

# XC800 Family

## AP08090

Sensorless Field Oriented Control (FOC) on XC878

### Application Note

V 1.0, 2009-04

**Edition 2009-04**

**Published by  
Infineon Technologies AG  
81726 Munich, Germany**

**© 2009 Infineon Technologies AG  
All Rights Reserved.**

#### **LEGAL DISCLAIMER**

THE INFORMATION GIVEN IN THIS APPLICATION NOTE IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED 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.

#### **Information**

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office ([www.infineon.com](http://www.infineon.com)).

#### **Warnings**

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

---

**XC878**

**Revision History: V 1.0 2009-04**

Previous Version(s):

Page	Subjects (major changes since last revision)
–	This is the first release

Trademarks

TriCore® is a trademark of Infineon Technologies AG.

**We Listen to Your Comments**

Is there any information in this document that you feel is wrong, unclear or missing? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:

[mcdocu.comments@infineon.com](mailto:mcdocu.comments@infineon.com)



## Table of Contents

	<b>Table of Contents</b> .....	4
<b>1</b>	<b>Target Market and Motivation</b> .....	6
<b>2</b>	<b>Theory of Sensorless Field Oriented Control (FOC)</b> .....	7
2.1	DC Motor Control .....	7
2.1.1	Open Loop Voltage Control .....	8
2.1.2	Closed Loop Torque Control .....	8
2.1.3	Closed Loop Speed and Torque Control .....	9
2.1.4	List of Equations .....	9
2.2	Space Vector Modulation .....	10
2.2.1	Reference Vector .....	11
2.3	Phase Current Measurement .....	14
2.3.1	Limitations of Phase Current Reconstruction .....	14
2.4	FOC Calculations .....	15
2.4.1	Stationary and Rotating Reference Frames .....	15
2.4.2	Sensorless FOC .....	17
<b>3</b>	<b>FOC Implementation on XC878</b> .....	19
3.1	Program Flow .....	20
3.2	Initialization .....	21
3.2.1	CC6 Timer Unit .....	21
3.2.2	ADC .....	21
3.2.3	CAN .....	22
3.2.4	T21 Timer .....	22
3.2.5	Application .....	22
3.3	Scheduler State Machine - T21 Interrupt .....	23
3.3.1	State IDLE .....	23
3.3.2	State BOOTST .....	23
3.3.3	State STARTUP .....	24
3.3.4	State FOC .....	25
3.3.5	State RAMPDWN .....	25
3.3.6	State STOP .....	26
3.3.7	State EMERGENCY .....	26
3.4	CAN Communication - CAN Interrupts .....	26
3.5	FOC Control Loop - T13 Interrupt .....	27
3.5.1	Shadow Transfer .....	27
3.5.2	FOC Calculation .....	27
3.5.3	Flux Estimator .....	29
3.5.4	PI-Controller with Anti Wind-Up Limitation .....	30
3.5.5	Speed Ramp .....	31
<b>4</b>	<b>How to Use the Setup</b> .....	32
4.1	Excel Sheet and FOC_Defines.h .....	32
4.2	Project Settings .....	33
4.3	Toolchain .....	34
4.4	DriveMonitor .....	35
4.4.1	Configuration .....	35
4.4.2	Monitoring the Startup Behavior .....	35
4.4.3	Monitoring the Runtime Behavior .....	37
4.4.4	Special Tricks - Field Weakening .....	38

<b>5</b>	<b>FOC Building Blocks Overview Drawing</b> .....	<b>39</b>
<b>6</b>	<b>References</b> .....	<b>40</b>

## 1 Target Market and Motivation

The Field Oriented Control (FOC) method is focussed on cost sensitive consumer drives, such as fans, pumps, compressors or geared motors.

One driving factor for a sensorless FOC on BLDC (Brush Less DC motor) or PMSM (Permanent Magnet Synchronous Motors) is decreasing the system cost: sensors can generate problems in manufacturing and reliability, and they require special wiring harnesses and connectors which increase the overall system cost.

The other driving factor is the significant improvement of energy efficiency of a motor. With FOC, efficiency can be raised up to 95%. This has a big impact on power consumption, motor dynamics, heat dissipation and noise.

Fans and pumps in many applications must operate very silently, otherwise any sound produced distributes in water or air pipes within the whole building. Usually this leads to concepts with sinusoidal currents in the motor coils. The zero-crossing technique for position detection of BLDC motors does not work, because block-type currents generate audible noise.

The FOC method described here solves the two obstacles at once:

- no need for a hall sensor or any other sensor type
- sinusoidal current shape for very efficient and quiet operation



**Figure 1 Target Applications for Sensorless Field Oriented Control (FOC)**

The limitation of the methods described in this application note is the single shunt current measurement which cannot guarantee a steady control under all circumstances. This method is therefore not suitable for high dynamic loads and positioning systems. However the algorithm can be adapted to 3-phase-shunt measurement which does not have this limitation.



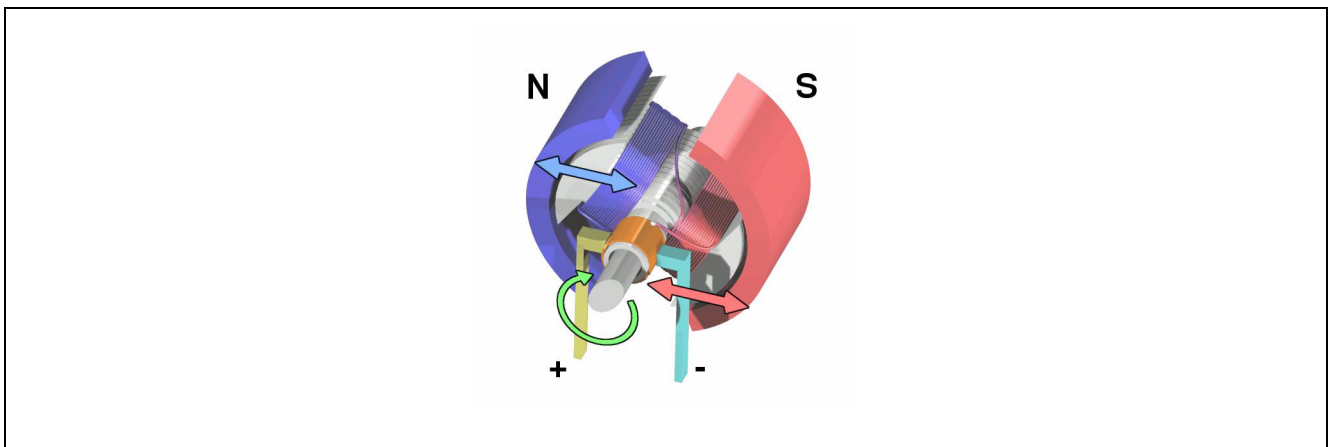
## 2 Theory of Sensorless Field Oriented Control (FOC)

This chapter is intended to give an overview about the theory of sensorless Field Oriented Control (FOC) of Permanent Magnet Synchronous Motors (PMSM). The control scheme which has been implemented for the XC878 and XE164 is based on this theory. The implementation details are described in [Chapter 3](#).

Before the FOC theory is described in detail, this chapter starts with some background motor control description.

### 2.1 DC Motor Control

The DC motor consists of a permanent magnet which builds the stator and a coil mounted at the rotor. A mechanical commutator is needed in order to feed the current to the coils. This commutator will cause the current to be oriented according to the field of the permanent magnets at the stator.



**Figure 2 DC Motor**

The DC Motor is controlled by the voltage at the rotor coils and the current is proportional to the torque of the motor. The speed of the motor depends on the voltage and the torque.

The nominal speed ( $\omega_{nom}$ ) and torque ( $m_{nom}$ ) are defined in the following equations:

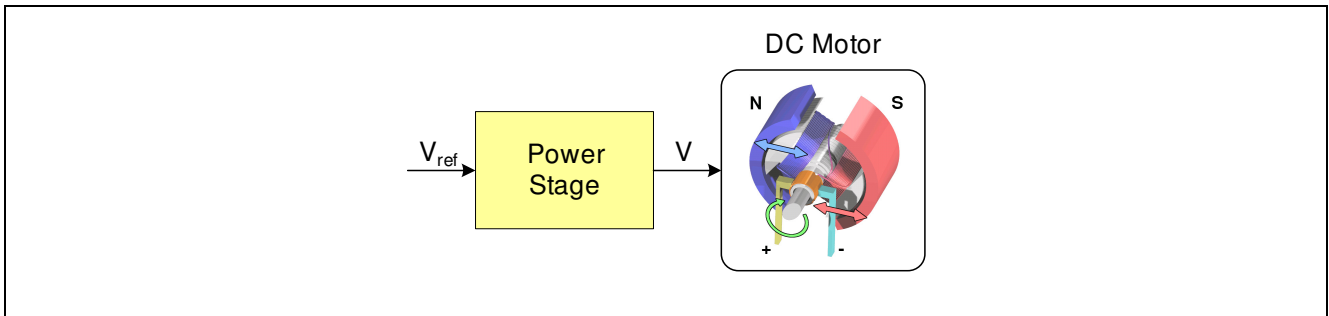
$$\omega_{nom} = \frac{v - i \cdot R}{K_c} \quad (1)$$

$$m_{nom} = K_c \cdot i \quad (2)$$

The time constants of speed and current control are very different, but a control cascade structure for speed and current control can be utilized.

### 2.1.1 Open Loop Voltage Control

In an open loop voltage control there is simply a reference voltage which will cause the power inverter to generate a given voltage at the motor. The mechanical load influences the speed and the current of the DC motor.



**Figure 3 Open Loop Voltage Control**

In steady state, the torque can be described as follows:

$$m_{\text{stat}} = K_c \cdot \frac{v - K_c \omega_{\text{stat}}}{R} \quad (3)$$

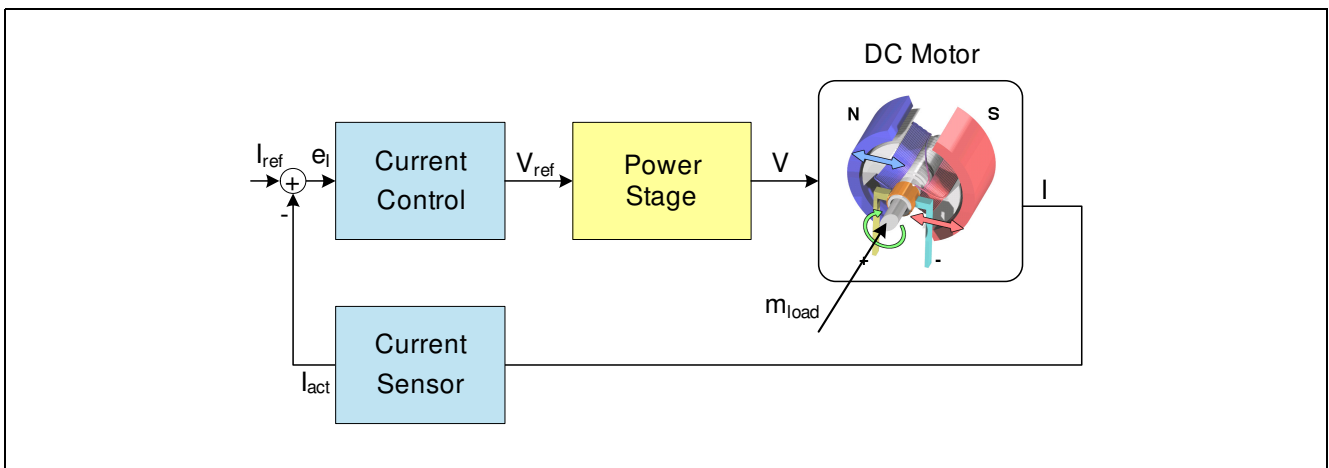
### 2.1.2 Closed Loop Torque Control

The torque control loop requires the measurement of the motor current with a current sensor; e.g. a shunt resistor.

In general, the torque is proportional to the current:

$$m_m = K_c \cdot i \quad (4)$$

As a result, a PI controller can be used for current control. The actual current measured by the current sensor, will be controlled to converge to the reference current. This can only be done by changing the reference Voltage ( $V_{\text{ref}}$ ) of the power stage. See the following figure:



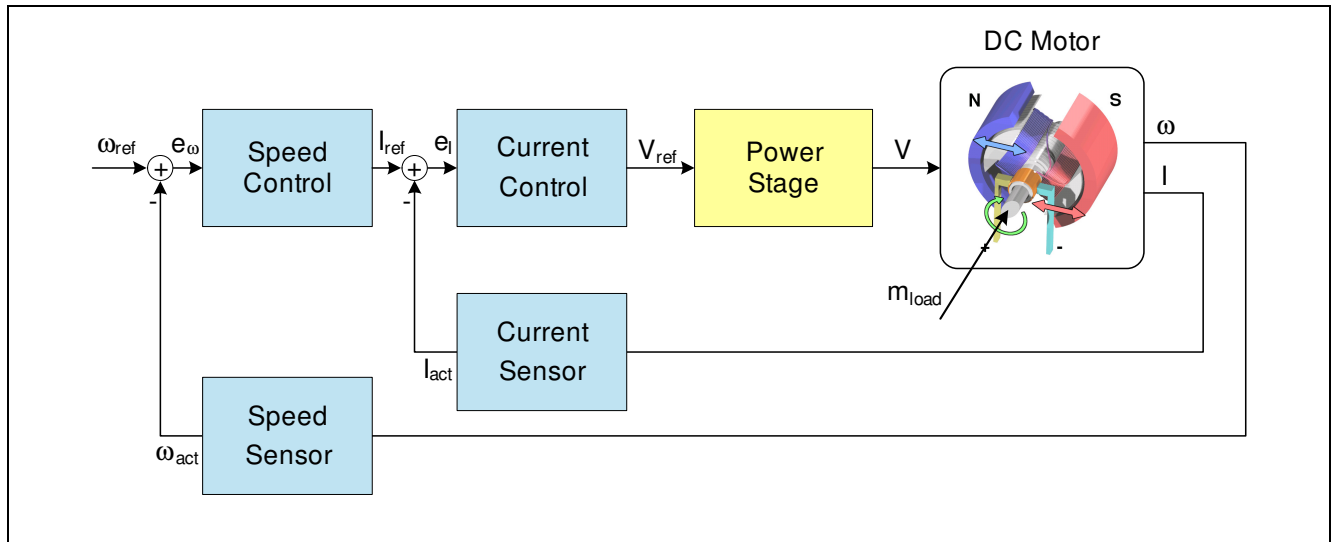
**Figure 4 Closed Loop Torque Control**



### 2.1.3 Closed Loop Speed and Torque Control

For a speed controlled DC motor, a cascaded control structure can be used. A cascade control structure is used because the speed change requirement is much slower than the the current requirement.

