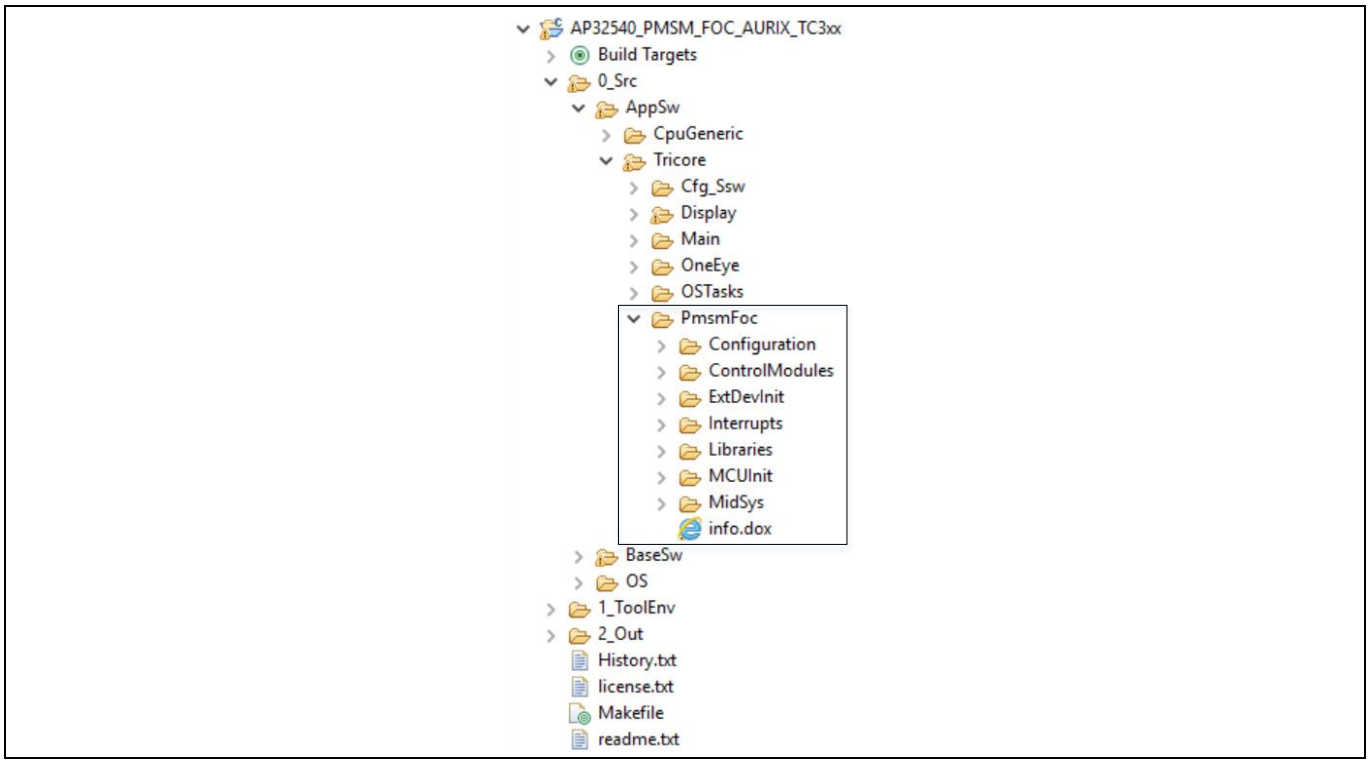

Figure 2 Project folder structure

Introduction

The PMSM FOC motor control application software is developed based on a well-defined layered approach.

The layered architecture, shown in Figure 3, is designed in such a way as to separate the modules into groups. This allows different modules in a given layer to be easily replaced without affecting the performance in other modules and the structure of the complete system.

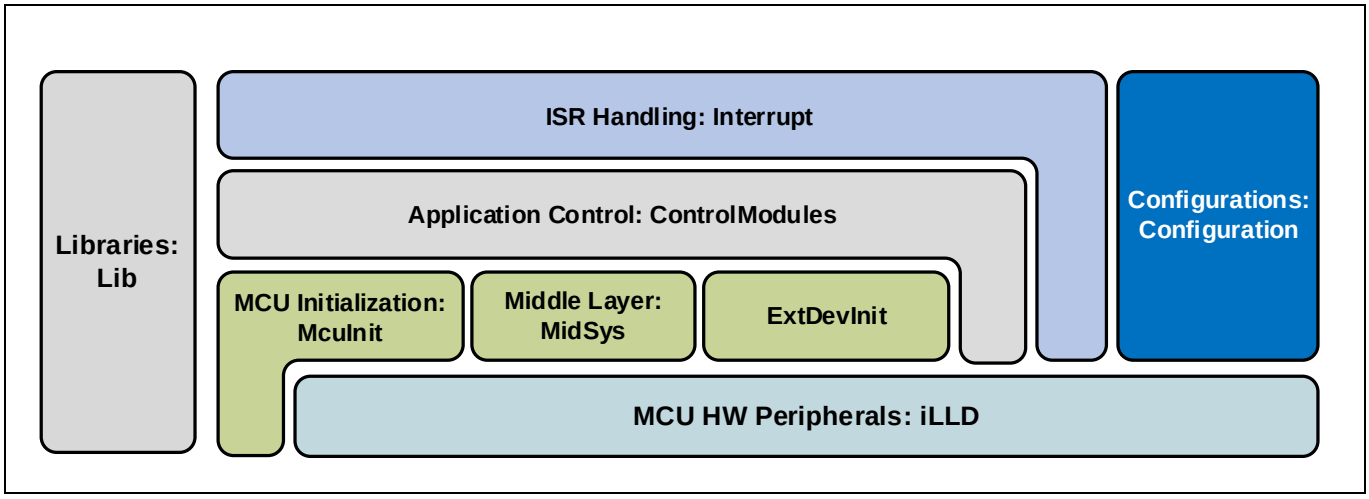

Figure 3 PMSM FOC software overview, layered structure

Configuration

The configuration is divided. The user configuration effects the general behavior of the software. The file PmsmFoc_UserConfig.h can be modified. Pre-defined hardware kit is available.

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

This layer is divided into:

- Controller Card
- Inverter Card
- Motors

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

The static configuration and required scaling are accessible with the following files:

- PmsmFoc _Macro.h
- PmsmFoc_VariablesScaling.h

Note: If user would like to use the AURIX™ TC3xx Motor Control Application Kit as it is, do not change the configuration and definition files.

All configurations can be found in the folder ‘Configuration’.

Application Control

This layer consists of FOC SW control modules. This includes the current reconstruction, PI controllers, state machine and ramping for example.

All the routines mentioned are called from the EVADC Interrupt Service Routine.

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

External devices

This layer controls the initialization of microcontroller peripherals used for control and communication with external devices such as TLE9180D and TLE35584.

All files for this layer can be found in the folder ‘ExtDevInit’.

ISR handling

This layer consists of a QSPI, EVADC and GPT12 ISR functions for communication with external devices, encoder top zero position handling, three phase shunt current sensing, DC-link voltage sensing and optionally high-side DC link current sensing and phase voltage sensing. All files are stored in the ‘Interrupts’ folder.

Libraries

This layer consists of a legacy libraries for mathematical operations and motor control functions. This includes the Clarke transform, Park transform, space vector modulation, etc.

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

MCU initialization

This layer controls the initialization of all microcontroller peripherals used in PMSM FOC application. It contains iLLD data structure for driver runtime variables. This layer closely interacts with iLLD and the MidSys layer to configure each peripheral.

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

Middle Layer

This layer provides routines for PWM generation, EVADC sensing, 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 software. For example when using Hall sensors you can add in 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’.

1.5 Limitations of use for PMSM FOC software

For this application note the current software version used is PMSM FOC software v1.0.x.

At the time of release of this example software, the following limitations in usage apply:

- Only a single motor drive is supported.
- Position and speed feedbacks from Hall sensors or resolver are not supported.
- This software is developed in BIFACES™ V1.0.3. It is not tested on other IDE (Integrated Development Environment) platforms.
- The software is compiled only with Hightec Gnuc compiler, v4.9.3.0.
- TLE9180 driver use predefined values during startup configuration.

The following are not currently documented:

- High-side DC-link sensing details.
- Register configuration.

2 PMSM FOC software components

2.1 Control schemes

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

- Speed control scheme
- Current control scheme

2.1.1 Speed control scheme

The speed control scheme shown in Figure 4, is a closed loop control. This scheme uses a cascaded speed and current control structures. This is due to the change response requirement for a speed control loop which is much slower than the one for current loop.

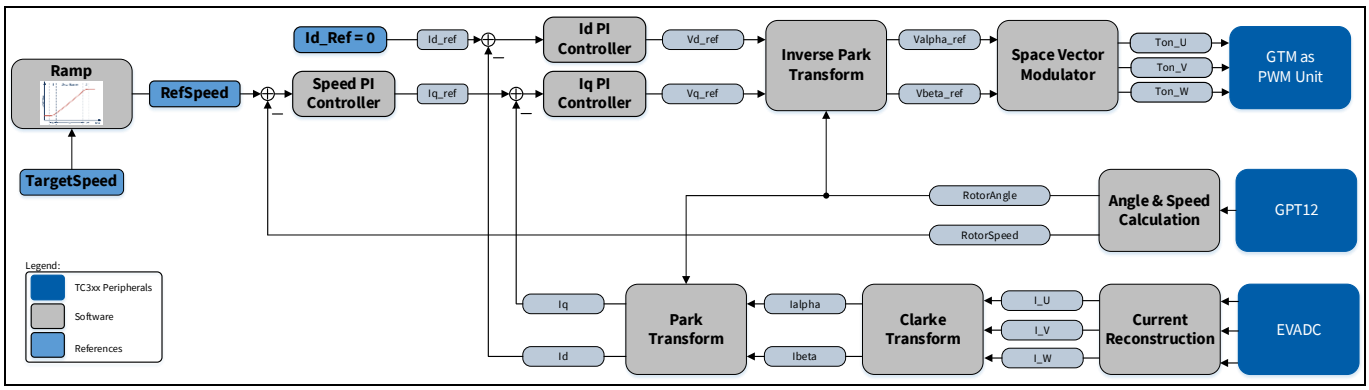

Figure 4 Control scheme: Speed control

The ramp generator input is connected to a user target speed value. The output of ramp generator is speed controller reference. The speed controller calculates the motor torque required to follow the target speed. While the current loops drive the motor currents needed to generate this torque. The proportional plus integral (PI) speed loop compensator acts on the error between the reference speed and the actual speed. The integral term forces the steady state error to zero while the proportional term improves the high frequency response. The PI compensator gains are adjusted depending on the motor and load characteristics to meet the target dynamic performance.

The current loops calculate the inverter voltages to drive the motor currents needed to generate the desired torque. The FOC uses the Clarke transform and a vector rotation to transform the motor winding currents into two quasi DC components, an I_d component that reinforces or weakens the rotor field and an I_q component that generates motor torque.

Two separate regulators control the I_d and I_q currents and a forward vector rotation transforms the current loop output voltages V_d and V_q into the two phase AC components (V_α and V_β). The SVPWM generates the three phase power inverter switching signals based on the V_α and V_β voltage inputs.

2.1.2 Current control scheme

A current control scheme shown in Figure 5, uses a current control structure. This control scheme is useful in applications where direct current control is important, such as e-bike or battery-operated devices for example.

Two ramp generators are used. The input of first one is connected to a user target I_q value (torque), while input of second one is connected to a user target I_d value (flux). The motor torque is maintained at torque reference value (I_q). Any change in the load will cause the speed of the motor to change but the torque remains constant.

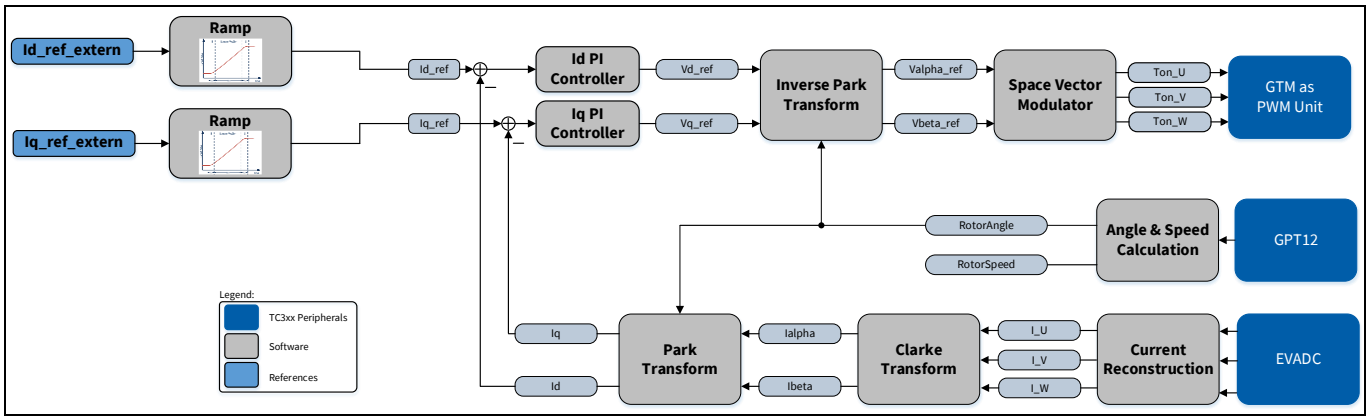

Figure 5 Control scheme: Current control

The current loops calculate the inverter voltages to drive the motor currents needed to generate the desired torque. The FOC uses the Clarke transform and a vector rotation to transform the motor winding currents into two quasi DC components, an I_d component that reinforces or weakens the rotor field and an I_q component that generates motor torque.

Two separate regulators control the I_d and I_q currents and a forward vector rotation transforms the current loop output voltages V_d and V_q into the two phase AC components (V_α and V_β). The SVPWM generates the three phase power inverter switching signals based on the V_α and V_β voltage inputs.

2.2 Motor start / speed change / motor stop operations control

The motor is started with the start command. The minimum target speed is defined as USER_MOTOR_SPEED _LOW_LIMIT_RPM, while the maximum target speed is defined as USER_MOTOR_SPEED_HIGH_LIMIT_RPM. The motor is stopped with the stop command at any point of time or after finishing calibration process.

2.3 Ramp generator

An input parameter is ramped from an initial value to an end value. The ramp generator input is connected to a value set by user. In current software implementation three ramp generators are used, one for speed reference and two external dq-axis current references.

The speed ramp slew rate is defined as USER_MOTOR_SPEED_RAMP_SLEW_RATE, while the external dq-axis current references ramp slew rates are defined as USER_MOTOR_CURRENT_D_RAMP_SLEW_RATE and USER_MOTOR_CURRENT_Q_RAMP_SLEW_RATE. All values are defined as maximum change of value per second in the user configuration file, PmsmFoc_Motor_Nanotec_DB42S02.h (refer to 3.2.3.6).

The speed ramp sampling period is defined as USER_MOTOR_SPEED_RAMP_PERIOD, while d- and q-axis current reference ramps sampling periods are defined as USER_MOTOR_CURRENT_D_RAMP_PERIOD and USER_MOTOR_CURRENT_Q_RAMP_PERIOD. All values are defined in seconds in the user configuration file, PmsmFoc_Motor_Nanotec_DB42S02.h (refer to 3.2.3.6). In current software implementation both current ramp generator functions are called at the sampling rate defined for the q-axis.

2.4 Stationary and rotating reference frame

The following figure shows the currents in three different co-ordinate systems and their form, respectively.

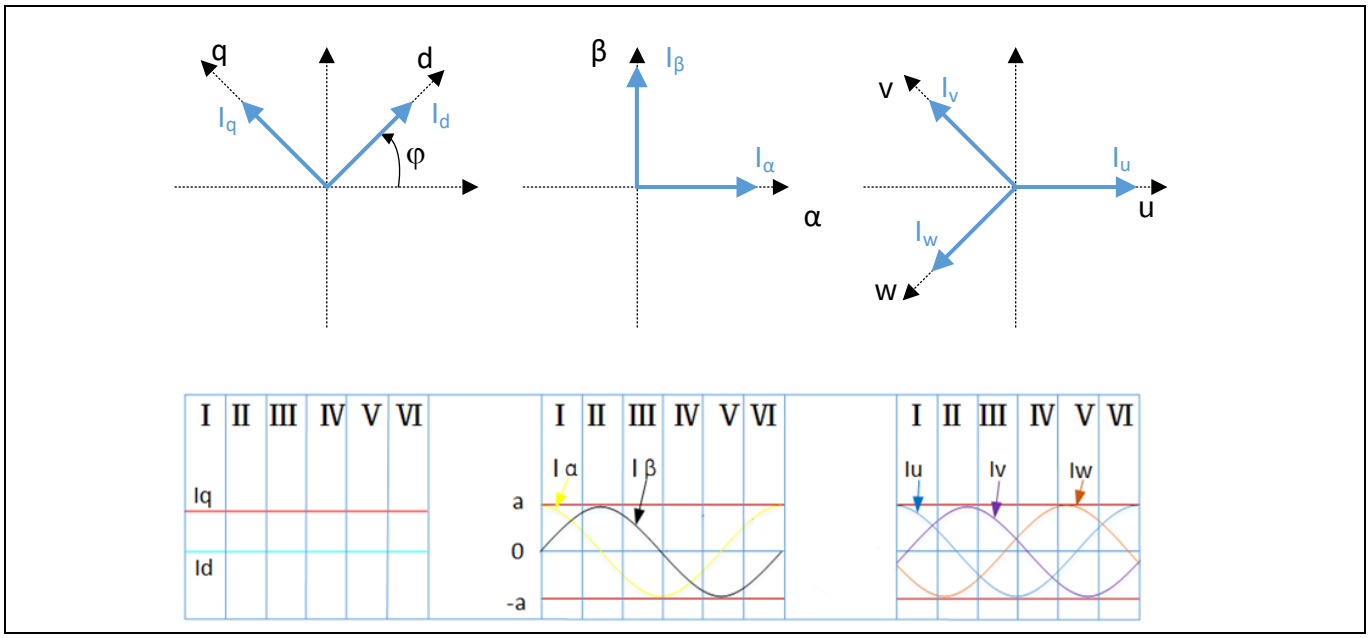

Figure 6 Reference Frame Alternate

From left to right:

- The first is the two rotation coordinates of stator. The currents I_q and I_d are direct current, but with 360-degree rotation.
- The second one is the two static coordinate of stator. The horizontal axis is the α-axis and the vertical axis is the β-axis. The value of the current I_α and I_β changes in accordance to the sine law.
- The third one is the three static coordinate of stator. There are three axes U, V, and W, which are spaced from each other by 120 degrees.

These three coordinates can be deduced from each other.

Inverse Park transform

$$i_\alpha = i_d \cdot cos\varphi - i_q \cdot sin\varphi \quad \text{Eq. 1}$$

$$i_\beta = i_d \cdot sin\varphi + i_q \cdot cos\varphi \quad \text{Eq. 2}$$

The transform from the two phase revolving system to the two phase static system is called the “inverse Park transform.

Inverse Clarke transform

$$i_u = i_\alpha \quad \text{Eq. 3}$$

$$i_v = \left(-i_\alpha + \sqrt{3} \cdot i_\beta\right)/2 \quad \text{Eq. 4}$$

$$i_w = \left(-i_\alpha - \sqrt{3} \cdot i_\beta\right)/2 \quad \text{Eq. 5}$$

The transform from the two phase static system to the three phase static system is called the “inverse Clarke transform.

The following two transform are used in the feedback path of a FOC control system.

Clarke transform

$$i_\alpha = i_u \quad \text{Eq. 6}$$

$$i_\beta = \frac{(i_v - i_w)}{\sqrt{3}} = \frac{(i_u + 2 \cdot i_v)}{\sqrt{3}} \quad \text{Eq. 7}$$

$$i_u + i_v + i_w = 0 \quad \text{Eq. 8}$$

The transform from the three static system to the two static system is called the “Clarke transform”. There are one or two phase current sensors which will measure the phase current, then determine the current i_α and i_β.

Park transform

$$i_d = i_\alpha \cdot cos\varphi + i_\beta \cdot sin\varphi \quad \text{Eq. 9}$$

$$i_q = -i_\alpha \cdot sin\varphi + i_\beta \cdot cos\varphi \quad \text{Eq. 10}$$

The transform from the two static phase system to the two phase revolving system is called the “Park transform”. This transform is used to get the direct current i_d and i_q which is the current feedback for the current control loop. In the two equations, the angle φ is defined as the rotor position angle.

2.5 Current sensing and calculation

2.5.1 Three phase current sensing

In order to implement sensor or sensorless field oriented control, it is crucial to measure the motor winding currents precisely. Motor phase current values are used for current control. Current is measured at every PWM cycle. The PWM period is defined as 1/ USER_INVERTER_PWM_FREQ_HZ. By default, in motor control power board, current sensing is implemented with three shunts in the low path of the B6 Bridge. The software only senses phase U and phase V current, and phase W current is calculated assuming the sum of the three phase current values is zero. The current sensing needs to be done when two low-side switches are closed. Therefore the triggers become active in the middle of the PWM signal. The trigger is released with an additional GTM TOM1 channel 7. The current sampling is done four times each channel per time stamp.

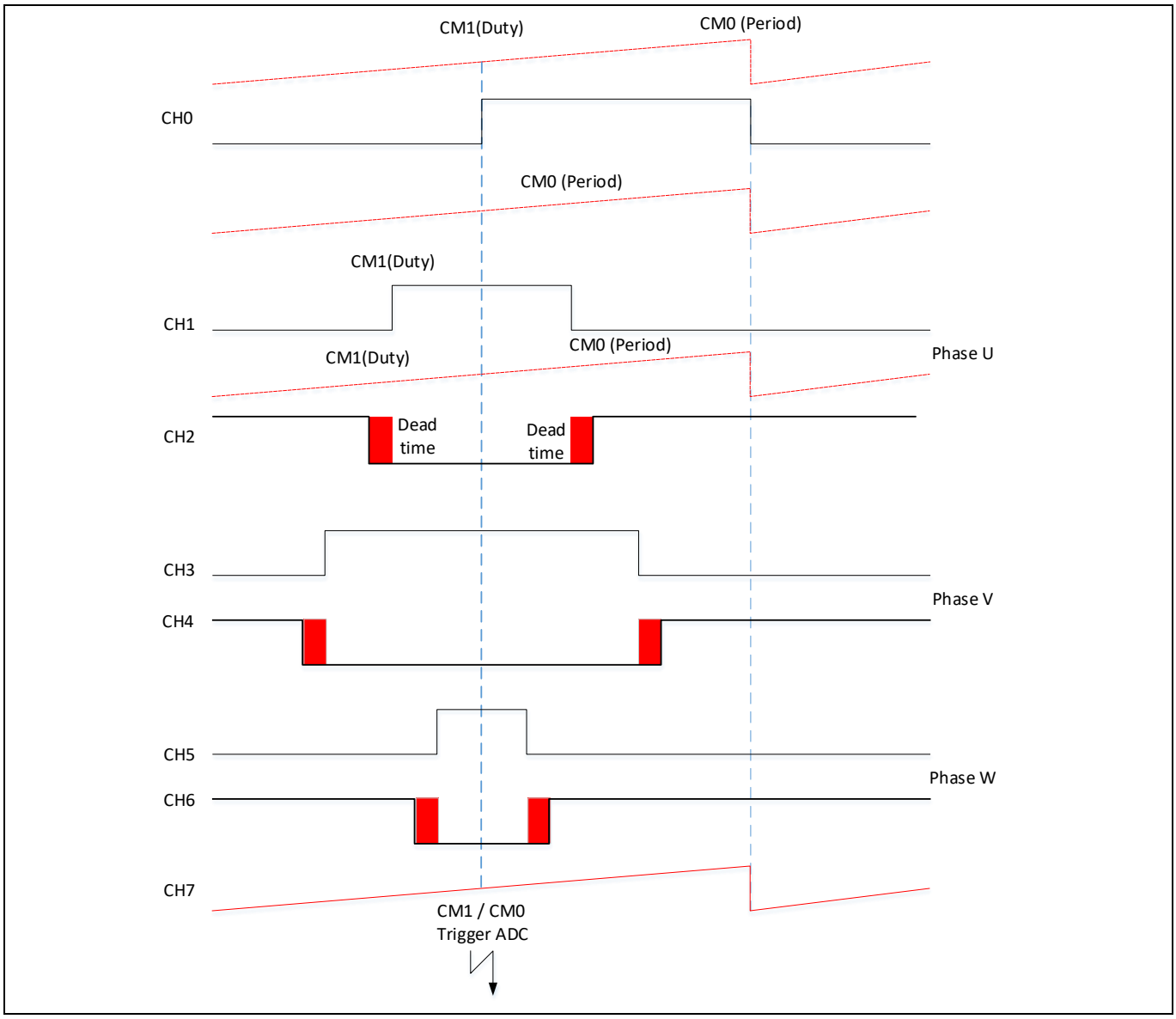

Figure 7 Current measurement, triggered by GTM TOM1 Channel 7

The following Figure 8 shows the details of the motor phase current feedback signal path. The voltage across the three phase shunt resistors is fed into the TLE9180D-31QK shunt positive and negative inputs (ISPx and ISNx, where x = 1, 2, 3). The TLE9180D-31QK has 3 integrated CSA. The outputs of the current sense amplifiers feed ADCs with an analog range from 0 V to 5 V. The DC output voltage at the outputs of the CSAs (VOx, x = 1, 2, 3) for zero differential input voltage is defined by the output of the reference buffer at Voltage Reference Output (VRO) pin. It is measured several times in the initialization phase when the motor does not rotate. The offset voltage must be considered each time when new phase currents are calculated. The offsets and gains of current sense amplifiers are programmable by using QSPI. Selected gains of current sense amplifiers are 30.18 (refer to Op_gain_1, Op_gain_2 and Op_gain_3 in Table 4). Selected VRO output voltage level is 2.5 V (refer to Op_0cl in Table 4).

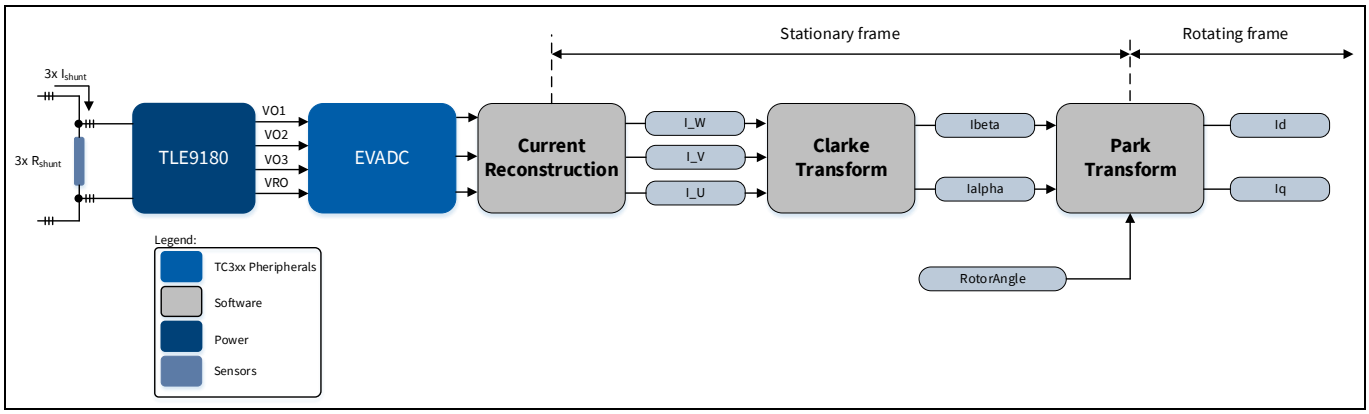

Figure 8 Motor current feedback signal path

Up to 4 EVADC kernels can be connected to synchronization groups to achieve parallel conversion of several input channels. Not all channels can be synchronized to each other, but certain groups can be formed. In the software example synchronization group A is used which contains Group 0, Group 1, Group 2 and Group 3.

For more details about triggering EVADC using GTM refer to [6].

2.5.2 High-side DC-link current sensing

The high-side DC-link current sensing could be used in motor control software for monitoring.