The speed control requires a speed sensor, such as a tachometer, and the current control requires a current sensor. The output of the speed control is the reference current for the current control. Usually a PI controller is used for speed and current control.



**Figure 5 Cascaded Speed and Current Control**

### 2.1.4 List of Equations

The mechanical and electrical equations of the DC motor are also valid for the PMSM.

*Note: These equations are referred to again later in this application note.*

**Voltage Equation:**

$$L \cdot \frac{di}{dt} + R \cdot i = v - v_{\text{bemf}} \quad (5)$$

**Induced Voltage:**

$$v_{\text{bemf}} = K_c \cdot \Psi \cdot \omega \quad (6)$$

**Mechanical Friction:**

$$m_{\text{Friction}} = r_{\text{Friction}} \cdot \omega \quad (7)$$

**Mechanical Equation:**

$$J \cdot \frac{d\omega}{dt} = m_m - m_{\text{Load}} - m_{\text{Friction}} \quad (8)$$

Angle:

$$\varphi = \int \omega dt \quad (9)$$

Speed in revolutions per minute (rpm):

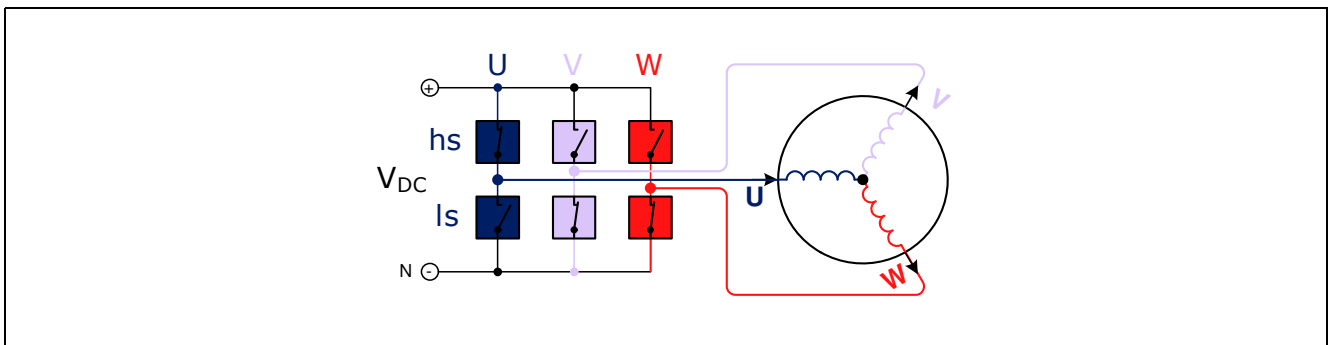
$$n = 60 \cdot \frac{\omega}{2\pi} \quad (10)$$

## 2.2 Space Vector Modulation

In this section, the modulation of a three leg voltage source inverter is described.

A three leg voltage source inverter contains six MOSFETs or IGBTs which act as switches. The switches connected to the positive supply rail are called high side switches (hs) and the switches connected to the negative rail of the power supply are called low side switches (ls).

In the following figure, a block diagram of a three leg voltage source inverter is shown:



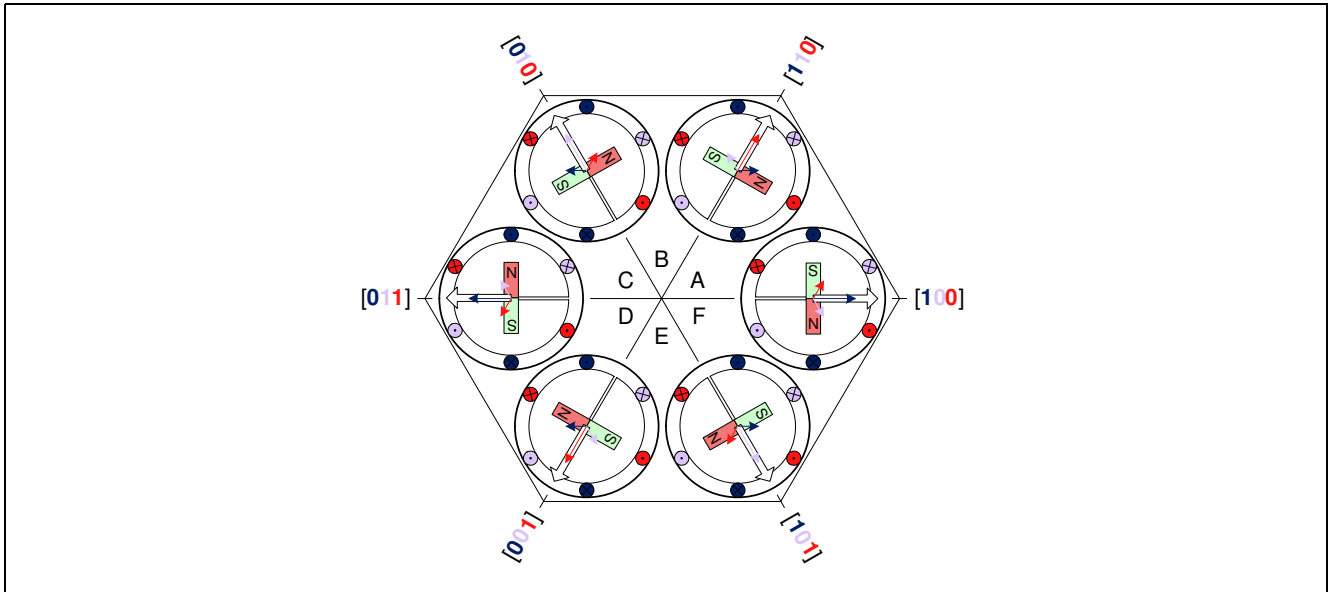
**Figure 6 Three Leg Voltage Source Inverter**

By switching the high side and low side switches on and off, there are eight possible states. These states do not cause cross currents inside the leg itself, but allow a current flowing to and from the motor.

State	$U_{hs}$	$U_{ls}$	$V_{hs}$	$V_{ls}$	$W_{hs}$	$W_{ls}$
000 (passive)	OFF	ON	OFF	ON	OFF	ON
100 (active)	ON	OFF	OFF	ON	OFF	ON
110 (active)	ON	OFF	ON	OFF	OFF	ON
010 (active)	OFF	ON	ON	OFF	OFF	ON
011 (active)	OFF	ON	ON	OFF	ON	OFF
001 (active)	OFF	ON	OFF	ON	ON	OFF
101 (active)	ON	OFF	OFF	ON	ON	OFF
111 (passive)	ON	OFF	ON	OFF	ON	OFF

The states can be seen as vectors in the space vector diagram of the coordinates U, V and W. There are six active and two passive vectors.

The space vector diagram of a U-V-W system contains six sectors (A, B, C, D, E, F) (see [Figure 7](#)).

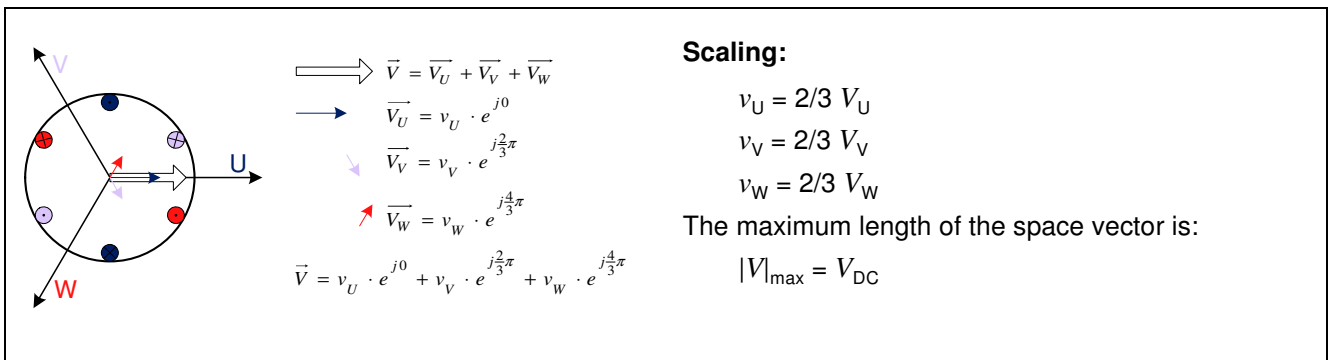


**Figure 7** Space Vector Diagram of a U-V-W System

The magnetic field (flux) of a three phase PMSM can be represented in a U-V-W space vector diagram.

In [Figure 7](#), the flux is simplified by a magnetic dipole. Whereas the stator coils are fixed in their position, the flux rotates dependent on the sector of the hexagon.

The voltages at the three coils correspond to the values of U, V and W. The resulting vector corresponds to the flux of the stator coils.



**Figure 8** Space Vectors in Complex Numbers

### 2.2.1 Reference Vector

The reference vector  $V_{ref}$  represents the resulting magnetic stator field (flux).

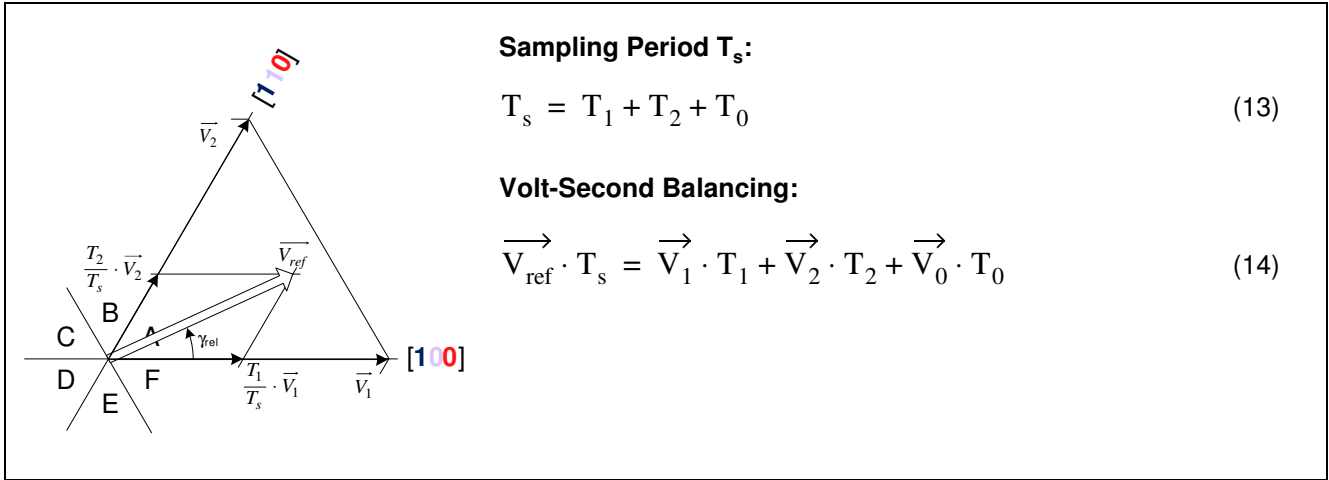
It is defined by the equation:

$$\vec{V}_{ref} = V_{ref} \cdot e^{j\phi} \quad (11)$$

It is rotating in space with the speed  $\omega$ :

$$\omega = 2\pi \cdot f \quad (12)$$

The reference vector can be approximated by two active vectors (for example V1 and V2) and one zero vector (V0). The plane is dissected into six sectors and the angle  $\varphi$  is transformed into the relative angle  $\gamma_{rel}$ .



**Figure 9 Reference Vector Approximation**

Taking into account the scaling of the reference vector and expressing the complex numbers in polar coordinates, the following equations are valid for the PWM on-time of a space vector modulation:

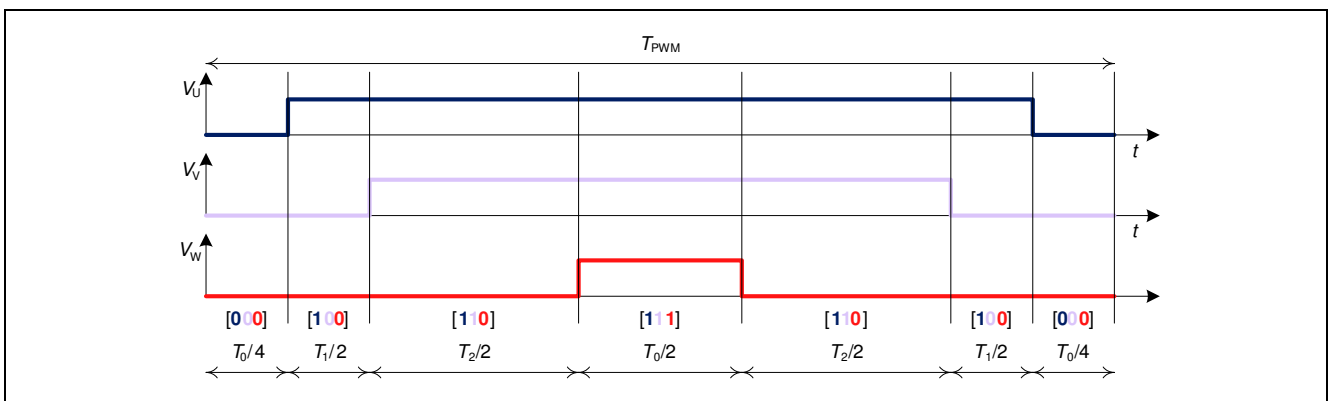
$$T_1 = \frac{\sqrt{3} \cdot T_s \cdot V_{ref}}{V_{DC}} \sin\left(\frac{\pi}{3} - \gamma_{rel}\right) \quad (15)$$

$$T_2 = \frac{\sqrt{3} \cdot T_s \cdot V_{ref}}{V_{DC}} \sin(\gamma_{rel}) \quad (16)$$

$$T_0 = T_s - T_1 - T_2 \quad (17)$$

The design of the switching sequence depends on the application requirements, but in order to reduce switching losses to a minimum the total number of a switching sequence is also usually minimized.