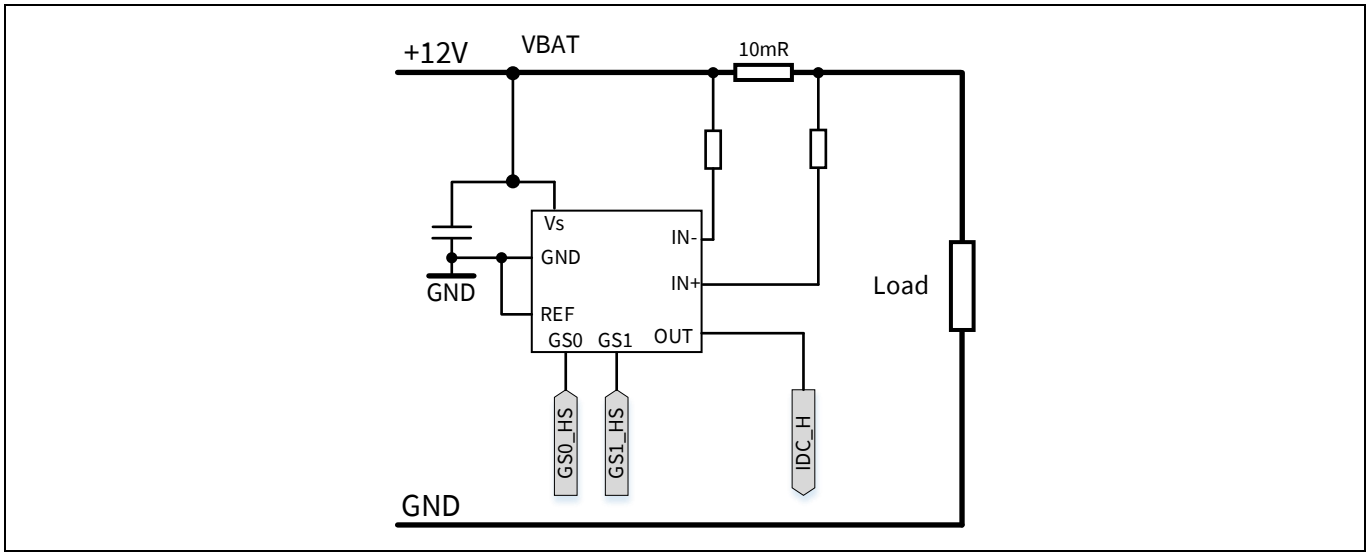

Figure 9 High-side DC-link current feedback signal path

2.6 Space vector modulation

Space Vector Modulation uses the vector representation of the individual leg. Because the low-side switches are complementary to the high-side ones, only high-side states are considered.

The SVM transforms the stator voltage vectors into PWM signals (compare match values) to drive the power switches.

The three phase inverter consists of three half-bridges with a total of 6 switches.

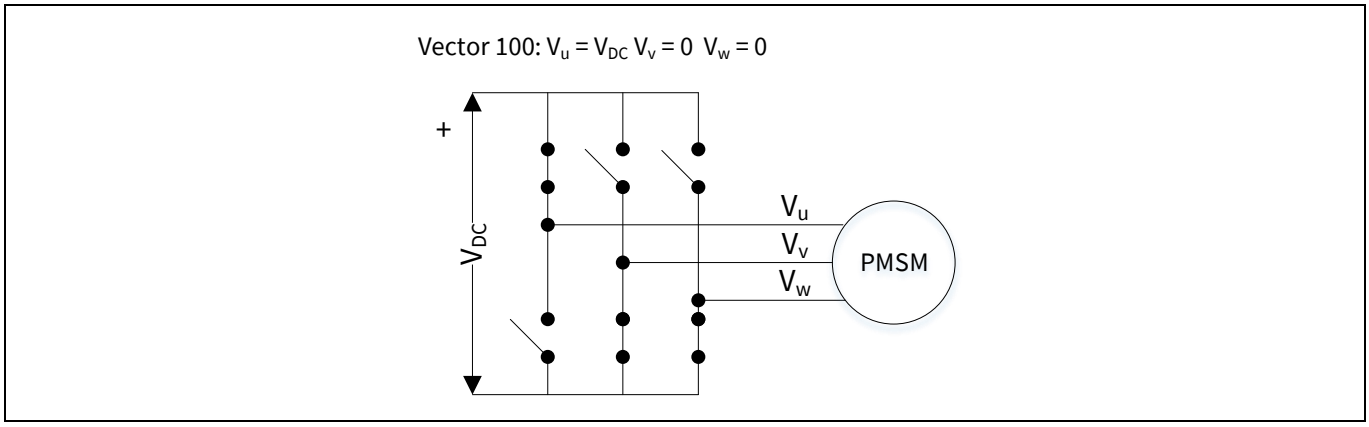

Figure 10 Inverter in state [100], example

The state of the inverter is defined by vector [WVU].

- W represents the state of the high-side switch w with 0 for opened and 1 for closed
- V represents the state of the high-side switch v with 0 for opened and 1 for closed
- U represents the state of the high-side switch u with 0 for opened and 1 for closed

The following 8 possible vectors are obtained:

$$\overrightarrow{V_{wvu}} = [W \quad V \quad U] \qquad \text{Eq. 11}$$

$$\overrightarrow{V_0} = [0 \quad 0 \quad 0] \qquad \text{Eq. 12}$$

$$\overrightarrow{V_1} = [0 \quad 0 \quad 1] \qquad \text{Eq. 13}$$

$$\overrightarrow{V_2} = [0 \quad 1 \quad 0] \qquad \text{Eq. 14}$$

$$\overrightarrow{V_3} = [0 \quad 1 \quad 1] \qquad \text{Eq. 15}$$

$$\overrightarrow{V_4} = [1 \quad 0 \quad 0] \qquad \text{Eq. 16}$$

$$\overrightarrow{V_5} = [1 \quad 0 \quad 1] \qquad \text{Eq. 17}$$

$$\overrightarrow{V_6} = [1 \quad 1 \quad 0] \qquad \text{Eq. 18}$$

$$\overrightarrow{V_7} = [1 \quad 1 \quad 1] \qquad \text{Eq. 19}$$

For the vectors [000] and [111] the stator is either connected to GND or V_{dc}. These vectors are called zero vectors.

All other vectors produce a space vector equal to:

$$\overrightarrow{V_k} = \frac{2}{3} V_{dc} e^{j\frac{(k-1)\pi}{3}} \qquad \text{Eq. 20}$$

where k = {1, 2, 3, 4, 5, 6}.

Arbitrary vector voltage $\overrightarrow{V_R}$, as shown in Figure 11, can be constructed from two nearby vectors $\overrightarrow{V_k}$ and $\overrightarrow{V_{k+1}}$by:

- Using PWM with switching period T_s
- Allocating times T_k and $T_{k+1}$for vector $\overrightarrow{V_k}$ and $\overrightarrow{V_{k+1}}$
- Remaining time for null vector $\overrightarrow{V_0}$ and $\overrightarrow{V_7}$

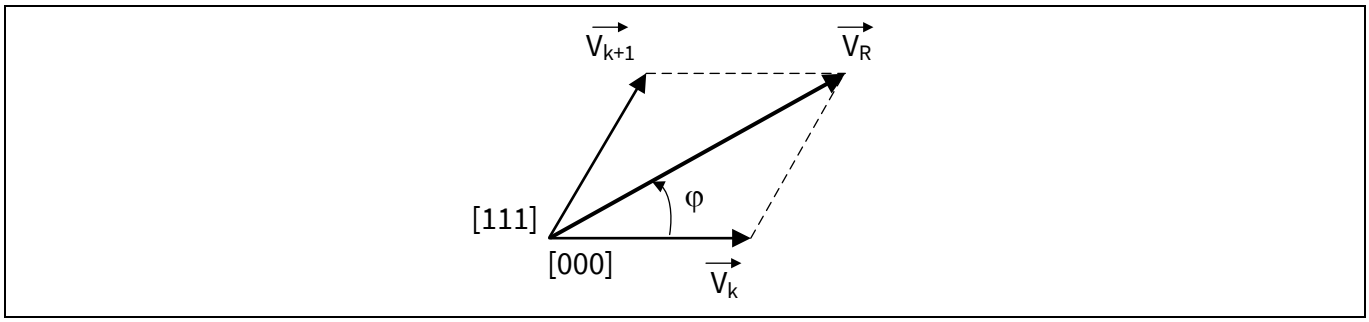

Figure 11 Arbitrary vector generation

Thus, the average of $\overrightarrow{V_R}$ over half of the switching period is described by:

$$\int_0^{\frac{T_s}{2}} \overrightarrow{V_R}\,dt = \int_0^{\frac{T_0}{2}} \overrightarrow{V_0}\,dt + \int_{\frac{T_0}{2}}^{\frac{T_0+T_k}{2}} \overrightarrow{V_k}\,dt + \int_{\frac{T_0+T_k}{2}}^{\frac{T_0+T_k+T_{k+1}}{2}} \overrightarrow{V_{k+1}}\,dt + \int_{\frac{T_0+T_k+T_{k+1}}{2}}^{\frac{T_s}{2}} \overrightarrow{V_7}\,dt \qquad \text{Eq. 21}$$

By definition:

$$T_0 + T_k + T_{k+1} = \frac{T_s}{2} \qquad \text{Eq. 22}$$

Because $\overrightarrow{V_0}$ and $\overrightarrow{V_7}$ are null vectors, and assumed that the average value of $\overrightarrow{V_R}$ is the same as $\overrightarrow{V_R}$ itself, then:

$$\frac{T_s}{2}\overrightarrow{V_R} = T_k\overrightarrow{V_k} + T_{k+1}\overrightarrow{V_{k+1}} \qquad \text{Eq. 23}$$

Previous equation can be expanded with the complex form of vector $\overrightarrow{V_k}$ and$\overrightarrow{V_{k+1}}$:

$$\frac{T_s}{2}\overrightarrow{V_R} = \frac{2}{3}V_{dc}\cdot\left(T_k\cdot e^{j(k-1)\cdot\frac{\pi}{3}} + T_{k+1}\cdot e^{jk\cdot\frac{\pi}{3}}\right) \qquad \text{Eq. 24}$$

or in vector form:

$$\frac{T_s}{2}\begin{bmatrix}V_\alpha\\V_\beta\end{bmatrix} = \frac{2}{3}V_{dc}\cdot\begin{bmatrix}cos\left((k-1)\cdot\frac{\pi}{3}\right) & cos\left(k\cdot\frac{\pi}{3}\right)\\ sin\left((k-1)\cdot\frac{\pi}{3}\right) & sin\left(k\cdot\frac{\pi}{3}\right)\end{bmatrix}\cdot\begin{bmatrix}T_k\\T_{k+1}\end{bmatrix} \qquad \text{Eq. 25}$$

Finally, the vector times T_k and T_{k+1} can be expressed as:

$$\begin{bmatrix} T_k \\ T_{k+1} \end{bmatrix} == \frac{\sqrt{3}}{2}\frac{T_s}{V_{dc}} \cdot \begin{bmatrix} sin\left(k \cdot \frac{\pi}{3}\right) & -cos\left(k \cdot \frac{\pi}{3}\right) \\ -sin\left((k-1) \cdot \frac{\pi}{3}\right) & cos\left((k-1) \cdot \frac{\pi}{3}\right) \end{bmatrix} \cdot \begin{bmatrix} V_\alpha \\ V_\beta \end{bmatrix} \quad \text{Eq. 26}$$

Modulation index vector in this SVM scheme is defined as:

$$\vec{m} = \frac{\pi}{2} \cdot \frac{\overrightarrow{V_R}}{\overrightarrow{V_{dc}}} = \begin{bmatrix} m_\alpha \\ m_\beta \end{bmatrix} \quad \text{Eq. 27}$$

Therefore, using modulation index notation can be written as:

$$\begin{bmatrix} T_k \\ T_{k+1} \end{bmatrix} = \frac{\sqrt{3}}{\pi} \cdot T_s \cdot \begin{bmatrix} sin\left(k \cdot \frac{\pi}{3}\right) & -cos\left(k \cdot \frac{\pi}{3}\right) \\ -sin\left((k-1) \cdot \frac{\pi}{3}\right) & cos\left((k-1) \cdot \frac{\pi}{3}\right) \end{bmatrix} \begin{bmatrix} m_\alpha \\ m_\beta \end{bmatrix} \quad \text{Eq. 28}$$

For the continuous operation of sine wave generation where amplitude of the generated sine wave can be made constant over time, the vector voltage $\overrightarrow{V_R}$ must be within the circle inside the hexagon as shown in Figure 12, which means:

$$\overrightarrow{V_R} = \frac{\sqrt{3}}{\pi} \cdot \left|\overrightarrow{V_k}\right| \quad \text{Eq. 29}$$

The range available for the continuous SVM is represented by the circle inside the hexagon in the Figure 12 and corresponds to a modulation index of 0.906, since the output voltage is limited by the DC-link voltage.

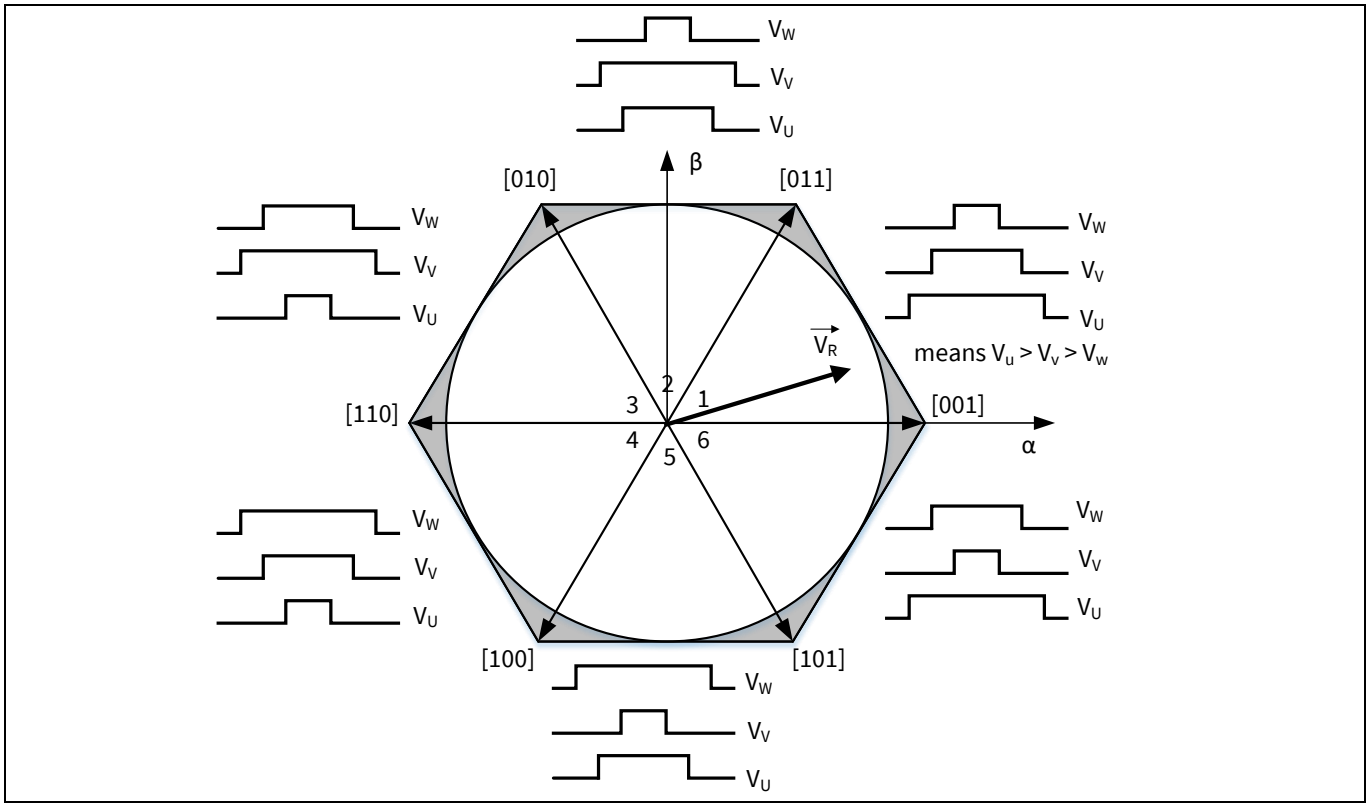

Figure 12 Diagram for Space Vector Modulation

The total hexagon area can still be used for saturated SVM, but introduces more harmonics.

In the continuous domain, the maximum fundamental phase voltage that can be produced by the inverter for a given DC link voltage is:

$$V_{max} = \frac{V_{dc}}{\sqrt{3}} \quad \text{Eq. 30}$$

As an example, a vector $\overrightarrow{V_R}$ which lies in sector 1 is constructed by vector $\overrightarrow{V_1}$ and $\overrightarrow{V_2}$. The timing diagram is shown in Figure 13.

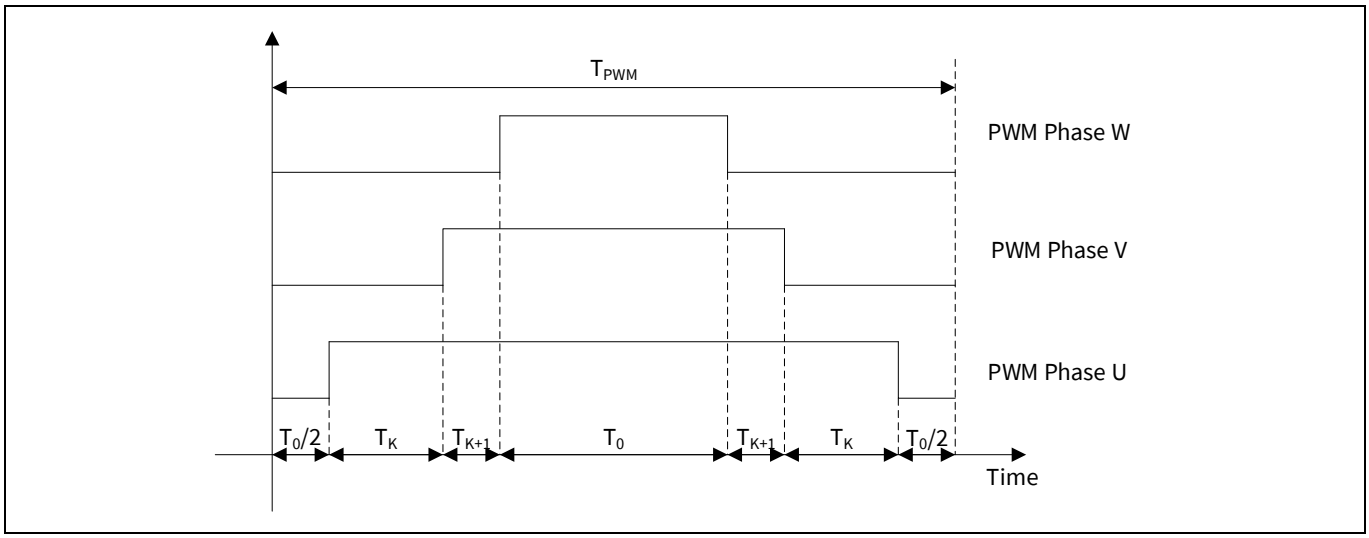

Figure 13 Timing diagram for voltage vector in sector 1

One can calculate the ON time of PWM signals for the phase U, V, and W, denoted T_u,T_v, and T_w, respectively. For voltage vector in sector 1:

$$\begin{bmatrix} T_u \\ T_v \\ T_w \end{bmatrix} = \begin{bmatrix} 2 \cdot (T_k + T_{k+1}) + T_0 \\ 2 \cdot T_{k+1} + T_0 \\ T_0 \end{bmatrix} \quad \text{Eq. 31}$$

The space vector algorithm produces the following phase potentials:

$$V_u(\omega, t) = \begin{cases} \frac{\sqrt{3}}{2} |\overrightarrow{V_R}| \cos\left(\omega t - \frac{\pi}{3}\right), & if\ 0 \le \omega t < \frac{\pi}{3} \\ \frac{3}{2} |\overrightarrow{V_R}| \cos(\omega t), & if\ \frac{\pi}{3} \le \omega t < \frac{2\pi}{3} \\ \frac{\sqrt{3}}{2} |\overrightarrow{V_R}| \cos\left(\omega t + \frac{\pi}{3}\right), & if\ \frac{2\pi}{3} \le \omega t < \pi \\ \frac{\sqrt{3}}{2} |\overrightarrow{V_R}| \cos\left(\omega t - \frac{\pi}{3}\right), & if\ \pi \le \omega t < \frac{4\pi}{3} \\ \frac{3}{2} |\overrightarrow{V_R}| \cos(\omega t), & if\ \frac{4\pi}{3} \le \omega t < \frac{5\pi}{3} \\ \frac{\sqrt{3}}{2} |\overrightarrow{V_R}| \cos\left(\omega t + \frac{\pi}{3}\right), & if\ \frac{5\pi}{3} \le \omega t < 2\pi \end{cases} \quad \text{Eq. 32}$$

$$V_v(\omega t) = V_u(\omega t - \frac{2\pi}{3}) \quad \text{Eq. 33}$$

$$V_w(\omega t) = V_u(\omega t + \frac{2\pi}{3}) \quad \text{Eq. 34}$$

And the following line to line voltages:

$$V_{uv}(\omega t) = \sqrt{3}\left|\overrightarrow{V_R}\sin\left(\omega t + \frac{\pi}{3}\right)\right| \quad \text{Eq. 35}$$

$$V_{vw}(\omega t) = V_{uv}\left(\omega t - \frac{2\pi}{3}\right) \quad \text{Eq. 36}$$

$$V_{wu}(\omega t) = V_{uv}\left(\omega t + \frac{2\pi}{3}\right) \quad \text{Eq. 37}$$

2.7 PWM Generation

The Generic Time Module (GTM) contains many sub-modules. These sub-modules can be combined to form a complex timer module that serves different applications. The term “generic” is applied because of its scalability and configurability. In our application the GTM module is used to generate PWM signals. In our demo application, where three phase current sensing is selected, channels 0 to 7 of TOM0 are used to generate the PWM signals:

- CH0 works as master timer.
- CH1 – CH6 are used to generate the SVPWMs for motor control algorithms.
- CH7 is used to generate the ADC sampling trigger signal.

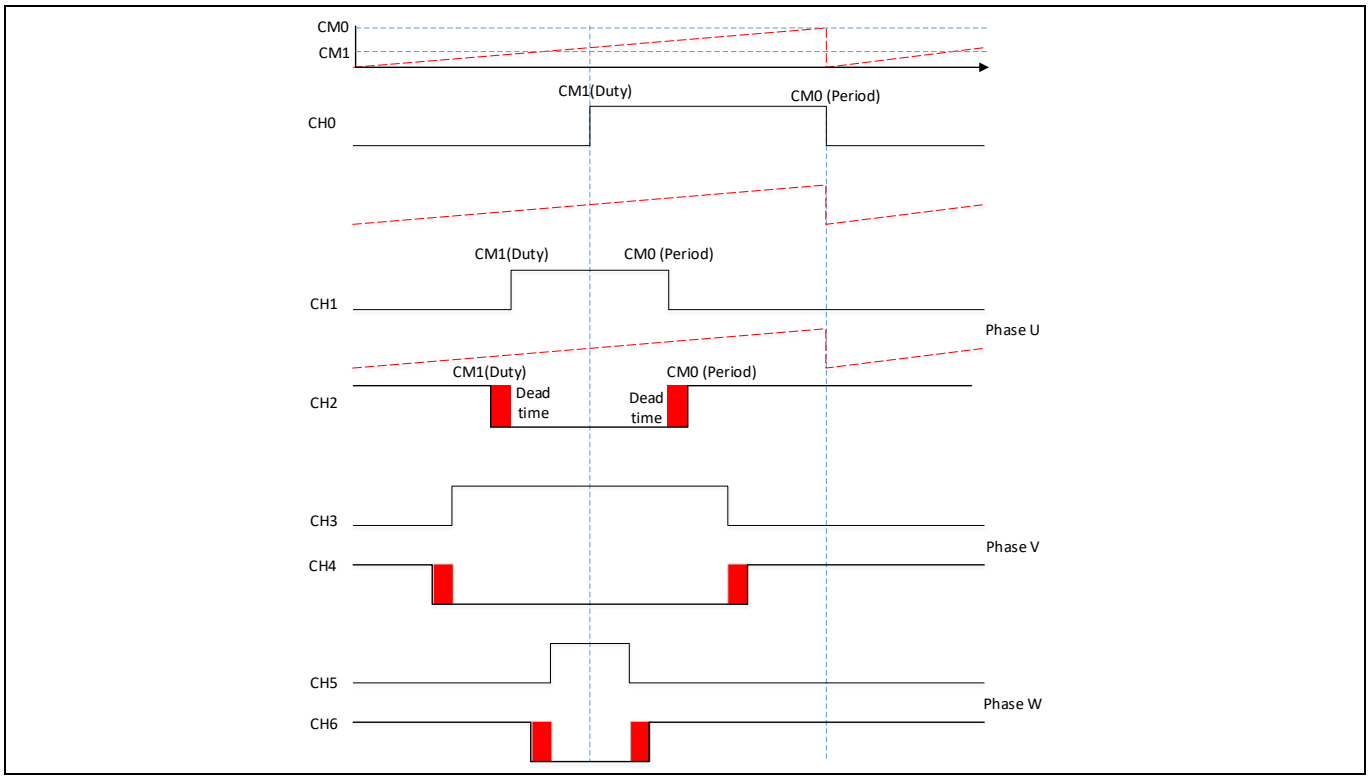

Figure 14 PWM generation with GTM

For more details about PWM generation using GTM refer to [6].

The gate driver TLE9180 processes the PWM signals and adapts the voltage on the gates of the high-side and low-side switches. The TLE9180 takes over polarity inversion, short current protection and advanced diagnostic features. For more information refer to the datasheet of the TLE9180 [5].

2.8 Motor speed and position feedback

An incremental encoder contains LED emitters, integrated circuits with light detectors and output circuitry. A disk with a markings pattern on its surface rotates between the emitter and detector IC, thus allowing and blocking the light of the emitter from reaching the detector IC. The outputs of the detector IC could be single-ended and differential signals. There are three output signals. Two of them provide a square wave signal with a 90 degree phase shift. The third one generates once per revolution a short pulse for synchronization.

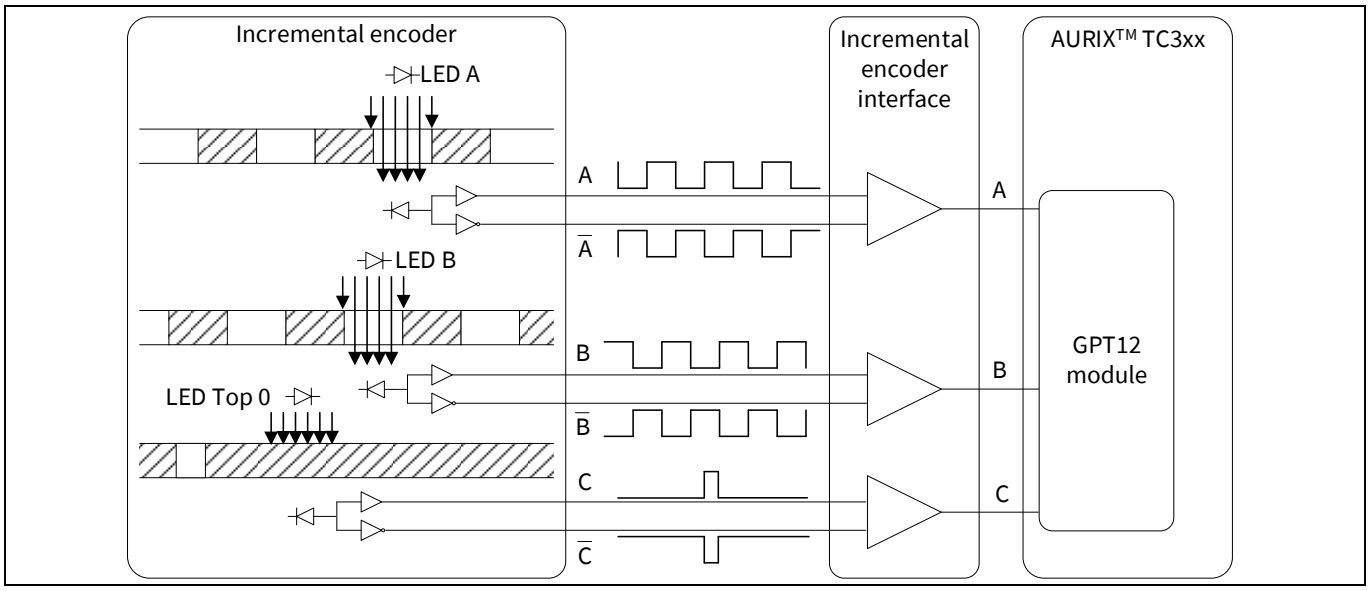

Figure 15 Encoder application

The encoder signal is processed by GPT12. The configuration of the encoder interface is as follows:

- Channel $A/\bar{A}$ is connected with P02.6 (T3INA)
- Channel $B/\bar{B}$ is connected with P02.7 (T3EUDA)
- Channel $C/\bar{C}$ is connected with P02.8 (T4INA)

The offset of each encoder has to be calibrated individually. Encoder resolution and maximal speed are defined as USER_MOTOR_ENCODER_PULSES_PER_REVOLUTION and USER_MOTOR_ENCODER_MAX_RPM, in the user configuration file, PmsmFoc_Motor_Nanotec_DB42S02.h (refer to 3.2.3.2).

2.9 DC-link voltage sensing

The DC-link voltage is directly measured using resistive divider at the power board low power supply connection. The output voltage of resistive divider is measured at every PWM cycle using 12-bit EVADC, triggered by GTM at the middle of PWM period. Measured DC bus voltage is internally represented in float 32-bit format.

The DC-link resistive divider ratio is defined as USER_INVERTER_DC_LINK_DIVIDER_RATIO, in the user configuration file, PmsmFoc_EMotorDrive_v_3_1.h (refer to 3.2.2.2). The phase DC-link voltage could be used in motor control software for monitoring, compensation of space vector modulation, under or over voltage protection.