**Figure 10** shows a seven segment switching sequence:



**Figure 10 PWM Pattern of a Seven Segment Switching Sequence**

The pattern is symmetric at  $T_{PWM}$  and  $T_{PWM}/2$ . As a result, the PWM generation should be based on an up/down counter providing a center aligned pattern.

The CAPCOM6 unit provides a center aligned mode for timer T12 and the period value is adjusted with the special function register T12PR.

For the calculation of the compare values for the three phase signals, the following scaling is used:

$$T_{s,1,2} = \hat{T}_{s,1,2} \cdot 2 \cdot T12_{\text{clock}} = T12PR \cdot 2 \cdot T12_{\text{clock}} \quad (18)$$

The compare values are calculated as follows:

$$CMP_0 = \frac{\hat{T}_0}{2} = \frac{\hat{T}_s}{2} - \frac{\hat{T}_1}{2} - \frac{\hat{T}_2}{2} = \frac{T12PR}{2} - \frac{\hat{T}_1 + \hat{T}_2}{2} \quad (19)$$

$$CMP_{1(A,C,E)} = \frac{\hat{T}_0}{2} + \hat{T}_1 = CMP_0 + \hat{T}_1 \quad (20)$$

$$CMP_{1(B,D,F)} = \frac{\hat{T}_0}{2} + \hat{T}_2 = CMP_0 + \hat{T}_2 \quad (21)$$

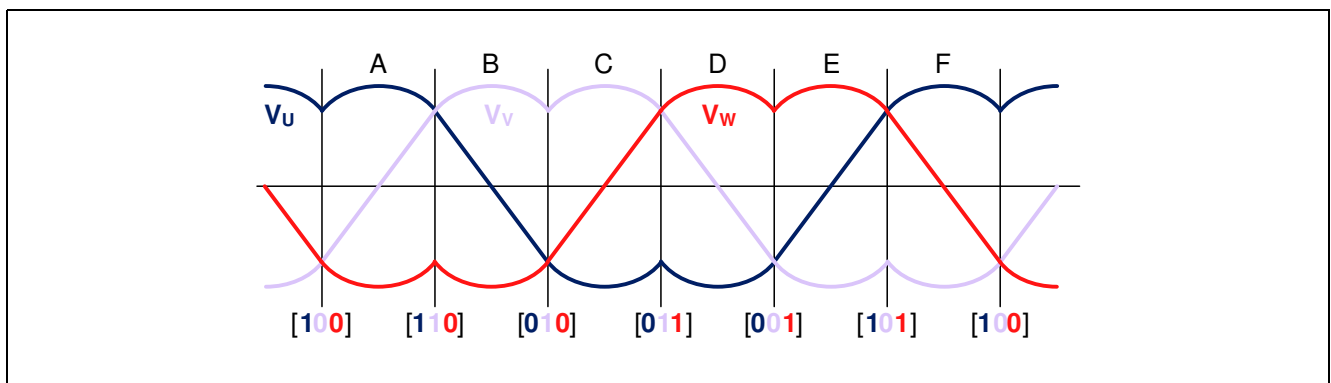
$$CMP_2 = \frac{\hat{T}_0}{2} + \hat{T}_1 + \hat{T}_2 = \frac{\hat{T}_s}{2} - \frac{\hat{T}_1}{2} - \frac{\hat{T}_2}{2} + \hat{T}_1 + \hat{T}_2 = \frac{T12PR}{2} + \frac{\hat{T}_1 + \hat{T}_2}{2} \quad (22)$$

The sector of the hexagon defines which compare value  $CMP_0$ ,  $CMP_1$  and  $CMP_2$  is used for phase U, V and W.

This is detailed in the following table:

Sector	A	B	C	D	E	F
phase U	$CMP_0$	$CMP_{1(B,D,F)}$	$CMP_2$	$CMP_2$	$CMP_{1(A,C,E)}$	$CMP_0$
phase V	$CMP_{1(A,C,E)}$	$CMP_0$	$CMP_0$	$CMP_{1(B,D,F)}$	$CMP_2$	$CMP_2$
phase W	$CMP_2$	$CMP_2$	$CMP_{1(A,C,E)}$	$CMP_0$	$CMP_0$	$CMP_{1(B,D,F)}$

The output voltage of each leg to neutral is shown in **Figure 11**:



**Figure 11 Output Voltage of Space Vector Modulation**

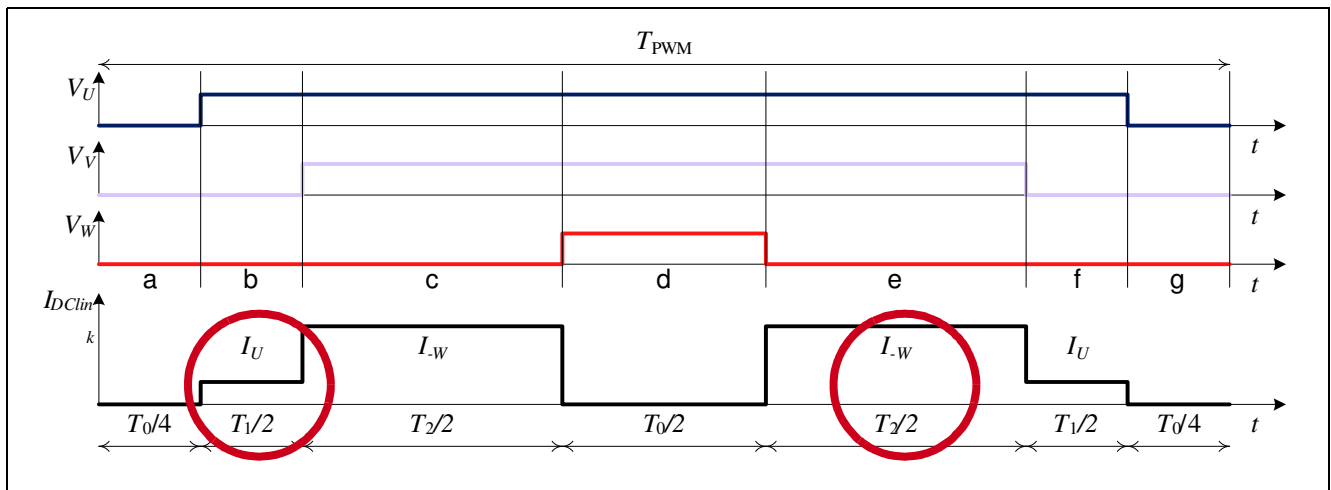
The voltages  $V_{UV}$ ,  $V_{UW}$  and  $V_{VW}$  at the motor are sinusoidal.

### 2.3 Phase Current Measurement

For many motor control schemes, the phase currents are required as input values. A cost efficient method requires only one shunt in the DC<sub>link</sub>.

Two phase currents can be reconstructed from the DC<sub>link</sub> current ( $I_{DClink}$ ) during one PWM period ( $T_{PWM}$ ).

The third phase current can be calculated by  $I_U + I_V + I_W = 0$ , but is redundant for the control.



**Figure 12 Phase Current Measurement within the PWM Pattern**

In order to realize this method, the ADC trigger points must be adjusted according to the PWM pattern.

Two different currents can be measured at PWM time,  $T_1$  and  $T_2$ . Depending on the actual sector, the measured currents have a different meaning.

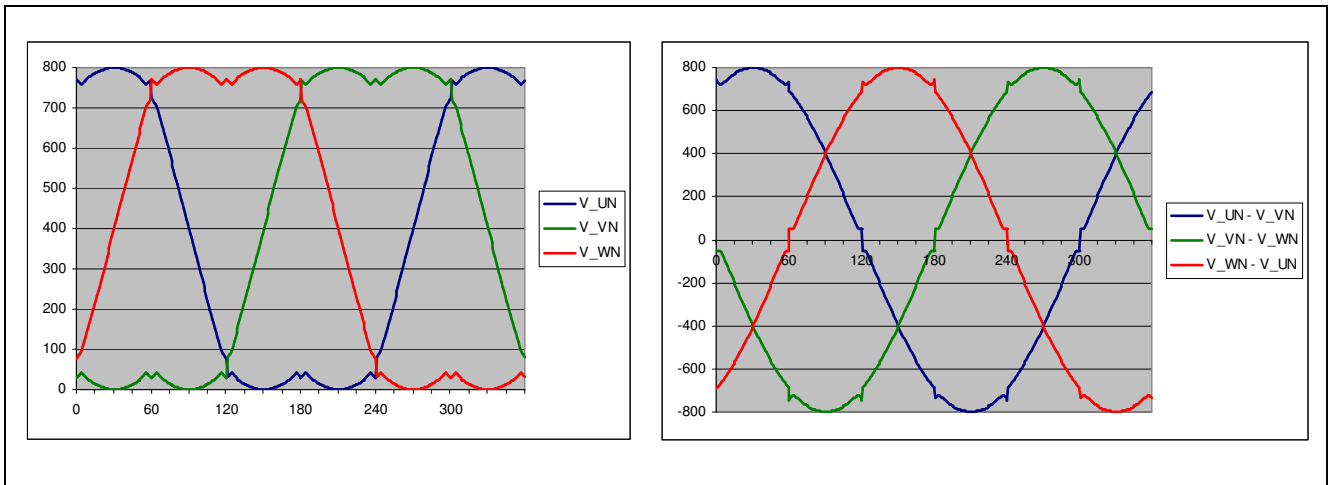
The following table shows these combinations and the calculation for phase current  $I_A$  and  $I_B$ :

Sector	A	B	C	D	E	F
$T_1$	U	V	V	W	W	U
$T_2$	-W	-W	-U	-U	-V	-V
$I_A = I_U$	U	$(-W) - V$	$-(-U)$	$-(-U)$	$(-V) - W$	U
$I_B = I_V$	$(-W) - U$	V	V	$(-U) - W$	$-(-V)$	$-(-V)$

#### 2.3.1 Limitations of Phase Current Reconstruction

When  $T_1$  or  $T_2$  equals 0, only one phase current can be reconstructed. To avoid this situation the PWM times  $T_1$  and  $T_2$  have to be limited to a minimum value. This causes a ripple in the phase-phase voltage and also in the phase current as well. A very fast ADC is required in order to maximize the performance of phase current reconstruction.

The following figure (Figure 13) shows the output voltage and phase voltage of a  $T_1$ - $T_2$ -limited space vector modulation.



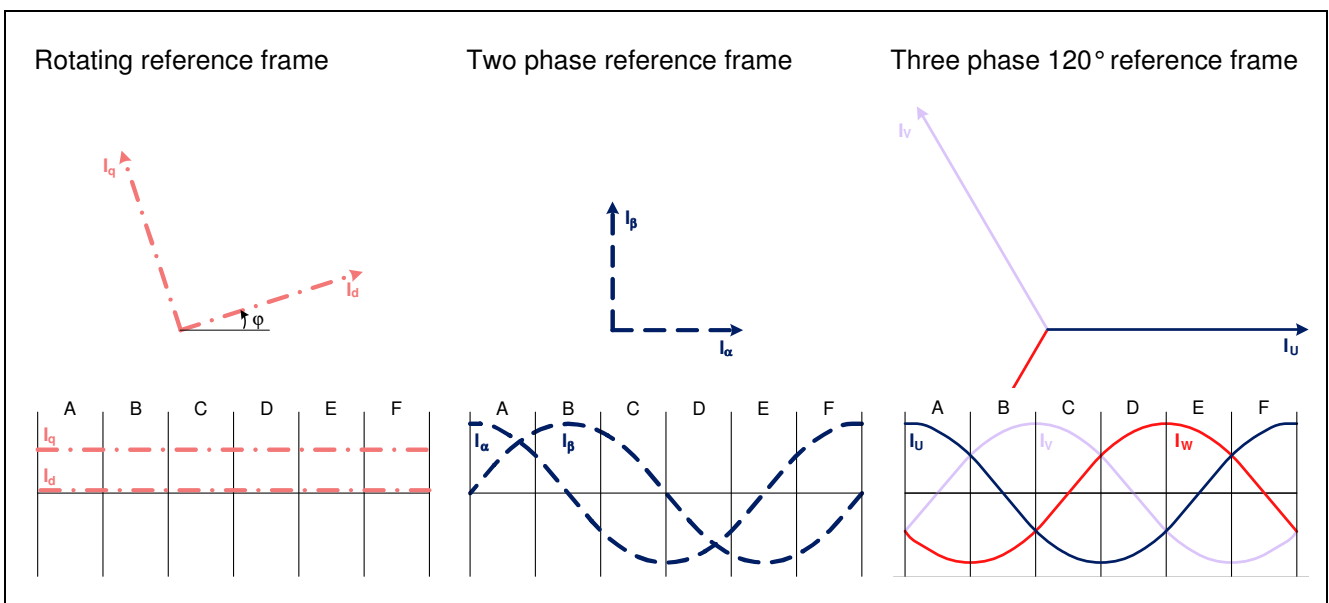
**Figure 13 Distortion in Output and Motor Voltage**

Although the output signal is slightly distorted, the most cost efficient method of phase current measuring is the reconstruction from the  $DC_{link}$  current via a single shunt. A very fast ADC is required to optimize the system performance. A direct trigger from the PWM Unit to the ADC reduces the CPU load significantly.

## 2.4 FOC Calculations

FOC is a method to generate a three phase sinusoidal signal which can be easily controlled in frequency and amplitude in order to minimize the current and to maximize the efficiency. The essential idea is the transform of three phase signals into two rotor-fix signals, and vice-versa. In the rotor-fix reference frame, the currents are stationary and easy to control. The inverse vector rotation causes the controlled voltages to rotate.

### 2.4.1 Stationary and Rotating Reference Frames



**Figure 14 Stationary and Rotating Reference Frames**

The transform from the three phase system to the two phase system is called a 'Clarke transform'.

The transform from two phase system to the rotating system is called a 'Park transform'.

The Clarke, Park and Inverse Park transforms are expressed in the following equations.



**Clarke Transform:**

$$i_{\alpha} = i_U \tag{23}$$

$$i_{\beta} = \frac{1}{\sqrt{3}} \cdot (i_U + 2i_V) \tag{24}$$

$$i_U + i_V + i_W = 0 \tag{25}$$

**Park Transform:**

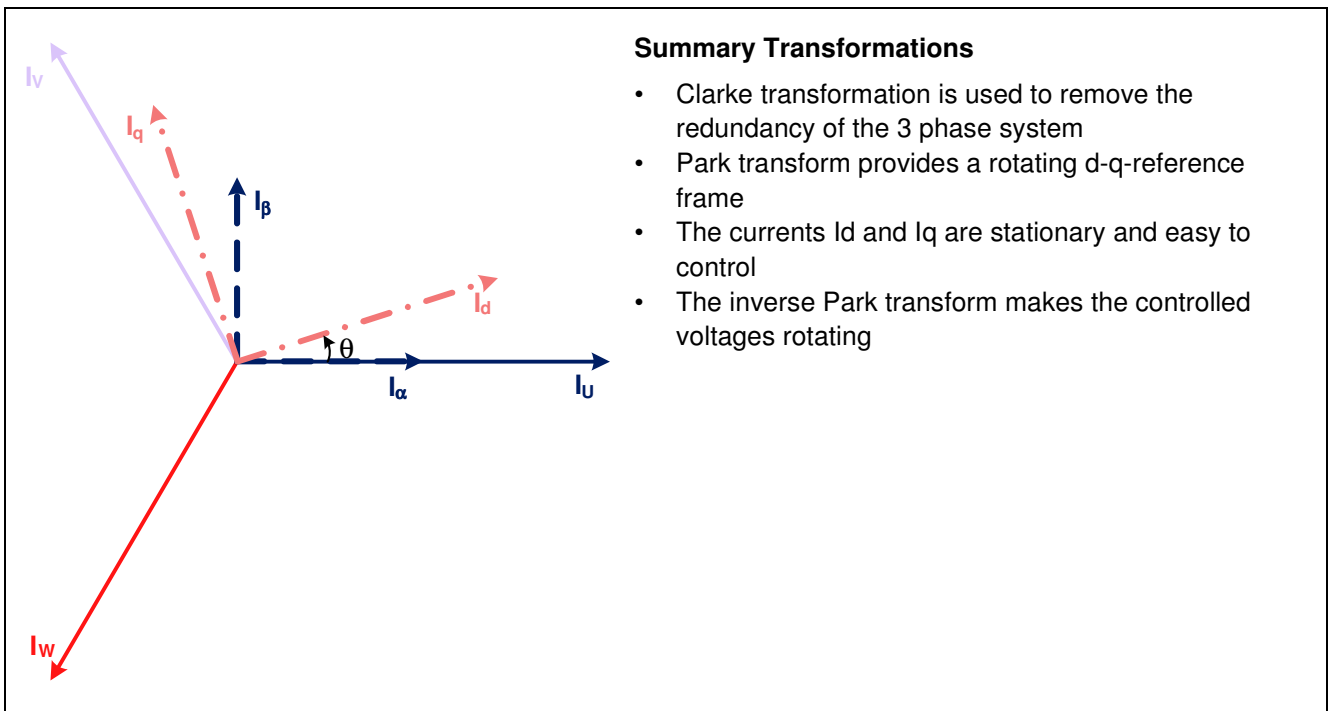
$$i_d = i_{\alpha} \cos \varphi + i_{\beta} \sin \varphi \tag{26}$$

$$i_q = -i_{\alpha} \sin \varphi + i_{\beta} \cos \varphi \tag{27}$$

**Inverse Park Transform:**

$$i_{\alpha} = i_d \cos \varphi - i_q \sin \varphi \tag{28}$$

$$i_{\beta} = i_d \sin \varphi + i_q \cos \varphi \tag{29}$$



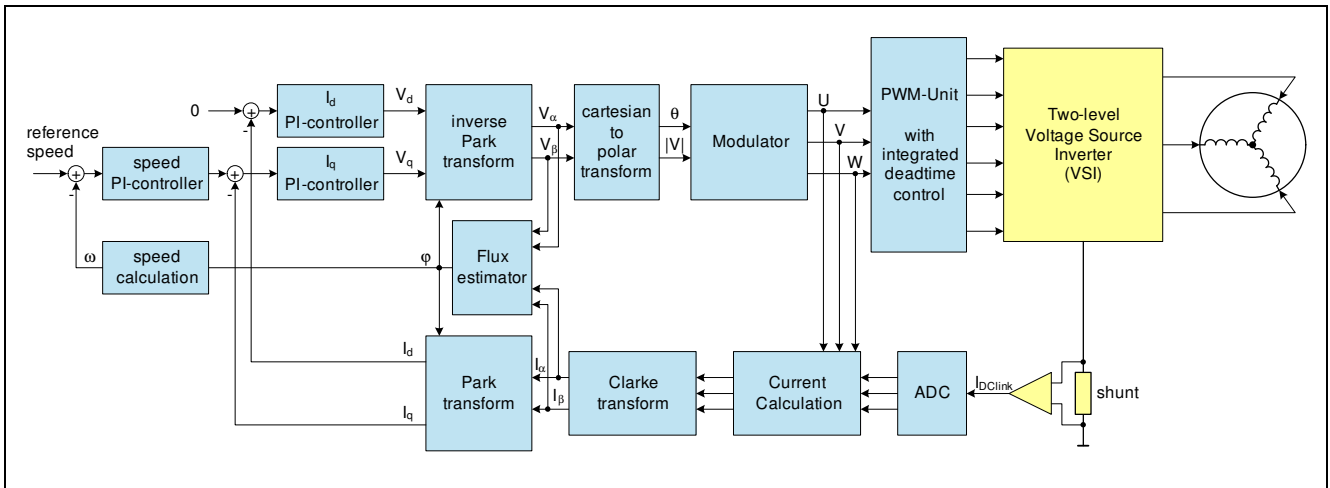
**Figure 15 Summary of Transformations**

### 2.4.2 Sensorless FOC

When the exact position of the rotor is not required, the costs for an encoder or other position sensor can be saved. The motor's back EMF is utilized in order to calculate the rotation angle and rotor position.

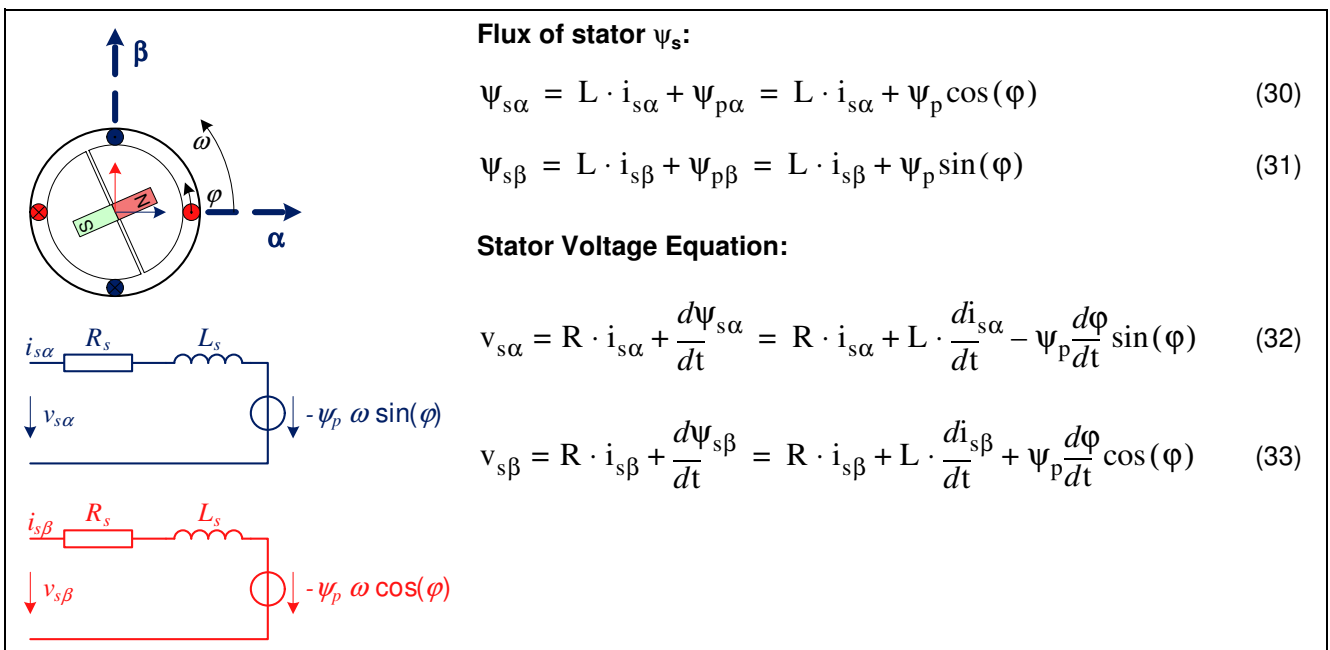
The back EMF is calculated in the flux estimator, which is based on the voltage model of the system in the two phase reference frame. A single shunt is enough to reconstruct the phase currents.

**Figure 16** shows the block diagram of the sensorless FOC algorithm. The flux estimator's input signals are taken from the orthogonal two phase stator system with the index  $\alpha$  and  $\beta$ . The output signal represents the rotor angle.



**Figure 16** Block Diagram of Sensorless Field Oriented Control (FOC)

The motor signals in the two phase stator system ( $\alpha$ - $\beta$ -system) are equivalent to the three phase system. As a result, an ideal two phase motor can be assumed in the  $\alpha$ - $\beta$ -system. Then there are only two equations to calculate. **Figure 17** shows the voltage equations of the motor in the  $\alpha$ - $\beta$ -system:



**Figure 17** Voltage Equations of Ideal 2-phase Motor

The flux of the stator (**Equation (30)**, **Equation (31)**) contain the mutual inductance  $L \cdot \vec{i}_s$  and the flux of the rotor with the permanent magnet  $\vec{\psi}_p$ .

The stator voltage equations ([Equation \(32\)](#), [Equation \(33\)](#)) contains the resistance of the coil  $R \cdot \vec{i}_s$  and the derivative of the flux of the stator (Faraday's Law).

Integrating the voltage equations, the stator flux is calculated as:

$$\Psi_{s\alpha} = \int (v_{s\alpha} - R \cdot i_{s\alpha}) dt \quad (34)$$

$$\Psi_{s\beta} = \int (v_{s\beta} - R \cdot i_{s\beta}) dt \quad (35)$$

The flux of the rotor and the orientation (angle  $\varphi$ ) of the permanent magnet of the rotor is calculated by insertion of [Equation \(34\)](#) in [Equation \(30\)](#), and [Equation \(35\)](#) in [Equation \(31\)](#).

$$\Psi_{p\alpha} = \Psi_{s\alpha} - L \cdot i_{s\alpha} = \int (v_{s\alpha} - R \cdot i_{s\alpha}) dt - L \cdot i_{s\alpha} \quad (36)$$

$$\Psi_{p\beta} = \Psi_{s\beta} - L \cdot i_{s\beta} = \int (v_{s\beta} - R \cdot i_{s\beta}) dt - L \cdot i_{s\beta} \quad (37)$$

$$\varphi = \text{atan}\left(\frac{\Psi_{p\beta}}{\Psi_{p\alpha}}\right) \quad (38)$$

Finally, the position of the rotor can be calculated by knowing the resistance  $R$  and inductance  $L$  of the motor. The stator voltage  $v_s$  is known by the algorithm. The current  $i_s$  needs to be measured in real-time.

### 3 FOC Implementation on XC878

Infineon's FOC Drive Application Kit (ordering code: KIT\_AK\_FOCDRIVE\_V1) comes with the following hardware:

- DriveCard XC878
- DriveCard XE164
- Low Voltage Inverter (23V-55V, 7.5A)
- PMSM motor (15W)
- DriveMonitor stick
- Power supply (24V)



**Figure 18 FOC Drive Application Kit**

This section describes the FOC implementation on the 8-bit device XC878.

The project setup is available for two 8051 toolchains:

1. Keil Compiler,  $\mu$ Vision IDE with Debugger and Simulator.
  - A Keil PK51 license is required as the code size has ~5kBytes in total.
  - An internet connection is needed for the license key. Please refer to Infineon Free Toolchain program: [www.infineon.com/Freetools](http://www.infineon.com/Freetools)
2. SDCC Compiler, MiniIDE, Hitop Debugger.
  - This toolchain is free of charge and the user can start immediately.

The project is DAVE compliant; i.e. the peripheral settings can be changed or added to using the DAVE code generator, and can then be recompiled afterwards.

A Microsoft Office Excel<sup>®</sup> worksheet is used as a user friendly interface for entering data on scaling and motor parameter adaptation. The Excel sheet is then used to generate the contents for a C header file.

The DriveMonitor is used for real-time monitoring of some interesting application variables, and also offers a control interface to the application itself. Data is exchanged via a CAN bus.

*Note: The description which follows is intended for the advanced user; i.e. some basic knowledge of the Infineon toolchain is a prerequisite. There are several application notes which explain this in detail (see [Chapter 6](#)), and hands-on-training is also available. For the less advanced user it is recommended to start with the DaveDrive autocode generator.*

### 3.1 Program Flow

Timer T21 operates as system timer and overflows every 500us. With this system tick the scheduler state machine switches to the application states and communicates to the host via CAN messages.

The FOC algorithm is calculated with every T13 interrupt at highest interrupt priority.

The emergency state is triggered by the CTRAP interrupt.