Note: In current software implementation is used only for monitoring when functionality is enabled.

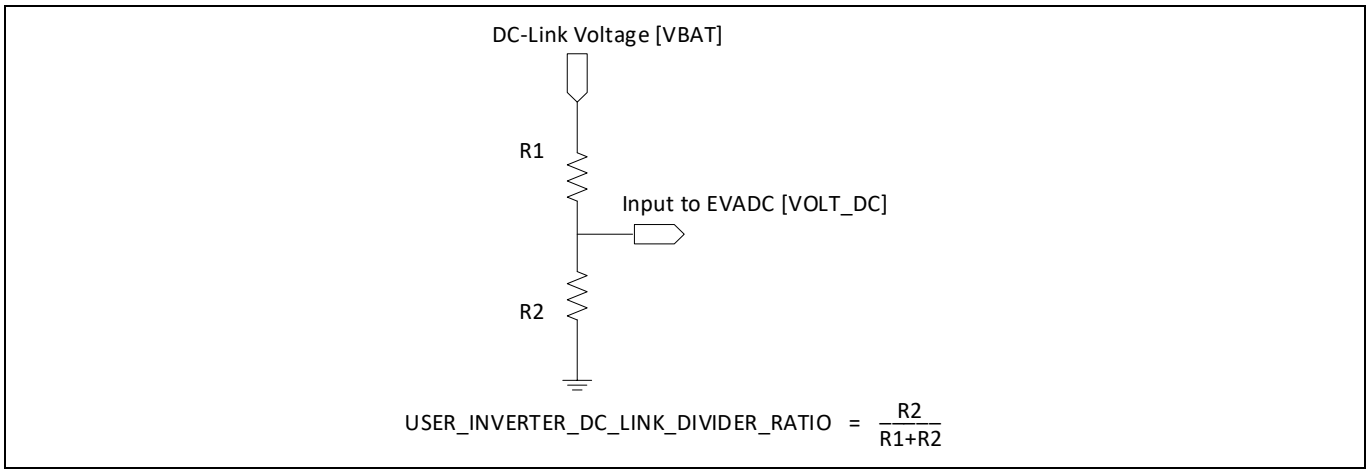

Figure 16 DC-Link voltage feedback signal path

2.10 Phase voltage sensing

The phase voltage is directly measured using resistive dividers at the phases. If functionality is enabled, the output voltage of resistive dividers are measured at every PWM cycle using 12-bit EVADC, triggered by GTM at the middle of PWM period. Measured phase bus voltages are internally represented in float 32-bit format.

The phase voltage resistive divider ratio is defined as USER_INVERTER_BEMF_DIVIDER_RATIO, in the user configuration file, PmsmFoc_EMotorDrive_v_3_1.h (refer to 3.2.2.5). The phase voltage sensing could be used in motor control software for monitoring or in catch spin feature.

Note: In current software implementation it is used only for monitoring when functionality is enabled.

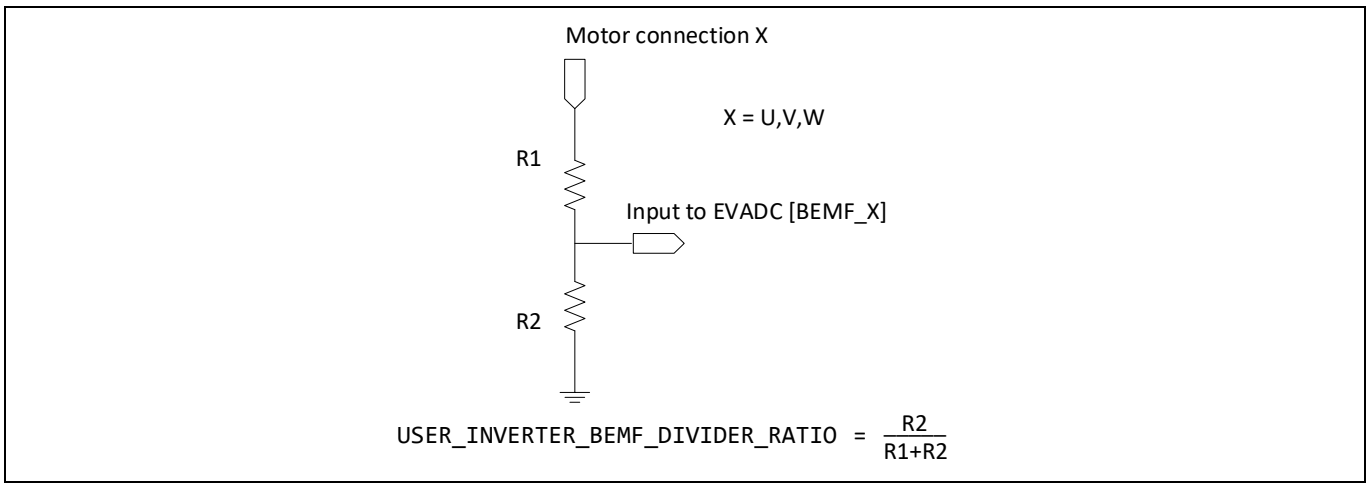

Figure 17 Phase voltage feedback signal path

2.11 Interrupts

Interrupt events and priorities in the PMSM FOC software are listed in the Table 3:

Table 3 Interrupt priorities

Interrupt events	Priorities	Comment
End of phase current sensing conversion	100	State machine execution
End of DC-link voltage sensing conversion	99	Updating DC-link voltage value, when functionality is enabled
End of phase voltage sensing conversion	98	Updating phase voltage values, when functionality is enabled
Incremental encoder top zero position event	20	Incrementing number of rotor turns
Interrupt service units for QSPI2 Transmit	45	TLF35584 QSPI Transmit interrupt handler
Interrupt service units for QSPI2 Receive	46	TLF35584 QSPI Receive Interrupt handler
Interrupt service units for QSPI2 Error	47	TLF35584 QSPI Error Interrupt handler
Interrupt service units for QSPI4 Transmit	50	TLE9180 QSPI Transmit interrupt handler
Interrupt service units for QSPI4 Receive	51	TLE9180 QSPI Receive Interrupt handler
Interrupt service units for QSPI4 Error	52	TLE9180 QSPI Error Interrupt handler

2.12 Motor control state machine

The PMSM FOC software has an internal state machine as shown in Figure 18.

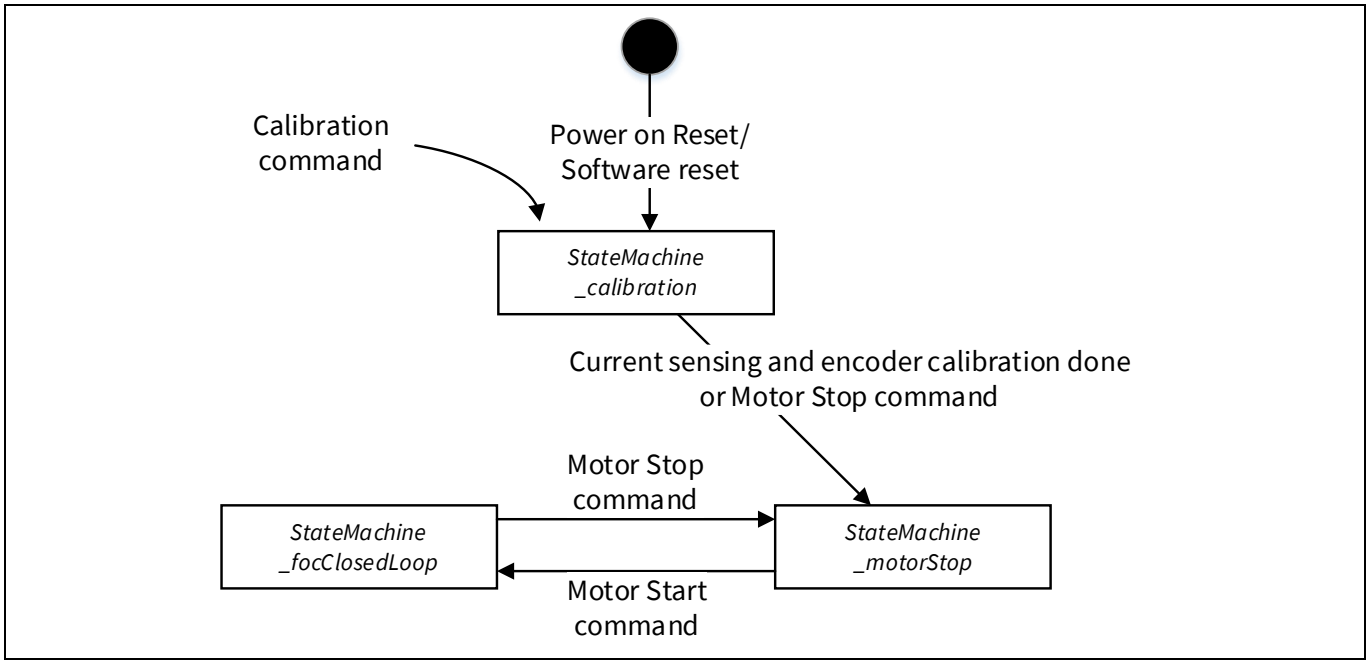

Figure 18 PMSM FOC state machine

StateMachine_motorStop

This state can be entered from all states with the stop command (and other state related conditions). The motor output is set to tristate and the inverter is disabled. This leads to an uncontrolled freewheeling of the motor.

StateMachine_calibration

This state is entered after power on reset or software reset or if calibration command is triggered. In this state current sensing calibration and incremental encoder calibration functions are executed. Exit from this state occurs when the motor stop command is received or if both calibrations are done correctly.

StateMachine_focClosedLoop

In this state the motor is running in FOC mode. The FOC functions are executed. Exit from this state occurs when the motor stop command is received.

The state machine flowchart is shown in Figure 19.

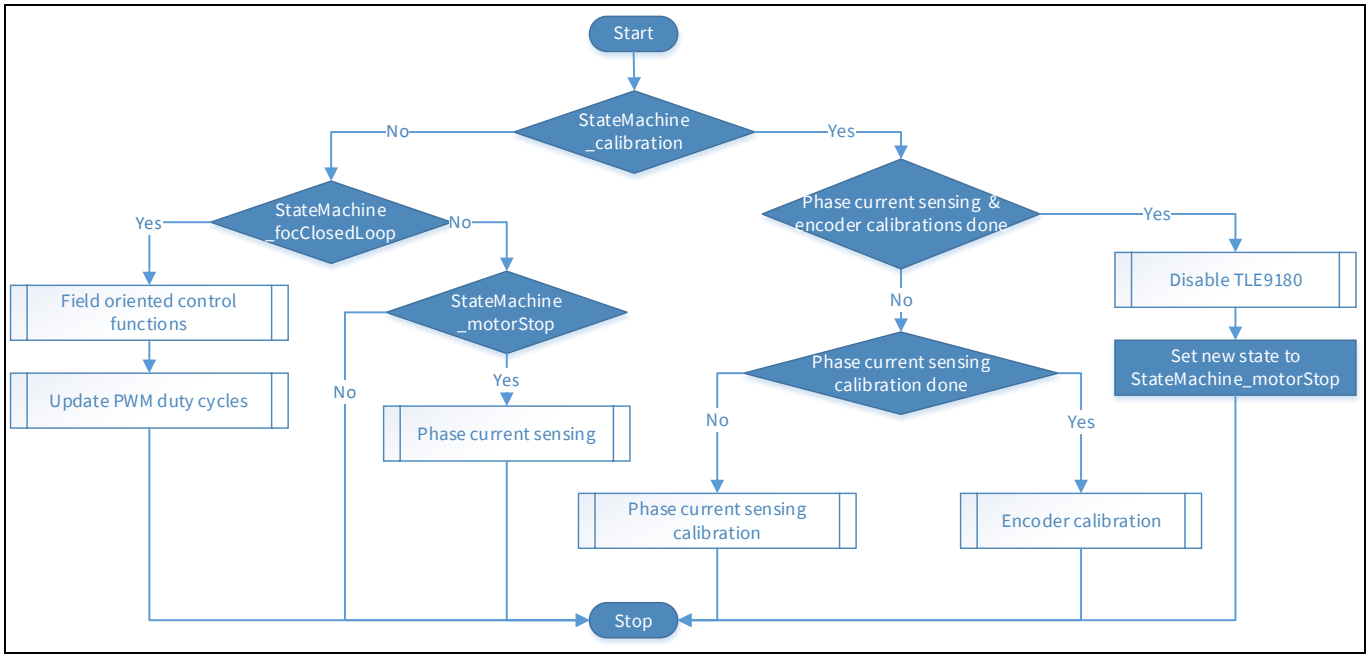

Figure 19 State machine flowchart

The field oriented control functions are executed in *StateMachine_focClosedLoop*. The flowchart is shown in Figure 20.