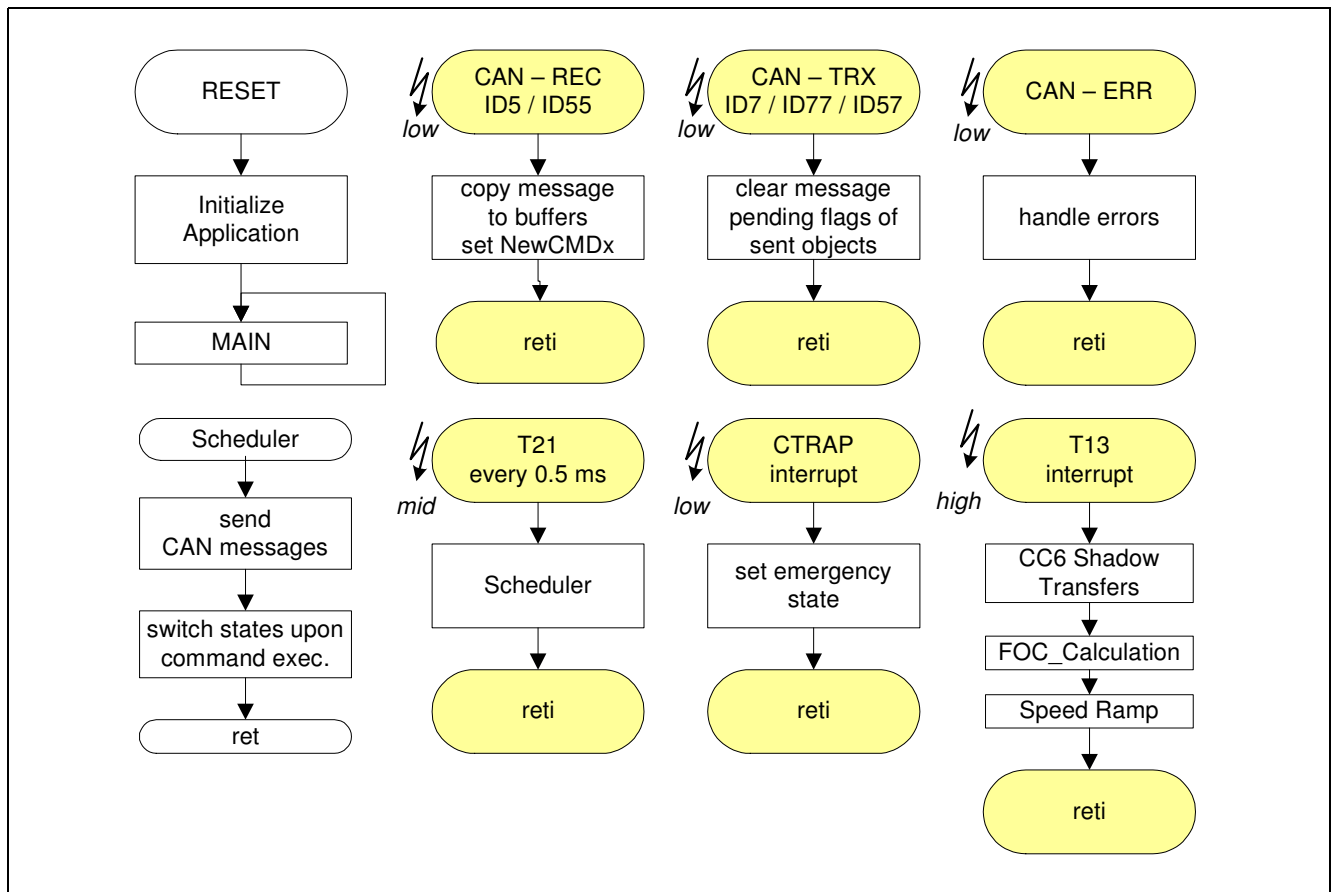


Figure 19 Overview of Main Functions and Interrupts for the Sensorless FOC Application

## 3.2 Initialization

The peripherals of the XC878 are initialized in separate files, before the application itself is initialized in the file main.c.

### 3.2.1 CC6 Timer Unit

The PWM is generated with the CC6 timer unit. Both timers T12 and T13 are used.

- T12 generates the signals for the Space Vector Modulation (SVM).
- T13 generates the two trigger events (compare and period match) for the ADC.

T13 runs in single shot mode and starts automatically with a zero-match of the T12.

The compare values for T12 and T13 and the period value for T13 is calculated within the function FOC\_calculation().

The period-match T13PM generates an interrupt. This is described in detail in [Chapter 3.5](#).

The passive level selection of the CC6 compare output signals fits to the Infineon 3 phase gate driver 6ED003L06-F. To avoid spikes and dangerous transitions at startup the CC6 has a special feature which is used for the initialization and for the Start/Stop/Emergency states. The CC6 trap state is forced by software. At the very beginning of CC6\_vlnit() the flag TRPF is set. This ensures that the output signals always show the defined passive level during initialization, until the flag TRPF is cleared.

*Note: DAVe does not set the TRPF flag.*

There are two application specific constants which have to be defined before compilation in the Excel sheet (foc\_defines.h):

- CCU6\_DEADTIME
- T12PERIODE

### 3.2.2 ADC

The ADC measures the DC-link current signal, the DC-link voltage and the potentiometer voltage. Two alternate analog signals can additionally be measured with this setup.

The ADC is triggered by the two T13 events, T13CM and T13PM.

The compare match (T13CM) triggers the sequential request source while the period match (T13PM) triggers the parallel request source.

The sequential source has a queue stage of three inputs: channels ch3, ch7 and one of either ch0, ch1, or ch6 in time-multiplex. The parallel source samples ch4.

Signal	Channel	Result	Source	Remark
Phase current $I_U$	ch3	RESR0	sequential	time critical
Phase current $I_V$	ch4	RESR1	parallel	time critical
Potentiometer	ch0	RESR3	sequential	time multiplex
DC link voltage VDC	ch1			
alternate signal	ch6			
alternate signal	ch7	RESR2	sequential	-

The function FOC\_calculation() reads the result registers and handles the software switched time-multiplexing of the three channels.

The ADC timing (sample and conversion time) influences the minimum duty cycles of the SVM. This is reflected in the Excel sheet (FOC\_defines.h) value  $T_{MIN}$ .

### 3.2.3 CAN

The MultiCAN module is used with one node, 5 message objects, a receive, a transmit and an error interrupt.

The CAN bus offers a fast and safe data exchange with a host computer.

Via the CAN module the FOC values can be monitored in real-time using the DriveMonitor, although the CAN module is not needed for the FOC algorithm itself. The scheduler routine handles the CAN interrupts and messages.

The big advantage of using CAN is very efficient message handling with very little CPU load, and without error handling. For additional information, please refer the DriveMonitor application note (See [Chapter 6](#)).

### 3.2.4 T21 Timer

Timer T21 is configured in 16-bit auto-reload mode, providing the system tick which triggers the scheduler interrupt. The timebase can be adjusted, but by default it is set to 500us.

### 3.2.5 Application

The application specific settings are performed in the following functions:

- Scheduler\_vInit()
  - Resets the global variables for the command handling.
- Scope\_vInit()
  - Initializes the CAN transmit buffers with the pointers for the monitoring data.
- MotorControl\_vInit()
  - Sets up the startup values for the open and closed loop controllers. Here the parameters entered in the Excel sheet (FOC\_defines.h) are read and the CTRAP pin is polled in order to detect a static fault. The state machine is then initialized to the `DEFAULT_STATE`. The default state is another parameter on the Excel sheet, and is either the Idle or the Bootstrap state.



### 3.3 Scheduler State Machine - T21 Interrupt

This state machine runs cyclic at every 500us. Here the application states are switched.

The entry into the state machine is the IDLE state after power-up. This is dependent on the Excel sheet parameter (FOC\_defines.h) and is either STATEBOOTST or DEFAULT\_STATE.

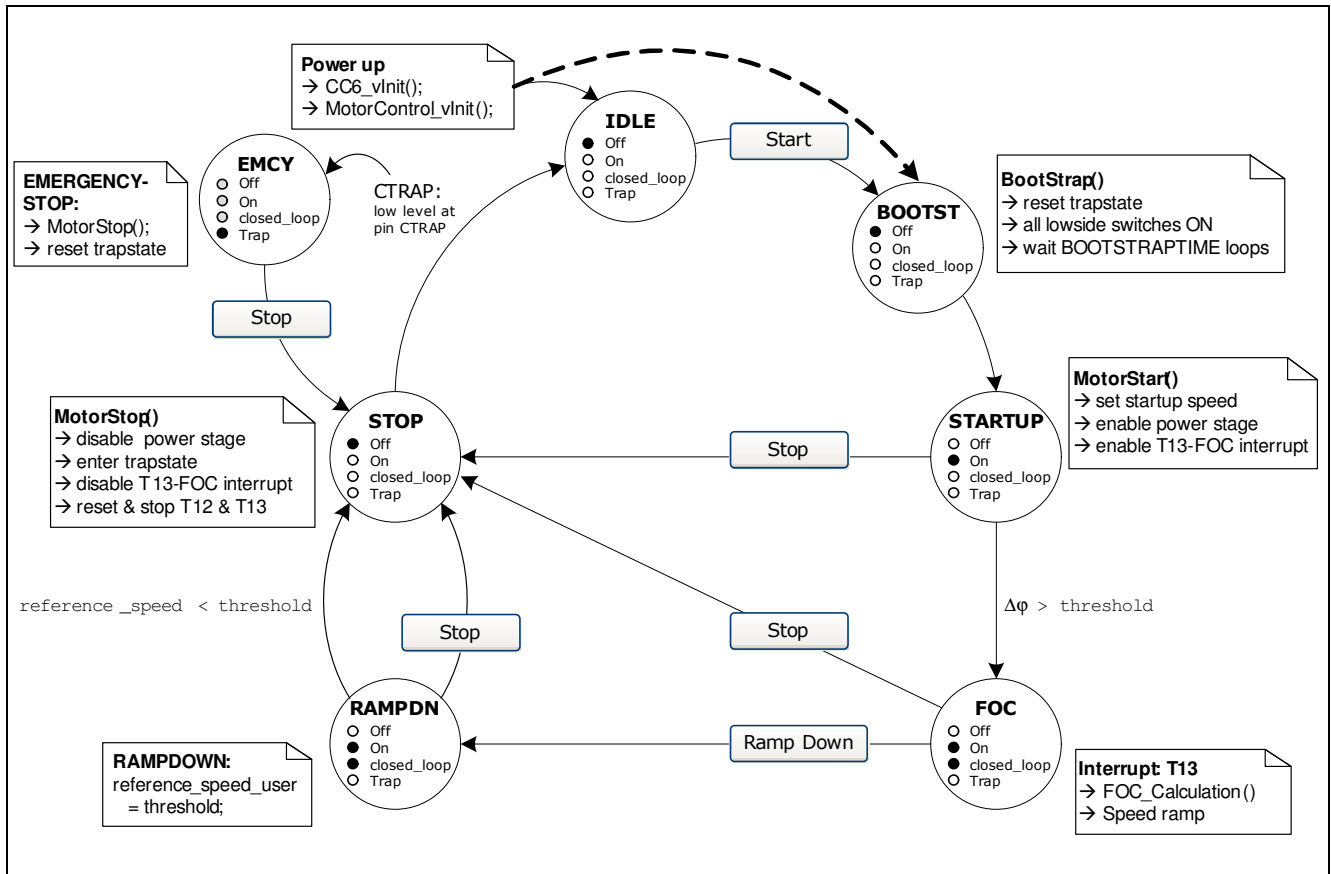


Figure 20 Application State Machine

#### 3.3.1 State IDLE

- Wait on Start-command from host.

#### 3.3.2 State BOOTST

This is the first state where the CC6 drive active states on it's output pins. The bootstrap capacitors are charged by switching all lowside switches ON and the highside switches OFF. This is performed with the following sequence:

- Enable CTRAP emergency feature (falling edge on pin CTRAP will cause a trap state) and CTRAP interrupt if set in Excel sheet (FOC\_defines.h): CTRAP\_Enable and CTRAP\_PIN (pinselection).
- Enable the POWER\_STAGE; i.e. switch the ENABLE\_PIN of the gate driver IC. The polarity and pin can be selected via the setting ENABLE\_PIN.
- Switch on all lowside switches by programming the CC6 shadow registers.
- Start timer T12 and generate PWM (constant 100% duty cycle).
- Release the trap state by resetting the TRPF.
- Stay in Bootstrap state until the global variable guc\_SpeedControlCounter has counted down. This variable is defined in the Excel sheet (FOC\_defines.h):
  - BOOTSTRAPTIME

### 3.3.3 State STARTUP

The startup mechanism is quite complex but has the following flow (see also [Figure 21](#) which follows):

- Switch on the PWM modulator by calling MotorStart().
  - Here the global variables are initialized, bit `gb_On` is set.
  - The startup speed and direction as well as the offset phase voltage are set in case of a “cold-start”. The global bit `gb_Once` is used to distinguish between a “cold-start” and a “warm-start”. This bit is set if it was zero before.
  - The timer T13 interrupt is enabled which allows the `FOC_calculation()` with the next system tick.
- There is a switch `CLOSEDLOOP`, which is another Excel sheet parameter (`FOC_defines.h`). If not defined, the algorithm will stay in an open loop V/f control and will never enter the FOC. The difference between these two is that the V/f control takes the rotor position ( $\Delta\phi$ ) from a fixed relationship between voltage and frequency regardless of the real position, whereas the FOC determines the angle in the flux estimator. The FOC therefore operates with the real rotor position.
- At startup, the flux estimator cannot calculate the correct angle as the input signals (current measurement) are not accurate enough. This means that the motor has to start in an open loop V/f control. As the V/f constant depends on the motor and load characteristic there are a few parameters which can be adjusted in the Excel sheet (`FOC_functions.h`)
  - Startup speed ([Figure 21](#), SpeedT0) `DELTA_ANGLE_START_T0`
  - Threshold speed ([Figure 21](#), Threshold) `DELTA_ANGLE_THRESHOLD`
 The `SpeedUser` (`DEFAULT_SPEED_REFERENCE` in `FOC_defines.h`) is the reference speed the user has to set. If this is below the threshold speed, the algorithm will stay in V/f control.
- The slew rate of the V/f is defined by Excel parameters `SPEEDRAMPSTEP` and `SPEEDRAMPTIME`. The steps are counted in T13 interrupt (See [Chapter 3.5.5](#)); i.e. the timebase for the speedramp is the T13 period.
- This Startup control block ramps with every system tick of the V/f ramp. If the threshold speed is reached, the global bit `gb_closed_loop` is set and the state variable is switched to the state FOC.
- The state Startup can be left by an emergency event or by command Stop.

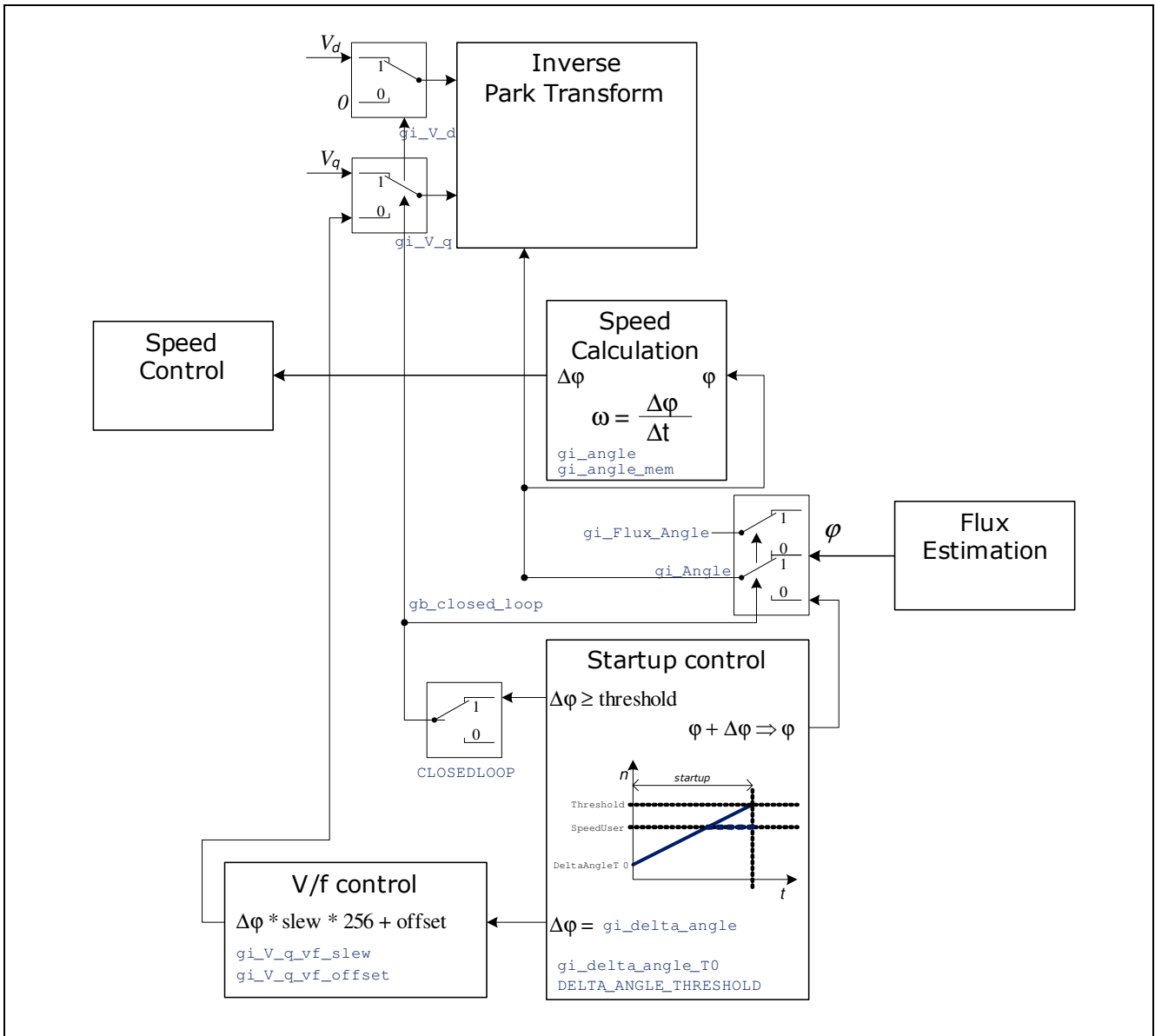


Figure 21 Startup Algorithm

### 3.3.4 State FOC

This state does nothing but run the sensorless FOC algorithm. The only input parameter is the speed reference (`gi_Speed_reference_user`). The algorithm tries to keep the speed constant even under load changes.

The speed reference information comes either from the DriveMonitor (via CAN) or from the onboard potentiometer. This can be configured in the Excel sheet (FOC\_functions.h) parameter `SPEED_POT1`.

In state FOC, global bit `gb_closed_loop` is one.

The state FOC can be left by an emergency event or by the commands Stop or Rampdown.

### 3.3.5 State RAMPDOWN

After receiving a Rampdown command, the state FOC is left and the speed is ramped down. This is done by simply setting the reference speed to the startup speed. The speed controller will reduce the  $I_q$  reference but limits it to a positive value; i.e. it does not actively brake. The motor will therefore slow down with its own inertia.

For active braking the  $I_q$  limitation can be changed in `FOC_Calculation()` at section "Speed Control". In this instance it is recommended to supervise the DC link voltage  $V_{DC}$  and limit the  $I_q$  amplitude.

On reaching the startup speed the state Stop is set for the next system tick.

### 3.3.6 State STOP

The state Stop is entered upon receipt of the command Stop. The function `MotorStop()` is called. Here the powerstage is switched off by the `ENABLE_PIN` and the interrupts T13 and CTRAP are disabled.

The trapstate is forced by software, and the timers T12 and T13 are stopped and reset.

The speed is reset to the default and the global bits `gb_On` and `gb_Off` are respectively reset and set.

The state variable is set to IDLE for the next state.

### 3.3.7 State EMERGENCY

In the case of a falling edge at the CTRAP pin due to a fault situation such as an over-current, the CTRAP interrupt is vectorized. This sets the state Emergency.

The `MotorStop()` function is called (see [Chapter 3.3.6](#)), the trapstate is released by clearing bit TRPF, and bit `gb_Trap` is set in order to signal a locked emergency situation. The Emergency state can only be left by the receipt of a Stop command or a RESET. A Start command has no effect.

If the Stop command is received, the next state is set to Stop. Then a Start command will restart the state machine.

In the case of a static low level at pin CTRAP, due for example to the none resolved fault situation, the Emergency state is entered in the bootstrap phase.

## 3.4 CAN Communication - CAN Interrupts

The CAN controller is configured to three interrupt vectors for receive, transmit and error handling.

The CAN Message Objects have the following IDs:

- ID5 - receive object - MO 0: SET/GET command and Buttons
- ID55 - receive object - MO 2: Unused
- ID7 - transmit object - MO1: Slow data for status flags and display field
- ID77 - transmit object - MO 3: Fast data for oscilloscope and progress bar
- ID57 - transmit object - MO 4: Respond to GET command

On receipt of a CAN message object the receive interrupt is vectorized. The corresponding message object is copied into the 8-byte wide receive buffer in case there is no command pending. The global command variable is updated and a bit indicating a new command is set.

At every system timer tick the scheduler state machine is executed and the bit indicating a new command is polled. If set, the corresponding command is executed and the bit is cleared afterwards in order to accept new commands.

This handshaking mechanism with the receive interrupt ensures that an incoming command has to be served first before a new command is accepted.

The three transmit message objects are configured for sending data from the target to the host. The transmit interrupt service routine is vectorized once the CAN message is completed. The corresponding transmit pending flag is cleared. Each transmit message object has a different trigger event.

The transmit messages have to be initialized. The transmit buffers `CANTrxBuf0[8]` and `CANTrxBuf1[8]` are written with the address of the data to be transmitted. This is performed in the function `Scope_vInit()`.

A detailed description of the command handling and the communication protocol can be found in the DriveMonitor application note (See [Chapter 6](#)).

### 3.5 FOC Control Loop - T13 Interrupt

The following figure (Figure 22) shows the timing diagram for the sensorless FOC algorithm.

This figure is a zoom into two PWM periods of the space vector modulation. It shows the resulting ADC trigger points for the single shunt current measurement. These trigger points are pre-calculated for the next cycle within the FOC routine. The speed ramp function is also called here. The hard real-time condition is that the FOC algorithm must fit into  $1.5 \cdot 2 \cdot T_{PWM}$  cycles.

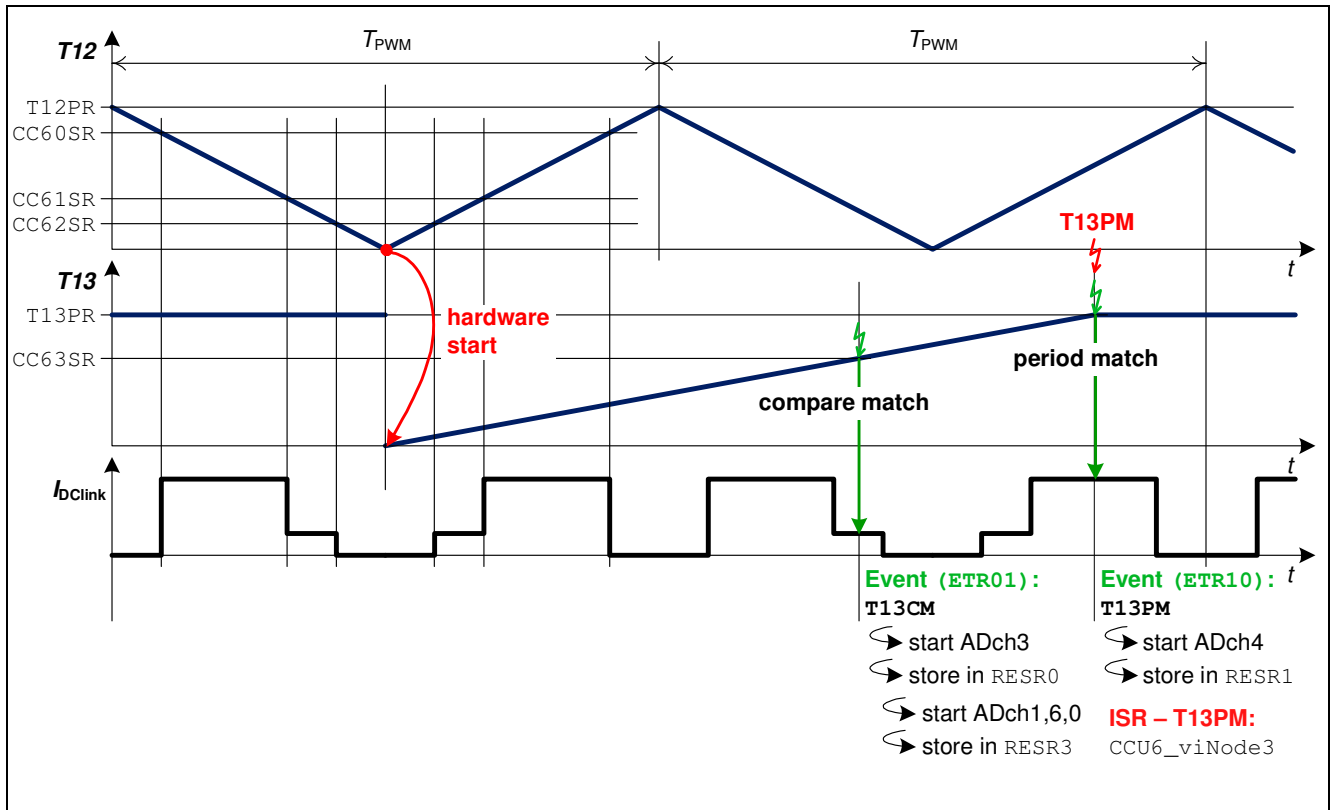


Figure 22 Timing of T12 PWM and T13 ADC Trigger Events for Current Measurement at the DC Link Shunt

#### 3.5.1 Shadow Transfer

First, the shadow transfer request bits for timer T12 and T13 are set. This ensures that the actual calculated compare values for the SVM and the current measurement are updated with the next period match event.

#### 3.5.2 FOC Calculation

After the shadow transfer, the FOC calculation routine is called.

The whole FOC algorithm is summarised in Figure 23 (see also Chapter 5 for a larger version of this diagram). The building blocks shown are exactly implemented, mostly in the file FOC\_Functions.c.

The FOC calculation has the following flow (starting from the bottom right corner of the diagram, moving in a clockwise direction):

- ADC result register control
- Current calculation and phase current extraction
- Clark transformation
- Flux estimation
- Park transformation

- Speed calculation and speed control
- Current control
- Inverse park transformation
- Cartesian to polar transformation
- Space vector modulation

The block startup control is described in [Chapter 3.3.3](#).

The FOC calculations are performed in fixed point number format Q15.

Multiplying two Q15 numbers will result in a Q30 number. The 16-bit Multiply and Divide Unit (MDU) as part of the vector computer is well suited for these calculations. The result holds a 32-bit value whose two most significant bytes represent the Q14 result. In order to get the Q15 result, this number has to be shifted left by one bit.