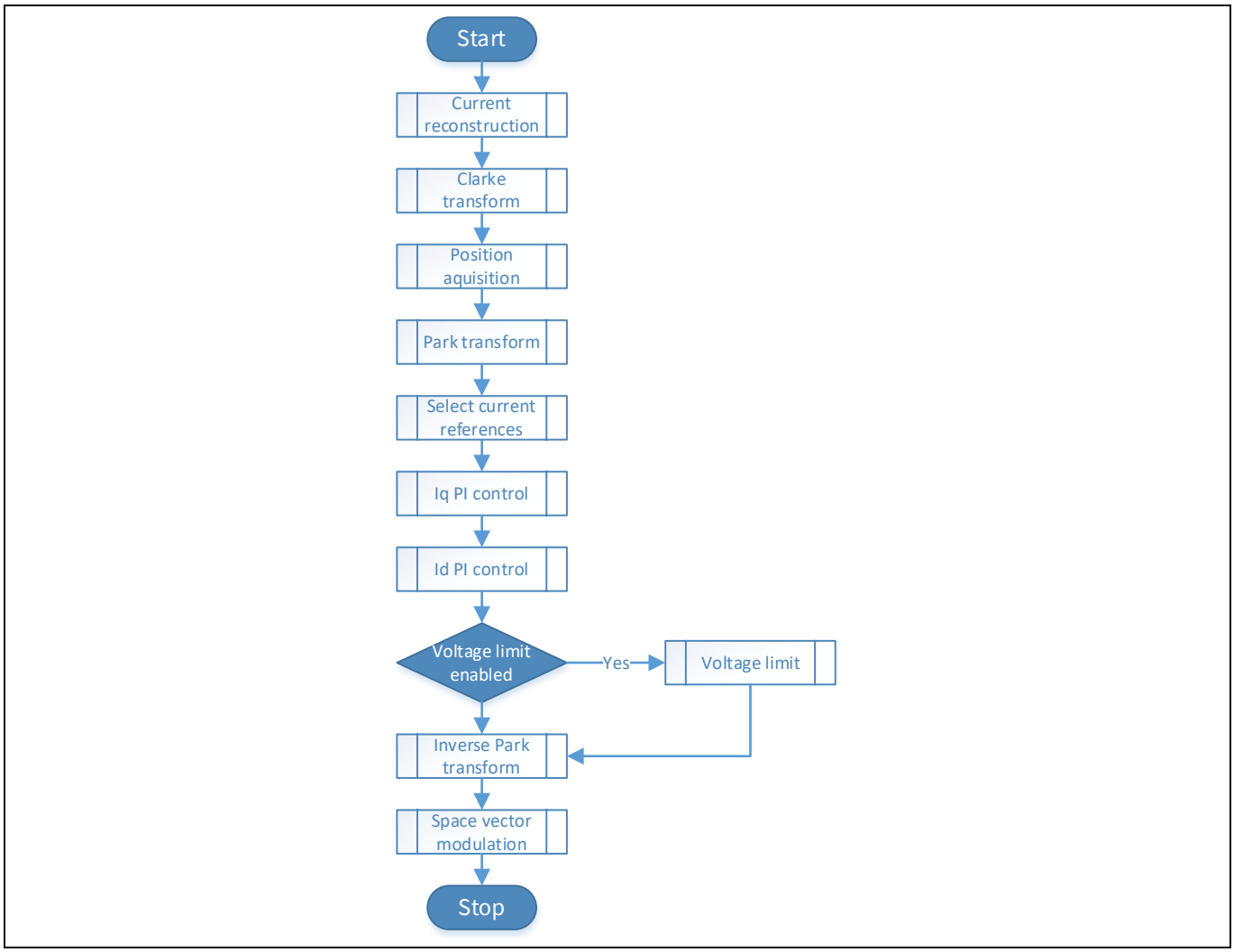

Figure 20 Field oriented control flowchart

2.13 Gate driver support

The TLE9180D-31QK [5], an advanced gate driver IC, is used in motor control power board [4]. The TLE9180D-31QK is dedicated to control 6 external N-channel MOSFETs forming an inverter for high current 3 phase motor drive application in the automotive sector. An integrated QSPI interface is used to configure the TLE9180D-31QK for the application after power-up. After successful power-up, adjusting parameters, monitoring data, configuration and error registers can be read through QSPI interface. Cyclic redundancy check over data and address bits ensures safe communication and data integrity.

Table 4 provides the predefined startup configuration of TLE9180D-31QK. More details about available registers can be found in [5].

Table 4 TLE9180D startup configuration

Register short name	Data [hex]	Description
Conf_Sig	AC	CRC8 Signature Byte
Conf_Gen_1	81	140°C, Input Pattern Supervision Disabled, SPI Window Watchdog Disabled, Limp Home Mode Activation Disabled, VCC Supervision Enabled, VCC Monitoring Threshold - 5V selected as VCC supply voltage

Register short name	Data [hex]	Description
Conf_Gen_2	0F	5V Overcurrent Detection Threshold, 0V BSx enabled, 0V LD enabled, 0V SD VDHP enabled, 3 VDH sense pins and 1 VDHP power pin enabled, Current Sense Amplifier 3 enabled, Current Sense Amplifier 2 enabled, Current Sense Amplifier and Reference Buffer enabled
Tl_vdh	70	48.18V VDHP Overvoltage Threshold, 3.96V VDHP Undervoltage Threshold
Tl_cbvcc	9A	9.99V CB Undervoltage Threshold, VCC Overvoltage Threshold is 10% of configured VCC supply voltage, VCC Undervoltage Threshold is 10% of configured VCC supply voltage
Fm_1	32	CB Undervoltage Failure Behavior - Warning, Overload Charge Pump 2 Failure Behavior - Shutdown of output stages, Undervoltage High-side Buffer Capacitor Failure Behavior - Auto Restart Error
Fm_3	2A	Vs Undervoltage Failure - Auto Restart Error, VDHP Undervoltage Failure Behavior - Auto Restart Error, VCC Undervoltage Failure Behavior - Auto Restart Error
Fm_4	4A	Vs Overvoltage Failure - Auto Restart Error, VDHP Overvoltage Failure Behavior - Auto Restart Error, VCC Overvoltage Failure Behavior - Auto Restart Error
Fm_6	2A	Current Sense Amplifier 3 Overcurrent Failure Behavior - ARE, Current Sense Amplifier 2 Overcurrent Failure Behavior - ARE, Current Sense Amplifier 1 Overcurrent Failure Behavior - ARE,
Op_gain_1	44	Current Sense Amplifier 1 Gain 1 - 30.81, Current Sense Amplifier 2 Gain 1 - 30.81
Op_gain_2	44	Current Sense Amplifier 1 Gain 2 - 30.81, Current Sense Amplifier 2 Gain 2 - 30.81
Op_gain_3	44	Current Sense Amplifier 3 Gain 2 - 30.81, Current Sense Amplifier 3 Gain 1 - 30.81
Op_0cl	9F	Zero Current Output Voltage Offset - 2.5V, Zero Current Output Voltage Offset Fine Adjustment - No adjustment

3 Configuration

The default configuration of the parameters in the PMSM FOC software is set based on the AURIX™ TC3xx Motor Control Application Kit. The configuration is split up in to 3 levels:

- User configuration
  - Allows fast access to the general configuration such as the FOC scheme and selection of one of the pre-configured, purchasable hardware boards.
- Hardware configuration
  - Allows for specialization of the controller card, inverter card, and motors. With this configuration you can adapt the hardware configuration to your application and board. Configurations such as pinout and maximum speed can be found here.
- FOC configuration
  - Allows you to change basic values and constant calculations. One example is the over-voltage definition of 120% of the DC link voltage. Most of this configurations are essential, calculated, and should not be changed by the user, which is why they are not described in this document.

3.1 User configuration

The user configuration allows for fast access to the general configuration such as the FOC scheme and selection of one of the pre-configured, hardware boards. All configuration options are available in the file PmsmFoc_UserConfig.h.

The default settings are for the “Nanotec motor DB42S02” used in the Infineon AURIX™ TC3xx Motor Control Application Kit, KIT_A2G_TC387_MOTORCTRL.

3.1.1 General

PMSM FOC Hardware Kit

Various configuration options are available for the software. The configurations are mainly independent from each other. The hardware consists of:

- Controller_Card (MCUCARD_TYPE)
- Inverter_Card (INVERTERCARD_TYPE)
- Motor (MOTOR_TYPE)

In this document a combination of all three is referred to as a:

- Hardware Kit (KIT_A2G_TC387_MOTORCTRL)

Additionally it is possible to exchange parts of the hardware board. To support these options you can select a custom hardware kit and adapt the MCUCARD_TYPE, INVERTERCARD_TYPE, and MOTOR_TYPE, and the associated paths.

```
#define PMSM_FOC_HARDWARE_KIT                    KIT_A2G_TC387_MOTORCTRL
```

  - Select a pre-defined hardware kit.

Options:

  - `KIT_A2G_TC387_MOTORCTRL` – Infineon AURIX™ TC387 Motor Control Application Kit
  - `CUSTOM_KIT` – User defined motor control system

Configuration

Current sensing

```
#define PHASE_CURRENT_RECONSTRUCTION
       USER_LOWSIDE_THREE_SHUNT_WITH_HIGHSIDE_MONITORING
```

- Defines the current sensing technique used.

Options:

- `USER_LOWSIDE_THREE_SHUNT_WITH_HIGHSIDE_MONITORING` – Three shunt current sensing technique with ADC standard conversion, with high-side DC-link current sensing
- `USER_LOWSIDE_THREE_SHUNT_WITHOUT_HIGHSIDE_MONITORING` – Three shunt current sensing technique with ADC synchronous conversion, without high-side DC-link current sensing

FOC control schematic

```
#define FOC_CONTROL_SCHEME                    SPEED_CONTROL
```

- Defines the FOC control scheme.

Options:

- `SPEED_CONTROL` – Speed controlled FOC, refer to 2.1.1.
- `CURRENT_CONTROL` – Current controlled FOC, refer to 2.1.2.

3.1.2 Custom Kit configuration

Controller_Card

```
#define MCUCARD_TYPE                          defined by PMSM_FOC_HARDWARE_KIT
#define MCUCARD_TYPE_PATH                     defined by PMSM_FOC_HARDWARE_KIT
```

- In the default configuration this is defined by the hardware board. To configure a custom MCU card or combination, the hardware board CUSTOM_KIT should be used. Only the hardware board combinations are tested.

Options for MCUCARD_TYPE:

- `CUSTOM_MCU`
- `APP_KIT_TFT_TC387A`

Note: It is necessary to adapt the MCUCARD_TYPE_PATH according to the selection.

Inverter_Card

```
#define INVERTERCARD_TYPE                     defined by PMSM_FOC_HARDWARE_KIT
#define INVERTERCARD_TYPE_PATH                defined by PMSM_FOC_HARDWARE_KIT
```

- In the default configuration this is defined by the hardware board. To configure a custom inverter card or combination, the hardware board CUSTOM_KIT should be used. Only the hardware board combinations are tested.

Options for INVERTERCARD_TYPE:

- `CUSTOM_INVERTER`
- `EMOTOR_DRIVE_V_3_1`

Note: It is necessary to adapt the INVERTERCARD_TYPE_PATH according to the selection

Motor

```
#define MOTOR_TYPE                              defined by PMSM_FOC_HARDWARE_KIT
#define MOTOR_TYPE_PATH                         defined by PMSM_FOC_HARDWARE_KIT
```

- In the default configuration this is defined by the hardware board. To configure a custom motor or combination the hardware board CUSTOM_KIT should be used. Only the hardware board combinations are tested.
- `NANOTEC_MOTOR_DB42S02`

Note: It is necessary to adapt the MOTOR_TYPE_PATH according to the selection.

3.1.3 Advanced user configuration

FOC control

```
#define DQ_DECOUPLING                                   ENABLED
```

- Enables or disables the dq-axis decoupling feature.

3.2 Hardware configuration

The detailed hardware configuration allows specialization of the controller card, inverter card, and motors. With this configuration you can adapt the hardware configuration to your application and board. Configurations such as pinout and maximum speed can be found here.

The configuration is separated in to three types:

- PMSM_FOC\Configuration\Controller_Card
- PMSM_FOC\Configuration\Inverter_Card
- PMSM_FOC\Configuration\Motors

3.2.1 Controller card

3.2.1.1 Interrupt priorities

```
#define INTERRUPT_PRIORITY_ENCODER_GPT12 (20)
```

- Defines the encoder zero position interrupt priority

```
#define INTERRUPT_PRIORITY_QSPI2_TX      (45)
```

- Defines the QSPI2 transmit interrupt priority

```
#define INTERRUPT_PRIORITY_QSPI2_RX      (46)
```

- Defines the QSPI2 receive interrupt priority

```
#define INTERRUPT_PRIORITY_QSPI2_ERR     (47)
```

- Defines the QSPI2 error interrupt priority

```
#define INTERRUPT_PRIORITY_QSPI4_TX      (50)
```

- Defines the QSPI4 transmit interrupt priority

```
#define INTERRUPT_PRIORITY_QSPI4_RX      (51)
```

- Defines the QSPI4 receive interrupt priority

```
#define INTERRUPT_PRIORITY_QSPI4_ERR     (52)
```

- Defines the QSPI4 error interrupt priority

```
#define INTERRUPT_PRIORITY_EVADC_HSCUR    (97)
```

- Defines the EVADC group conversion, used for high-side DC-link current sensing

```
#define INTERRUPT_PRIORITY_EVADC_VBEMF    (98)
```

- Defines the EVADC group conversion, used for phase voltage sensing

```
#define INTERRUPT_PRIORITY_EVADC_VDCL     (99)
```

- Defines the EVADC group conversion, used for DC-link voltage sensing

```
#define INTERRUPT_PRIORITY_EVADC_CUR      (100)
```

- Defines the EVADC group conversion, used for phase current sensing

3.2.1.2 PWM outputs (GTM configuration)

```
#define PHASE_U_HS                &IfxGtm_TOM1_2_TOUT12_P00_3_OUT
```

- Defines the phase U high-side PWM output

```
#define PHASE_U_LS                &IfxGtm_TOM1_1_TOUT11_P00_2_OUT
```

- Defines the phase U high-side PWM output

```
#define PHASE_V_HS                &IfxGtm_TOM1_4_TOUT14_P00_5_OUT
```

- Defines the phase V high-side PWM output

```
#define PHASE_V_LS                &IfxGtm_TOM1_3_TOUT13_P00_4_OUT
```

- Defines the phase V high-side PWM output

```
#define PHASE_W_HS                &IfxGtm_TOM1_6_TOUT16_P00_7_OUT
```

- Defines the phase W high-side PWM output

```
#define PHASE_W_LS                &IfxGtm_TOM1_5_TOUT15_P00_6_OUT
```

- Defines the phase W low-side PWM output

3.2.1.3 Incremental encoder (GPT12 configuration)

```
#define ENCODER_GPT12_PIN_A       &IfxGpt120_T3INA_P02_6_IN
```

- Defines the encoder pin A

```
#define ENCODER_GPT12_PIN_B       &IfxGpt120_T3EUDA_P02_7_IN
```

- Defines the encoder pin B

```
#define ENCODER_GPT12_PIN_Z       &IfxGpt120_T4INA_P02_8_IN
```

- Defines the encoder pin Z (top zero)

```
#define ENCODER_REVERSED          TRUE
```

- Defines the encoder sensor direction is reversed

```
#define ENCODER_RESOLUTION        USER_MOTOR_ENCODER_PULSES_PER_REVOLUTION
```

- Defines the the encoder resolution

```
#define ENCODER_OFFSET            USER_MOTOR_ENCODER_OFFSET
```

- Defines the default encoder offset

```
#define ENCODER_UPDATE_PERIOD     (1/ ((float32) USER_INVERTER_PWM_FREQ_HZ))
```

- Defines the encoder function update period

```
#define ENCODER_SPEED_MODE_THRESHOLD
                    2 * IFX_PI * USER_MOTOR_SPEED_PULSE_COUNTING_RPM / 60
```

- Defines the threshold for pulse counting method

```
#define ENCODER_BASE_MIN_SPEED   (float32) (USER_MOTOR_SPEED_LOW_LIMIT_RPM
                                 / 60.0 * 2 * IFX_PI)
```

- Defines the minimum base speed

```
#define ENCODER_BASE_MAX_SPEED   (float32) (USER_MOTOR_SPEED_HIGH_LIMIT_RPM /
                                 60.0 * 2 * IFX_PI)
```

- Defines the maximum base speed

```
#define ENCODER_GPT12_HOST_CPU IfxSrc_Tos_cpu0
```