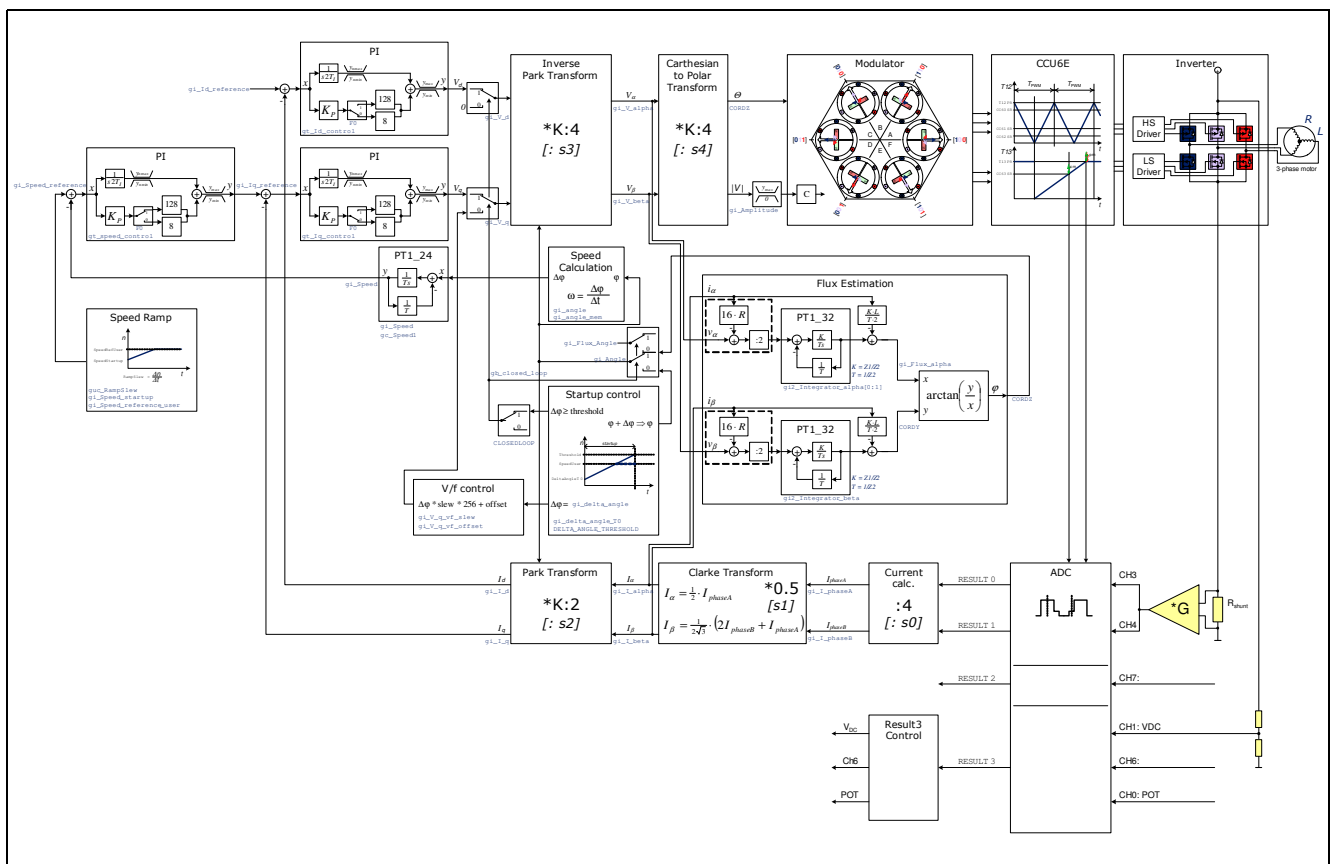


Figure 23 FOC Building Blocks

### Phase Current Calculation

The ADC is configured to measure phase currents via the DLink current within one PWM period. The result of the measurement is stored in result register 0 and result register 1. The current calculation block takes the values from these result registers and calculates the phase currents  $I_U$  and  $I_V$ , which are  $I_{phaseA}$  and  $I_{phaseB}$  in the diagram, respectively. They are signed integer values which are scaled by a quarter full scale. The scale factor is called  $s0$  in the Excel file.

### Clarke Transform

The Clarke Transform is calculated according to [Equation \(23\)](#), [Equation \(24\)](#) and [Equation \(25\)](#). The result is divided by 2, due to calculation time optimization with the MDU. As a result, the Clarke transform scales the current by 0.5, which is the scale factor  $s1$  in the Excel file.

### Park and Inverse Park Transform

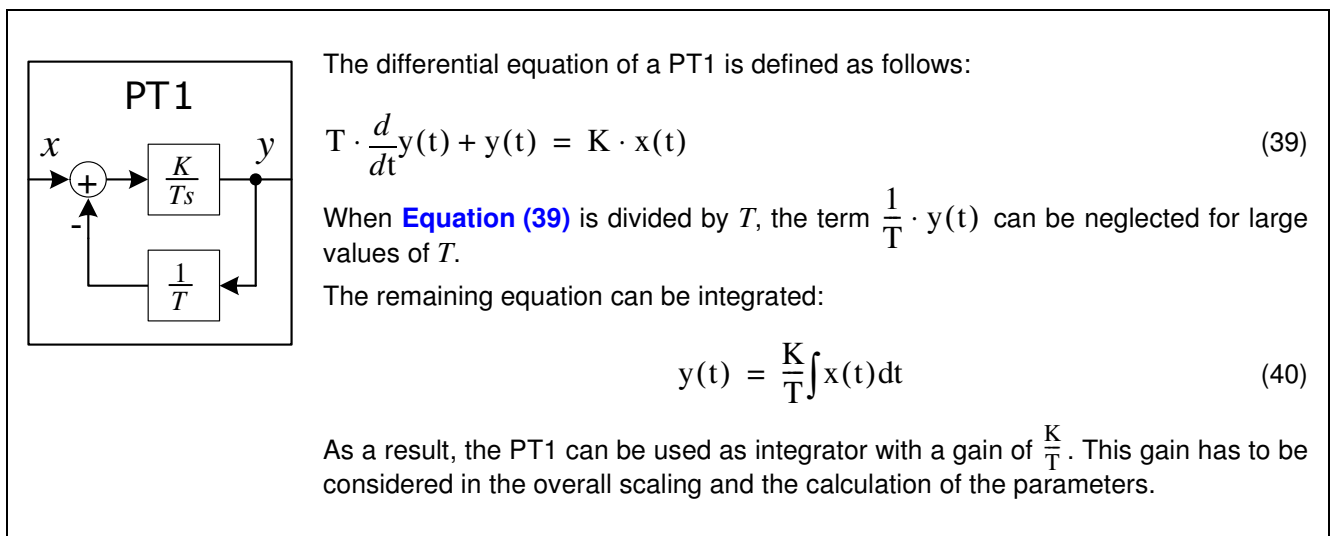
The Park and inverse Park transform is calculated by the CORDIC module of the vector computer. The CORDIC can be used in six different modes. Here the circular rotation mode is selected, which scales the result by a factor  $K \approx 1.64676$ . The additional scale down by dividing with 2 (Park transform,  $s2$ ) and 4 (inverse Park transform,  $s3$ ) is an optimization of the FOC algorithm.

### Carthesian to Polar Transform

The Carthesian to Polar transform is calculated with the CORDIC module in circular vectoring mode. The scale factor  $K \approx 1.64676$  results in the approximation scheme. Together with the additional division by 4 the scale value  $s4$  is defined.

### 3.5.3 Flux Estimator

The flux estimator has been discussed in [Section 2.4.2](#). The final equations ([Equation \(36\)](#) and [Equation \(37\)](#)) require an integrator which is implemented as PT1 controller. See [Figure 24](#) for details.



**Figure 24 PT1 Controller**

The implementation of the flux estimator makes use of the vector computer wherever possible, and the parallel use of CORDIC and MDU provides the highest performance.

The CORDIC is used for calculation of  $v_{s\alpha} - R \cdot i_{s\alpha}$  in linear rotation mode. In this mode, the Z data of CORDIC is handled as a number in signed 4Q11 format. Because the calculation of all other values is in signed Q15 format, the resistance  $R$  appears with a gain of 16 in [Figure 23](#).

As shown in [Figure 24](#), the PT1 controller provides a gain of  $\frac{K}{T}$ . Using this gain, the inductance  $L$  of the motor can be adjusted to a reasonable resolution in the algorithm.

The Excel file provides the adjustment of both  $K$  and  $T$ . It is recommended to adjust  $K$  in order to increase the resolution of the inductance  $L$ . Low inductive motors in particular will require an adjustment of this value. The integration window, represented by the  $T$  value, can be adjusted according to the application's needs. Small values of  $T$  will cause the flux estimator to act very fast. As a result noise in the current measurement does not accumulate too much due to the integration. Large values of  $T$  will result in a high accuracy of the integrator, but the dynamic of the system is reduced.

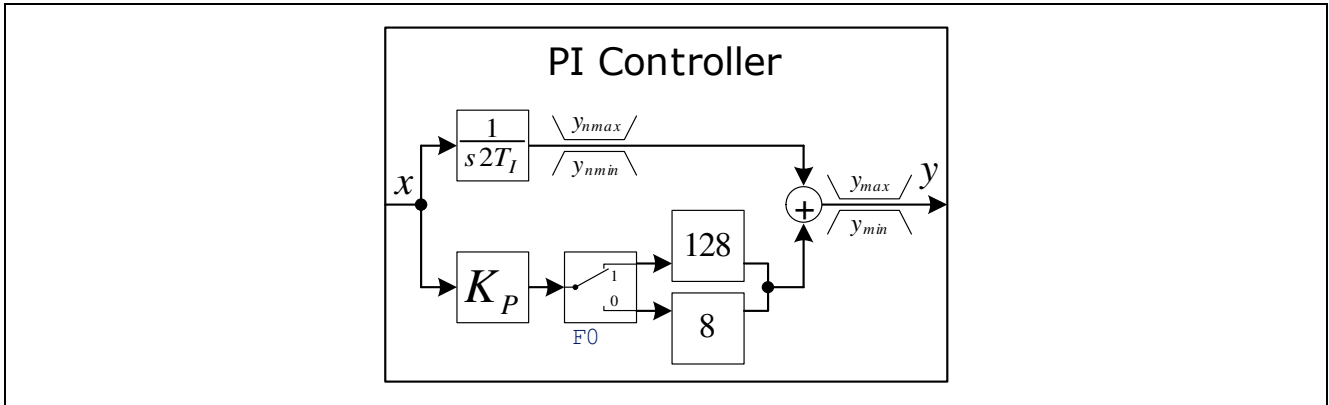


### 3.5.4 PI-Controller with Anti Wind-Up Limitation

The speed control and the current control of the implemented FOC utilize the same PI controller. This PI controller has an additional adjustable gain for the constant  $K_p$  and a separate limitation for the integral part. The additional gain is necessary to support a wide range of  $K_p$  values with the same controller.

The additional gain can be selected by the flag F0:

- F0 = 0: Gain = 8
- F0 = 1: Gain = 128



**Figure 25 PI Controller with Anti Wind-Up Limitation  $y_n$**

The limitation  $y_{max}$  and  $y_{min}$  of the integral part provides an anti wind-up limitation.

A PI controller can wind-up when the  $K_i$  value is small and the difference between the given value and actual value can not be compensated by the whole system. Then the integral part continues to increase, but there is no effect due to the output limitation of the PI controller. When the given or actual value changes and the PI controller should compensate in the opposite direction, the integral part has to reduce by the small  $K_i$  value. As a result, the output of the PI controller is misaligned. In order to reduce the duration of this misalignment, the integral part can be limited independently to the output of the PI controller.

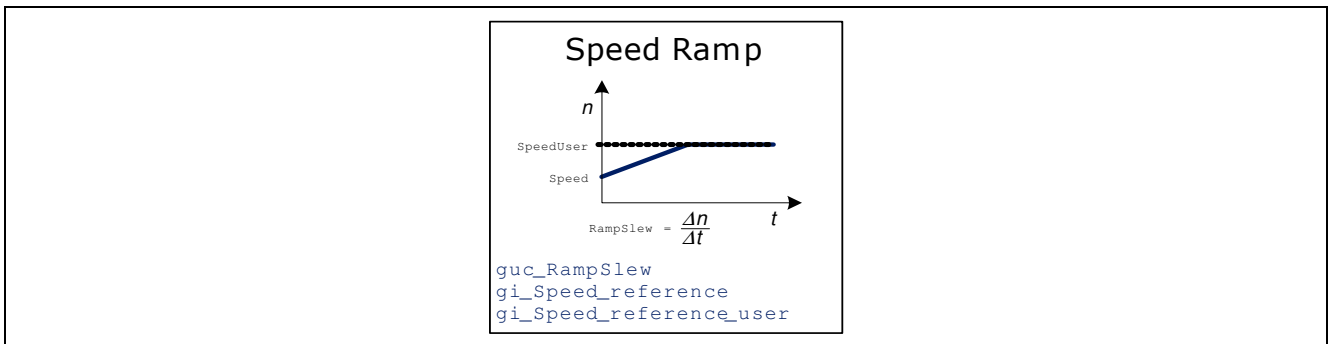
### 3.5.5 Speed Ramp

The speed ramp is dependent on the slew rate. The slew rate is defined by the Excel parameters `SPEEDRAMPSTEP` and `SPEEDRAMPTIME`.

The speed steps are counted in T13 interrupt, which is triggered by a T12 zero match event; i.e. the timebase for the speedramp is the T12 period.

The speed formula is:

- Speed slew rate =  $\Delta n / \Delta t$  with  $\Delta n = \text{SPEEDRAMPSTEP}$  and  $\Delta t = \text{SPEEDRAMPTIME} * \text{T12period}$



**Figure 26 Speed Ramp**

The speed ramp handles three cases:

- V/f open loop at startup
- FOC
- FOC in case of rampdown

In the latter two cases the global bit `gb_ramp` is set to indicate that reference has not yet been reached.

## 4 How to Use the Setup


### 4.1 Excel Sheet and FOC\_Defines.h

The parameterization of the application is performed via a header file (FOC\_Defines.h). For ease of use, an Excel sheet (see [Figure 27](#)) is used to generate the header file contents. The algorithm itself (FOC\_functions.c and scheduler.c) can be left untouched.

The Excel sheet has two pages: the input page *Input Parameter*, and the output page *FOC\_defines.h*.

The *Input Parameters* are set in the yellow shaded fields. These are the most relevant parameters and have the following structure:

- *Motor Parameter* - The resistance and inductance of the motor windings. The input value refers to one phase; i.e. phase-to-start and not phase-to-phase.
- *Startup Parameter* - Values which are used in the startup block.
- *Inverter Parameter* - When a customized inverter is used, it can be adapted here.
- *Current Measurement* - For adjustment of the current amplification or the shunt.
- *Speed and Current Controller* - Sets the integral and proportional factor for the PI-controllers.
- *Flux Integration Scaling* - Sets the PT1 filter for the flux estimator (integration window time and gain).
- *Speed Filter* - For adjustment of the PT1 filter for the speed filter.
- *Speed Ramp* - For adjustment of the speedramp time. If the Speed Ramp time in the startup parameters is not fast enough, the incremental steps can be increased.