- Defines the GPT12 host CPU

3.2.1.4 TLE9180 configuration (GPIO and QSPI)

```
#define TLE9180_ENABLE_PIN       &IfxPort_P33_11
```

- Defines the Enable Pin GPIO

```
#define TLE9180_ERROR_PIN        &IfxPort_P15_2
```

- Defines the TLE9180 Error Pin GPIO

```
#define TLE9180_INHIBIT_PIN      &IfxPort_P20_0
```

- Defines the TLE9180 Inhibit Pin GPIO

```
#define TLE9180_SAFEOFF_PIN      &IfxPort_P33_10
```

- Defines the TLE9180 Safe Off Pin GPIO

```
#define TLE9180_SPI_CS_PIN       IfxQspi4_SLSO3_P22_2_OUT
```

- Defines the QSPI slave select pin

```
#define TLE9180_SPI_MODULE       MODULE_QSPI4
```

- Defines the QSPI object

```
#define TLE9180_SPI_CLOCK_PIN    IfxQspi4_SCLK_P22_3_OUT
```

- Defines the QSPI SLCK out pin

```
#define TLE9180_SPI_MOSI_PIN     IfxQspi4_MTSR_P22_0_OUT
```

- Defines the QSPI MTSR out pin

```
#define TLE9180_SPI_MISO_PIN     IfxQspi4_MRSTB_P22_1_IN
```

- Defines the QSPI MRSTA input pin

```
#define TLE9180_SPI_HOST_CPU     IfxSrc_Tos_cpu0
```

- Defines the handler of the interrupts

```
#define TLE9180_SPI_USE_DMA      FALSE
```

- Defines the use of DMA for Data transfer/s

```
#define TLE9180_SPI_TX_DMA_CH    IfxDma_ChannelId_none
```

- Defines the DMA channel no for the QSPI transmit

```
#define TLE9180_SPI_RX_DMA_CH    IfxDma_ChannelId_none
```

- Defines the DMA channel no for the QSPI receive

3.2.1.5 TLF35584 configuration (GPIO and QSPI)

```
#define TLF35584_SPI_CS_PIN      IfxQspi2_SLSO1_P14_2_OUT
```

- Defines the QSPI slave select pin

```
#define TLF35584_SPI_MODULE      MODULE_QSPI2
```

- Defines the QSPI object

```
#define TLF35584_SPI_CLOCK_PIN   IfxQspi2_SCLK_P15_3_OUT
```

- Defines the QSPI SLCK out pin

```
#define TLF35584_SPI_MOSI_PIN    IfxQspi2_MTSR_P15_6_OUT
```

- Defines the QSPI MTSR out pin

```
#define TLF35584_SPI_MISO_PIN    IfxQspi2_MRSTB_P15_7_IN
```

- Defines the QSPI MRSTA input pin

```
#define TLF35584_SPI_HOST_CPU    IfxSrc_Tos_cpu0
```

- Defines the handler of the interrupts

```
#define TLF35584_SPI_USE_DMA     FALSE
```

- Defines the use of DMA for data transfer/s

```
#define TLF35584_SPI_TX_DMA_CH   IfxDma_ChannelId_none
```

- Defines the DMA channel no for the QSPI transmit

```
#define TLF35584_SPI_RX_DMA_CH   IfxDma_ChannelId_none
```

- Defines the DMA channel no for the QSPI receive

```
#define INA225AIDGK_GS0          &IfxPort_P33_2
```

- Defines the INA225AIDGK gain selection pin 0 GPIO

```
#define INA225AIDGK_GS1          &IfxPort_P33_4
```

- Defines the INA225AIDGK gain selection pin 1 GPIO

3.2.2 Inverter board configuration

3.2.2.1 General

```
#define USER_INVERTER_PWM_FREQ_HZ              (20000U)
```

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

```
#define USER_INVERTER_MAX_ADC_VDD_V            (5.0f)
```

- This value defines the maximum input of the EVADC in volts. The AURIX™ family has a wide input range. Common is 5V and 3.3V. The input value is in float. Most physical scaling's are linked to this define

3.2.2.2 Supply voltage and supply voltage sensing

```
#define USER_INVERTER_VDC_LINK_V               (12.0f)
```

- This macro defines the nominal DC voltage in volts, used in the motor inverter board. It is used for scaling

```
#define USER_INVERTER_DC_LINK_DIVIDER_RATIO  (float) (5.6f/(5.6f+56.0f))
```

- See 2.4. The DC link voltage divider ratio is R2/(R1+R2)
- This value influences the scaling of the ADC results

```
#define USER_INVERTER_DCLINVOLTAGESENSE_GAIN
(USER_INVERTER_MAX_ADC_VDD_V / (4096 * USER_INVERTER_DC_LINK_DIVIDER_RATIO))
```

- Defines the DC-link voltage sensing scaled gain value

```
#define USER_INVERTER_DCLINVOLTAGESENSE_OFFSET                    (0.0)
```

- Defines the DC-link voltage sensing offset initial value

```
#define USER_INVERTER_DCLINVOLTAGESENSE_OFFSET_NOM                (0)
```

- Defines the DC-link voltage sensing offset nominal value

3.2.2.3 Output, Bridge Driver and B6 Bridge

```
#define USER_INVERTER_PWM_RISING_DEADTIME_SEC              (0.0000010f)
#define USER_INVERTER_PWM_FAILING_DEADTIME_SEC             (0.0000010f)
```

- This macros defines the dead-time in microseconds. This values has to be defined according to the switches and bridge drivers. A too small value leads to a short cut. A high 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 to the applied voltage

3.2.2.4 Current sensing

```
#define USER_INVERTER_THREE_PHASE_SHUNT_OHM                (0.01f)
```

- Current shunt resistance in ohm. This value is used to calculate USER_INVERTER_I_MAX_A which is later used to limit the reference current

```
#define USER_INVERTER_DC_LOW_SIDE_SHUNT_OHM                (0.01f)
```

- Low-side DC-link current shunt resistance in ohms

```
#define USER_INVERTER_DC_HIGH_SIDE_SHUNT_OHM               (0.01f)
```

- High-side DC-link current shunt resistance in ohms

```
#define USER_INVERTER_PHASECURSENSE_AMPLIFIER_GAIN         (30.81)
```

- Defines the selected TLE9180 amplifier gains

```
#define USER_INVERTER_I_MAX_A
      ((USER_INVERTER_MAX_ADC_VDD_V/ (USER_INVERTER_THREE_PHASE_SHUNT_OHM
      * USER_INVERTER_PHASECURSENSE_AMPLIFIER_GAIN))/2)
```

- Defines the maximum phase current value measured by using selected phase shunts and selected amplifier gain values

```
#define USER_INVERTER_PHASECURSENSE_I_MIN     (-USER_INVERTER_I_MAX_A)
```

- Defines the minimum phase current value

```
#define USER_INVERTER_PHASECURSENSE_I_MAX     (USER_INVERTER_I_MAX_A)
```

- Defines the maximum phase current value

```
#define USER_INVERTER_PHASECURSENSE_I_GAIN
            (-(USER_INVERTER_MAX_ADC_VDD_V/
            (4096*USER_INVERTER_PHASECURSENSE_AMPLIFIER_GAIN *
            USER_INVERTER_THREE_PHASE_SHUNT_OHM)))
```

- Defines the phase current sense gain value

```
#define USER_INVERTER_PHASECURSENSE_I_OFFSET
```

```
            (2.5 * USER_INVERTER_PHASECURSENSE_AMPLIFIER_GAIN *
            USER_INVERTER_THREE_PHASE_SHUNT_OHM))
```

- Defines the phase current sense offset value

```
#define USER_INVERTER_PHASECURSENSE_V_GAIN USER_INVERTER_MAX_ADC_VDD_V/4096
```

- Defines the phase current sense amplifier reference voltage gain value

```
#define USER_INVERTER_PHASECURSENSE_V_OFFSET                    (0.0)
```

- Defines the phase current sense amplifier reference voltage offset value

```
#define USER_INVERTER_PHASECURSENSE_V_OFFSET_NOM                (2048)
```

- Defines the phase current sense amplifier reference voltage nominal offset value

```
#define USER_INVERTER_PHASECURSENSE_REFILL_TRIG
      (1 << IFX_EVADC_G_Q_QINR_RF_OFF) | (1 << IFX_EVADC_G_Q_Q0R_EXTR_OFF)
```

- Defines the phase current sense refill trigger value

```
#define USER_INVERTER_PHASECURSENSE_CALIBRATION_COUNT            (2048)
```

- Defines the number of loops used for current sensing calibration

```
#define USER_INVERTER_HS_DCLINK_CUR_SENSE_DEFAULT_GAIN          (200.0f)
```

- Defines the high-side DC-link current sensing selected gain value

```
#define USER_INVERTER_HS_DCLINK_CUR_SENSE_GAIN
      ((USER_INVERTER_MAX_ADC_VDD_V/ (4096
      * USER_INVERTER_HS_DCLINK_CUR_SENSE_DEFAULT_GAIN
      * USER_INVERTER_DC_HIGH_SIDE_SHUNT_OHM)))
```

- Defines the high-side DC-link current sensing scaled gain value

```
#define USER_INVERTER_HS_I_MAX_A
      ((USER_INVERTER_MAX_ADC_VDD_V/ (USER_INVERTER_DC_HIGH_SIDE_SHUNT_OHM
      * USER_INVERTER_HS_DCLINK_CUR_SENSE_DEFAULT_GAIN))/2)
```

- Defines the maximum DC-link current value measured by using single shunt and default amplifier gain value

```
#define USER_INVERTER_HS_DCLINK_CUR_SENSE_OFFSET                (0.0)
```

- Defines the high-side DC-link current sensing offset initial value

```
#define USER_INVERTER_HS_DCLINK_CUR_SENSE_OFFSET_NOM            (0.0)
```

- Defines the high-side DC-link current sensing offset nominal value

```
#define USER_INVERTER_HS_DCLINK_CUR_SENSE_I_MIN   (-USER_INVERTER_HS_I_MAX_A)
```

- Defines the high-side DC-link current sensing minimum value

```
#define USER_INVERTER_HS_DCLINK_CUR_SENSE_I_MAX   (USER_INVERTER_HS_I_MAX_A)
```

- Defines the high-side DC-link current sensing maximum value

3.2.2.5 Phase voltage sensing

```
#define USER_INVERTER_BEMF_DIVIDER_RATIO      (float) (5.6f/ (5.6f+56.0f))
```

- Defines the R1/(R2+R1) ratio for BEMF Voltage sensing circuit ratio

```
#define USER_INVERTER_BEMFSENSE_GAIN
(USER_INVERTER_MAX_ADC_VDD_V /(4096 * USER_INVERTER_BEMF_DIVIDER_RATIO))
```

- Defines the phase voltage sensing scaled gain value

```
#define USER_INVERTER_BEMFSENSE_OFFSET                    (0.0)
```

- Defines the phase voltage sensing offset initial value

```
#define USER_INVERTER_BEMFSENSE_OFFSET_NOM                (0)
```

- Defines the phase voltage sensing offset nominal value

3.2.3 Motor specific configuration

3.2.3.1 Motor parameters

```
#define USER_MOTOR_RESISTANCE_PHASE_TO_PHASE              (0.19f)
```

- Defines the motor phase to phase resistance in ohms

```
#define USER_MOTOR_INDUCTANCE_PHASE_TO_PHASE              (0.245e-3)
```

- Defines the motor phase to phase stator inductance in henry

```
#define USER_MOTOR_POLE_PAIR                              (4U)
```

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

```
#define USER_MOTOR_WIRING_CONNECTION                      DELTA
```

- Defines the Motor wiring Connection. Motor winding connection can be DELTA or WYE

```
#define USER_MOTOR_RESISTANCE_PER_PHASE
      (USER_MOTOR_RESISTANCE_PHASE_TO_PHASE*3/2)
```

- Defines the motor resistance per phase in Ohm in case that motor winding connection is DELTA

```
#define USER_MOTOR_INDUCTANCE_PER_PHASE
      (USER_MOTOR_INDUCTANCE_PHASE_TO_PHASE*3/2)
```

- Defines the motor inductance per phase in H in case that motor winding connection is DELTA

```
#define USER_MOTOR_RESISTANCE_PER_PHASE
      (USER_MOTOR_RESISTANCE_PHASE_TO_PHASE/2)
```

- Defines the motor resistance per phase in Ohm in case that motor winding connection is WYE

```
#define USER_MOTOR_INDUCTANCE_PER_PHASE
      (USER_MOTOR_INDUCTANCE_PHASE_TO_PHASE/2)
```

- Defines the motor inductance per phase in H in case that motor winding connection is WYE

3.2.3.2 Incremental encoder configuration

```
#define USER_MOTOR_SPEED_PULSE_COUNTING_RPM               (1200.0f)
```

- Defines the speed point when switching from one to another position sensing method

```
#define USER_MOTOR_ENCODER_PULSES_PER_REVOLUTION          (1000U)
```

- Defines the incremental encoder pulses per revolution value

```
#define USER_MOTOR_ENCODER_MAX_RPM                        (6000.0f)
```

- Defines the incremental encoder maximum speed value

3.2.3.3 Speed limits

```
#define USER_MOTOR_SPEED_LOW_LIMIT_RPM                    (1.0f)
```

- Defines the motor minimum speed limit value in RPM

```
#define USER_MOTOR_SPEED_HIGH_LIMIT_RPM      (USER_MOTOR_ENCODER_MAX_RPM)
```

- Defines the motor maximum speed limit value in RPM, based on maximum speed of used incremental encoder

3.2.3.4 Current PI controllers settings

```
#define USER_MOTOR_KPD_MULTI                                    (0.50f)
```

- Defines the D-axis current regulator Kp parameter adjustment value
- Empirical value, based on tests

```
#define USER_MOTOR_KID_MULTI                                    (1.0f)
```

- Defines the D-axis current regulator Ki parameter adjustment value
- Empirical value, based on tests

```
#define USER_MOTOR_KPQ_MULTI                                    (0.5f)
```

- Defines the Q-axis current regulator Kp parameter adjustment value
- Empirical value, based on tests

```
#define USER_MOTOR_KIQ_MULTI                                    (1.0f)
```

- Defines the Q-axis current regulator Ki parameter adjustment value
- Empirical value, based on tests

```
#define USER_MOTOR_CURRENT_PI_CONTROLLER_BANDWITH          (1000.0f)
```

- Defines the Bandwidth of the PI Controller [Hz]

```
#define USER_MOTOR_FOC_CALC_TSTATOR
       (USER_MOTOR_INDUCTANCE_PER_PHASE)/ (USER_MOTOR_RESISTANCE_PER_PHASE)
```

- Defines the stator time constant

```
#define USER_MOTOR_FOC_K_FACTOR
       (1.0/USER_INVERTER_VDC_LINK_V*IFX_TWO_OVER_PI))
```

- Defines the DQ-axis current controller parameter scaling value

```
#define USER_MOTOR_FOC_CALC_KI
       (USER_MOTOR_RESISTANCE_PER_PHASE)/
       (1/ (2*IFX_PI*USER_MOTOR_CURRENT_PI_CONTROLLER_BANDWITH))
       *USER_MOTOR_FOC_K_FACTOR
```

- Defines the DQ-axis current controller Ki parameter value

```
#define USER_MOTOR_FOC_CALC_KP
       (USER_MOTOR_FOC_CALC_KI*USER_MOTOR_FOC_CALC_TSTATOR)
```

- Defines the DQ-axis current controller Kp parameter value

```
#define USER_MOTOR_PI_ID_KP
       (USER_MOTOR_KPD_MULTI*(USER_MOTOR_FOC_CALC_KP))
```

- Defines the D-axis current regulator adjusted Kp parameter value

```
#define USER_MOTOR_PI_ID_KI
       (USER_MOTOR_KID_MULTI*(USER_MOTOR_FOC_CALC_KI))
```

- Defines the D-axis current regulator adjusted Ki parameter value

```
#define USER_MOTOR_PI_IQ_KP
       (USER_MOTOR_KPQ_MULTI*(USER_MOTOR_FOC_CALC_KP))
```

- Defines the Q-axis current regulator Kp parameter value

```
#define USER_MOTOR_PI_IQ_KI
      (USER_MOTOR_KIQ_MULTI*(USER_MOTOR_FOC_CALC_KI))
```

- Defines the Q-axis current regulator Ki parameter value

```
#define USER_MOTOR_PI_ID_LIMIT_MAX                    (0.85f)
```

- Defines the D-axis current regulator max limit value

```
#define USER_MOTOR_PI_IQ_LIMIT_MAX                    (0.85f)
```

- Defines the Q-axis current regulator max limit value

```
#define USER_MOTOR_PI_ID_CONTROL_PERIOD               (50e-6)
```

- Defines the D-axis current regulator control period value

```
#define USER_MOTOR_PI_IQ_CONTROL_PERIOD               (50e-6)
```

- Defines the Q-axis current regulator control period value

```
#define USER_MOTOR_IQLIMIT                            (5.0f)
```

- Defines the max limit value of Q-axis current reference

```
#define USER_MOTOR_IDLIMIT                            (4.0f)
```

- Defines the max limit value of D-axis current reference

3.2.3.5 Speed PI controllers settings

```
#define USER_MOTOR_SPEED_KP_MULTI                     (0.000005f)
```

- Defines the speed regulator Kp parameter adjustment value
- Empirical value, based on tests

```
#define USER_MOTOR_SPEED_KI_MULTI                     (0.00004f)
```

- Defines the speed regulator Kp parameter adjustment value
- Empirical value, based on tests

```
#define USER_MOTOR_SPEED_CONTROL_KP
      (USER_MOTOR_SPEED_KP_MULTI*(USER_MOTOR_FOC_CALC_KP))
```

- Defines the Kp parameter value

```
#define USER_MOTOR_SPEED_CONTROL_KI
      (USER_MOTOR_SPEED_KI_MULTI*(USER_MOTOR_FOC_CALC_KI))
```

- Defines the Ki parameter value

```
#define USER_MOTOR_SPEED_CONTROL_MIN                  (-0.3f)
```

- Defines the PI controller output minimum value in relative units

```
#define USER_MOTOR_SPEED_CONTROL_MAX                  (1.0f)
```

- Defines the PI controller output maximum value in relative units

```
#define USER_MOTOR_SPEED_CONTROL_MAX_RPM    (USER_MOTOR_SPEED_HIGH_LIMIT_RPM)
```

- Defines the maximum speed in RPM

```
#define USER_MOTOR_SPEED_CONTROL_MIN_RPM              (200.0f)
```

- Defines the minimum speed in RPM

```
#define USER_MOTOR_SPEED_CONTROL_PERIOD               (5e-3)
```

- Defines the controller period in seconds

```
#define USER_MOTOR_SPEED_CONTROL_REF                    (0.0f)
```

- Defines the initial motor speed reference value. Unit is RPM or 1/min

3.2.3.6 Speed and dq-axis current references ramp settings

```
#define USER_MOTOR_SPEED_RAMP_SLEW_RATE                 (500.0f)
```

- Defines the maximum speed slew rate, value per second

```
#define USER_MOTOR_SPEED_RAMP_PERIOD                    (50e-3f)
```

- Defines the sampling period of the speed ramp function

```
#define USER_MOTOR_CURRENT_D_RAMP_SLEW_RATE             (100e-3f)
```

- Defines the maximum d-axis current slew rate, value per second

```
#define USER_MOTOR_CURRENT_D_RAMP_PERIOD                (5e-3f)
```

- Defines the sampling period of the d-axis current ramp function

```
#define USER_MOTOR_CURRENT_Q_RAMP_SLEW_RATE             (100e-3f)
```

- Defines the maximum q-axis current slew rate, value per second

```
#define USER_MOTOR_CURRENT_Q_RAMP_PERIOD                (5e-3f)
```

- Defines the sampling period of the q-axis current ramp function

4 PMSM FOC software data structure

Motor control module data structure defines the main structure used in the PMSM FOC application.

Table 5 Main motor control data structure

Category (Structure)	Sub-category (Sub-structure)	Description
MotorControl	Interfaces	User interface
	ControlParameters	Contains control parameters information
	Diagnostic	Contains diagnostic information
	Inverter	Contains inverter information
	PmsmFoc	Contains field oriented control information
	PositionAcquisition	Contains position sensor information
	MotorParameters	Contains motor parameters

5 PMSM FOC software API functions

In this chapter used or implemented PMSM FOC software APIs are listed.

Table 6 List of APIs

Layer	API Function name
ControlModules	PmsmFoc_initMotorControl (...)
	PmsmFoc_initControlVariables (...)
	PmsmFoc_doFieldOrientedControl (...)
	PmsmFoc_reconstructCurrent (...)
	PmsmFoc_doClarkeTransform (...)
	PmsmFoc_doParkTransform (...)
	PmsmFoc_setIdqRef (...)
	PmsmFoc_doIdControl (...)
	PmsmFoc_doIqControl (...)
	PmsmFoc_getVdqMagnitude (...)
	PmsmFoc_doInverseParkTransform (...)
	PmsmFoc_doSpeedControl (...)
	PmsmFoc_doEncoderCalibration (...)
	PmsmFoc_resetEncoderCalibrationStatus (...)
	PmsmFoc_doSpeedRefLinearRamp (...)
	PmsmFoc_doCurrentRefLinearRamp (...)
	PmsmFoc_getSpeedRefLinearRamp (...)
	PmsmFoc_getCurrentRefLinearRamp (...)
	PmsmFoc_setSpeedRefLinearRamp (...)
	PmsmFoc_setCurrentRefLinearRamp (...)
	PmsmFoc_Interface_setMotorTargetSpeed (...)
	PmsmFoc_Interface_startMotor (...)
	PmsmFoc_Interface_stopMotor (...)
	PmsmFoc_Interface_setDemo (...)
	PmsmFoc_Interface_doDemo (...)
	PmsmFoc_SpeedControl_init (...)
	PmsmFoc_SpeedControl_do (...)
	PmsmFoc_SpeedControl_reset (...)
	PmsmFoc_SpeedControl_enable (...)
	PmsmFoc_SpeedControl_isEnabled (...)
	PmsmFoc_SpeedControl_disable (...)
	PmsmFoc_SpeedControl_resetLimitFlag (...)
	PmsmFoc_SpeedControl_setLimit (...)
	PmsmFoc_SpeedControl_setMaxSpeed (...)
	PmsmFoc_SpeedControl_setRefSpeed (...)

PMSM FOC software API functions

Layer	API Function name
	PmsmFoc_SpeedControl_getMaxSpeed (...)
	PmsmFoc_SpeedControl_getRefSpeed (...)
	PmsmFoc_SpeedControl_getPi (...)
	PmsmFoc_SpeedControl_setKpKi (...)
	PmsmFoc_SpeedControl_getSpeed (...)
	PmsmFoc_StateMacine_doControlLoop (...)
ExtDevInit	PmsmFoc_Tle9180_Init (...)
	PmsmFoc_Tle9180_loadConfiguration (...)
	PmsmFoc_Tle9180_InitSpiChannel (...)
	PmsmFoc_Tlf35584_Init (...)
	PmsmFoc_Tlf35584_InitSpiChannel (...)
Interrupts	PmsmFoc_Evadc_PhaseCurSense_Isr (...)
	PmsmFoc_Evadc_CurrentDCLinkSenseHs_Isr (...)
	PmsmFoc_Evadc_BemfVoltageSense_Isr (...)
	PmsmFoc_Evadc_DcLinkVoltageSense_Isr (...)
	PmsmFoc_Gpt12_Encoder_TzIsr (...)
	PmsmFoc_Qspi_Tlf35584_TxIsr (...)
	PmsmFoc_Qspi_Tlf35584_RxIsr (...)
	PmsmFoc_Qspi_Tlf35584_ErrIsr (...)
	PmsmFoc_Qspi_Tle9180_TxIsr (...)
	PmsmFoc_Qspi_Tle9180_RxIsr (...)
	PmsmFoc_Qspi_Tle9180_ErrIsr (...)
Libraries	N.A.
MCUInit	PmsmFoc_Evadc_initEvadc (...)
	PmsmFoc_Evadc_initModule (...)
	PmsmFoc_Evadc_initGroups (...)
	PmsmFoc_Evadc_initCurrentSenseChannels (...)
	PmsmFoc_Evadc_initBemfVoltageSenseChannels (...)
	PmsmFoc_Evadc_initDcLinkVoltageSenseChannels (...)
	PmsmFoc_Evadc_initGroup0 (...)
	PmsmFoc_Evadc_initGroup1 (...)
	PmsmFoc_Evadc_initGroup2 (...)
	PmsmFoc_Evadc_initGroup3 (...)
	PmsmFoc_Evadc_initGroup4 (...)
	PmsmFoc_Evadc_initGroup0Queue0 (...)
	PmsmFoc_Evadc_initGroup1Queue0 (...)
	PmsmFoc_Evadc_initGroup1Queue1 (...)
	PmsmFoc_Evadc_initGroup1Queue2 (...)
	PmsmFoc_Evadc_initGroup2Queue0 (...)

PMSM FOC software API functions

Layer	API Function name
	PmsmFoc_Evadc_initGroup2Queue1 (...)
	PmsmFoc_Evadc_initGroup2Queue2 (...)
	PmsmFoc_Evadc_initGroup3Queue0 (...)
	PmsmFoc_Evadc_initGroup3Queue1 (...)
	PmsmFoc_Evadc_initGroup3Queue2 (...)
	PmsmFoc_Evadc_initGroupXQueue0CurrentSenseTriShuntHsMon (...)
	PmsmFoc_Evadc_initGroupXQueue1VoltageSenseDcLink (...)
	PmsmFoc_Evadc_initGroupXQueue2VoltageSenseBemf (...)
	PmsmFoc_Gpt12_initGpt12 (...)
	PmsmFoc_Gtm_initGtm (...)
	PmsmFoc_Gtm_initTom (...)
	PmsmFoc_Gtm_initTriggerChannel (...)
	PmsmFoc_Gtm_initTrigToEvadcCurSense (...)
	PmsmFoc_Gtm_initTrigToEvadcVoltageSense (...)
	PmsmFoc_initHardware (...)
	PmsmFoc_Qspi_initQspi (...)
	PmsmFoc_Qspi_initQspi2 (...)
	PmsmFoc_Qspi_initQspi4 (...)
MidSys	PmsmFoc_CurrentDCLinkSenseHs_init (...)
	PmsmFoc_CurrentDCLinkSenseHs_doCalibration (...)
	PmsmFoc_CurrentDCLinkSenseHs_setGain (...)
	PmsmFoc_CurrentDCLinkSenseHs_resetCalibrationStatus (...)
	PmsmFoc_CurrentDCLinkSenseHs_getCalibrationStatus (...)
	PmsmFoc_CurrentDCLinkSenseHs_limitsExecute (...)
	PmsmFoc_CurrentDCLinkSenseHs_updateAnalogInput (...)
	PmsmFoc_CurrentDCLinkSenseHs_getRawCurrentValue (...)
	PmsmFoc_PhaseCurrentSense_init (...)
	PmsmFoc_PhaseCurrentSense_doCalibration (...)
	PmsmFoc_PhaseCurrentSense_limitsExecute (...)
	PmsmFoc_PhaseCurrentSense_updateAnalogInputDRC (...)
	PmsmFoc_PhaseCurrentSense_updateAnalogInput (...)
	PmsmFoc_PhaseCurrentSense_getRawPhaseCurrentValues (...)
	PmsmFoc_PhaseCurrentSense_resetCalibrationStatus (...)
	PmsmFoc_PhaseCurrentSense_getCalibrationStatus (...)
	PmsmFoc_pwm3PhaseOutput_getPeriod (...)
	PmsmFoc_pwm3PhaseOutput_getDeadTime (...)
	PmsmFoc_PositionAcquisition_init (...)
	PmsmFoc_PositionAcquisition_updatePosition (...)
	PmsmFoc_PositionAcquisition_updateSpeed (...)

PMSM FOC software API functions

Layer	API Function name
	PmsmFoc_doPwmSvmUpdate (...)
	PmsmFoc_doSvPwmModulation (...)
	PmsmFoc_SensorAdc_init (...)
	PmsmFoc_DcLinkVoltageSense_init (...)
	PmsmFoc_DcLinkVoltageSense_updateAnalogInput (...)
	PmsmFoc_DcLinkVoltageSense_getValue (...)
	PmsmFoc_DcLinkVoltageSense_resetCalibrationStatus (...)
	PmsmFoc_BemfVoltageSense_init (...)
	PmsmFoc_BemfVoltageSense_updateAnalogInput (...)
	PmsmFoc_BemfVoltageSense_getValue (...)
	PmsmFoc_BemfVoltageSense_resetCalibrationStatus (...)

6 References

[1] AURIX™ TC3xx User Manual, V1.5.0

[2] Nanotec Datasheet Brushless DC Motor DB42S02

[3] Application Kit TC3X7 User Manual, V1.0

[4] AURIX™ TC3xx Motor Control Power Board, V1.0

[5] Datasheet of TLE9180D-31QK, V1.0

[6] AP32356 PWMAC with AURIX™ TC3xx devices, V1.0

Revision history

Document version	Date of release	Description of changes
V1.0	September 2020	Initial version

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

Edition 2020-09-16

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
AP32540

IMPORTANT NOTICE

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

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

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

WARNINGS

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

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