 <b>XC878 sensorless FOC</b> <b>Parameter Calculation</b> <small>v1.0.00</small>	
<b>Motor Parameter:</b> <b>MAXON EC32-flat</b>	
R	6,85 ohm
L	3865 µH
Pole Pairs	4
<b>Startup Parameter:</b>	
Startup Speed T0	0 rpm
Startup Speed Threshold	800 rpm
Startup V/f offset ( $f = 0$ )	1 V
Startup V/f slew rate	0,307692308 V/Hz
Reference Speed User	2000 rpm
Speed Ramp	5000 rpm/s
<b>Inverter Parameter:</b>	
DC link voltage	24 V
dead time	1 µs
switch delay (max)	0,7 µs
PWM frequency	15000 Hz
Bootstrap precharge time	1,5 ms
ENABLE pin logic level	1
1: high active -1: low active	
ENABLE_PIN	P5 0
CTRAP Enable	1
CTRAP_PIN	P3 6
Autostart enable	1
Use poti for speed	1
<b>Current Measurement:</b>	
R_shunt	0,02 ohm
R_IN	1 kOhm
R_feedback (R14)	33 kOhm
maximum Voltage at ADC	5 V
<b>Speed PI Controller:</b>	
Ki	0,01 ( < 0,5 )
Kp	10 ( < 128 )
<b>Current PI Controller:</b>	
scale	0,1 ( <= 0,1 )
<b>Scale Factors for Drive Monitor</b>	
<b>Startup:</b>	
Startup V/f offset ( $f = 0$ )	0,000410 V
Startup V/f slew rate	0,000056 V/Hz
<b>Speed:</b>	
speed	1,716614 rpm
<b>Voltages:</b>	
V(d,q)	0,000410 V
V(α,β)	0,000997 V
V(svm amplitude)	0,018008 V
<b>Currents:</b>	
I(d,q)	0,001478 A
I(α,β)	0,001795 A
I(u,v,w)	0,000898 A
<b>PI Controller:</b>	
Speed kp	0,003906
Speed ki	0,000031
Current kp	0,000244
Current ki	0,000031

**Figure 27 Excel Sheet Input Page (The Most Relevant Values for FOC and DriveMonitor)**

It should be noted that the parameters influence each other. For example, a very low inductance value  $L$  will scale to a very low value  $L'$ , which is used for the flux estimator. In this instance the input parameter  $K$  in the *Flux Integrator Scaling* should be increased.

*Note: If the Excel sheet shows cells marked in red, then the scaling parameters have to be changed accordingly.*

After changing the Input Parameter the contents of the output page have to be copied to the file `FOC_defines.h`, and the C-project has to be re-compiled and downloaded to the target.

The Excel file also creates the scaling factors for the DriveMonitor. For instance the speed scaling changes when the number of pole pairs changes. These values have to be adapted inside the DriveMonitor configuration (CAN control editor).

## 4.2 Project Settings

The complete C-project is DAVe compliant; i.e. the peripheral configuration is performed with DAVe. This has the advantage that the initialization is visible on a user friendly interface and re-configuration or add-on's are straightforward. For example, adding UART or SPI, or removing CAN, is possible with DAVe, and a re-generation will not destroy the existing code.

The setup is done for two different 8051 compilers (Keil and SDCC). Most files are in the C-language but the algorithm in `FOC_functions.c` is in Assembler (ASM).

As the endianness of Keil (big endian) and SDCC (little endian) are different, the interfacing between C and the ASM must be adapted. For the FOC project this is mainly the 16-bit data variables and the structures for the PI controllers. These variables are used in both ASM and C, hence the ASM code is different. Please refer to the following examples:

**Table 1 Reading a 16-bit Variable in Keil / SDCC**

<b>Keil</b>	<code>mov r7,gi_Id_reference</code> <code>mov r6,gi_Id_reference+1</code>
<b>SDCC</b>	<code>mov r7,_gi_Id_reference+1</code> <code>mov r6,_gi_Id_reference</code>

**Table 2 Accessing struct  $Yn[ ]$  in Keil / SDCC**

<b>Keil</b>	<code>inc r0</code> <code>mov a,@r0 ; store yn_max in R2</code> <code>mov r2,a</code> <code>dec r0</code>
<b>SDCC</b>	<code>inc r0</code> <code>inc r0</code> <code>mov a,@r0 ; store yn_max in R2</code> <code>mov r2,a</code> <code>dec r0</code> <code>dec r0</code>

The C-code for Keil and SDCC is identical except for some declarations.

Because of the DriveMonitor communication, the monitoring variables must have dedicated addresses. These addresses are set in `main.c`:

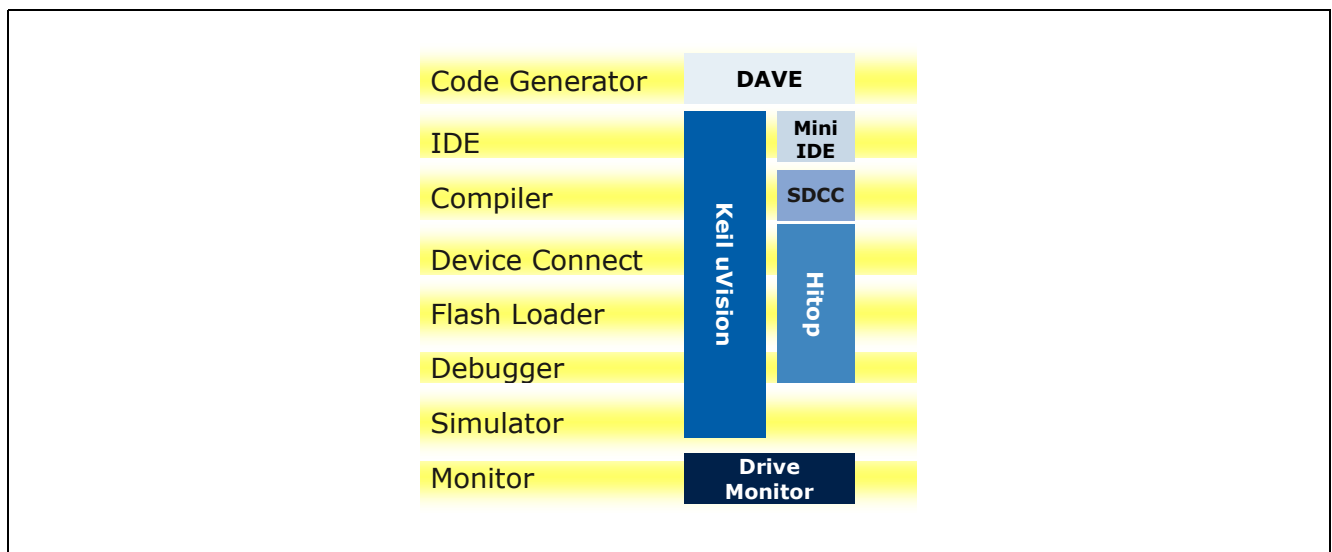
**Table 3 Variable at fixed Address in Keil / SDCC**

<b>Keil</b>	<code>data int gi_angle _at_ (0x0030);</code>
<b>SDCC</b>	<code>data at 0x0030 int gi_angle;</code>

### 4.3 Toolchain

The two compilers Keil and SDCC use different toolchains. Where Keil offers with the  $\mu$ Vision IDE one tool which serves all aspects, the SDCC compiler needs a separate IDE (MiniIDE) and a debugger (Hitop).

Keil offers a free evaluation version which is limited to 2kByte codesize. There is a free toolchain program at IFX (a registration link is given in [Chapter 6](#)). The SDCC based toolchain is completely free of charge.



**Figure 28 Toolchain for 8051 for Compiler Keil and SDCC**

## 4.4 DriveMonitor

The DriveMonitor is a hardware tool which bridges USB to CAN. With the DriveMonitor it is possible to monitor data of the target on a host PC, in real-time.

For more information please refer to the DriveMonitor application note (see [Chapter 6 \[1\]](#)).

### 4.4.1 Configuration

The configuration of the DriveMonitor setup is described in the DriveMonitor application note (see [Chapter 6 \[1\]](#)).

### 4.4.2 Monitoring the Startup Behavior

Different motors have different startup behavior. Starting up a synchronous motor is always quite tricky. The first revolutions have to be in an open loop mode before the FOC algorithm can take over.

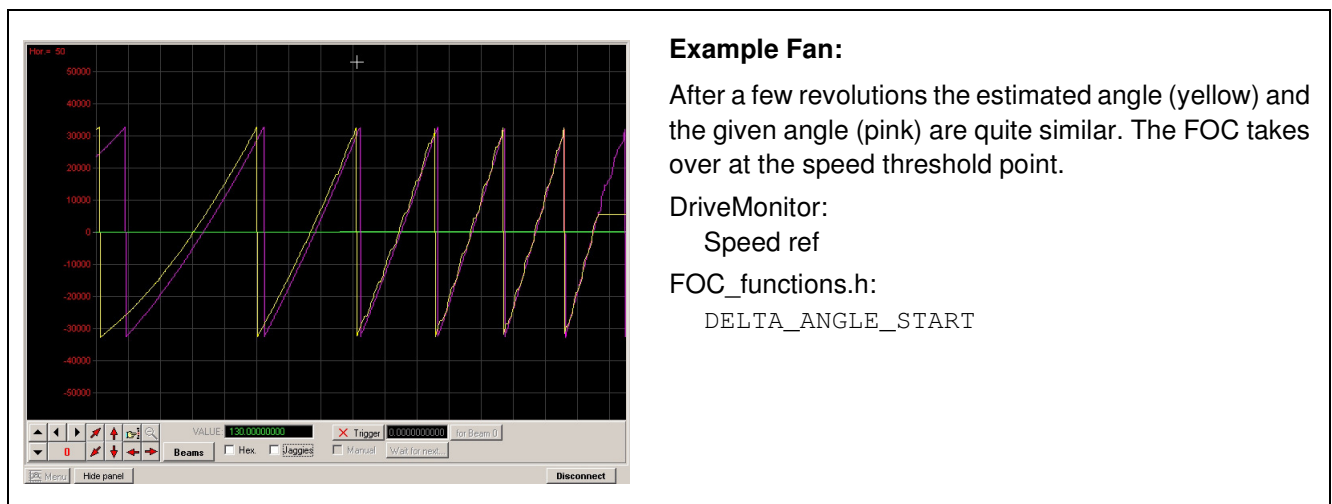
There are some values which primarily influence the startup behavior. These values are to be found in the Excel sheet under *Startup Parameter*, and are discussed here.

After setting the correct R and L values, compilation and downloading the DriveMonitor can be started and used in Setting 1 | Startup.

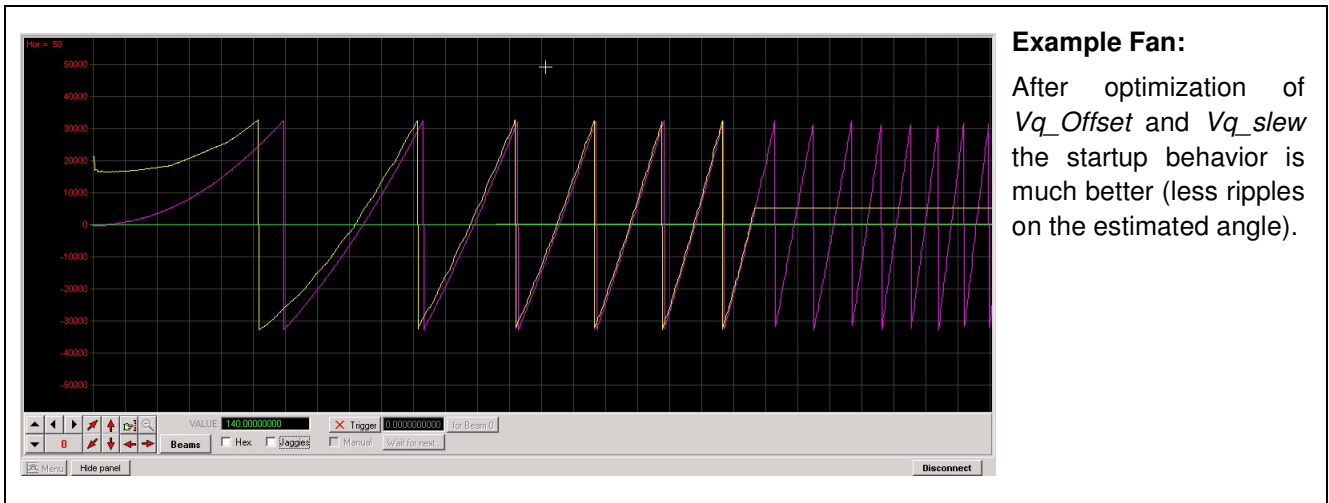
After connecting to the target, the button *Scope Start* should be pressed and then the button *Start*.

The motor should start turning and after a few seconds the *Stop* button can be pressed. The scope shows in this setting two interesting signals:

- pink: `gi_angle` from the Startup Control block which is the given angle following the V/f ramp depending on the slew rate.
- yellow: `gi_Flux_angle` from the Flux Estimator block, which is the estimated (calculated) angle based on the current measurement.



**Figure 29 Startup Behavior of a Fan Before Optimization**



**Figure 30 Startup Behavior of a Fan After Optimization**

The point where the FOC takes over from the V/f control is of special interest. The V/f algorithm increases the speed continuously until the threshold value is reached (Excel: Startup Speed Threshold, FOC\_Functions.h: DELTA\_ANGLE\_THRESHOLD). At this point the current and speed controllers cannot deliver the correct output as they start here. The algorithm can only prepare the controllers accordingly.

Therefore the *current controller's* integral part is preset with current amplitude ( $g_i_{V_q}$ ) based on the V/f calculation, otherwise the amplitude would drop immediately. This is accomplished in Scheduler.c by pre-loading `gt_Iq_control_yn[0:1]`:

```
case STATESTARTUP:
    MotorStart();
    #ifdef CLOSEDLOOP
    if( (gi_delta_angle > 0? gi_delta_angle : -gi_delta_angle) > DELTA_ANGLE_THRESHOLD)
    {
        gb_closed_loop = 1;

        //*****
        /*      preload integral part of PI controller
        //*****
        gt_Iq_control_yn[0] = (gi_V_q>>8) & 0x00ff;
        gt_Iq_control_yn[1] = (gi_V_q) & 0x00ff;

        guc_Scheduler = STATEFOC;
    }
    #endif //CLOSEDLOOP
    break;
```

The *speed controller* must adjust at runtime. Therefore it is advisable that the speed controller is started with an input step; i.e. the difference between the DELTA\_ANGLE\_THRESHOLD and the DEFAULT\_SPEED\_STARTUP.

This can be accomplished by making the Startup Speed Threshold smaller than the Speed Ramp in the Excel sheet.

In DriveMonitor, the value Speed Ramp can be adjusted by *Speed\_ref*; i.e. the user can play with this value.

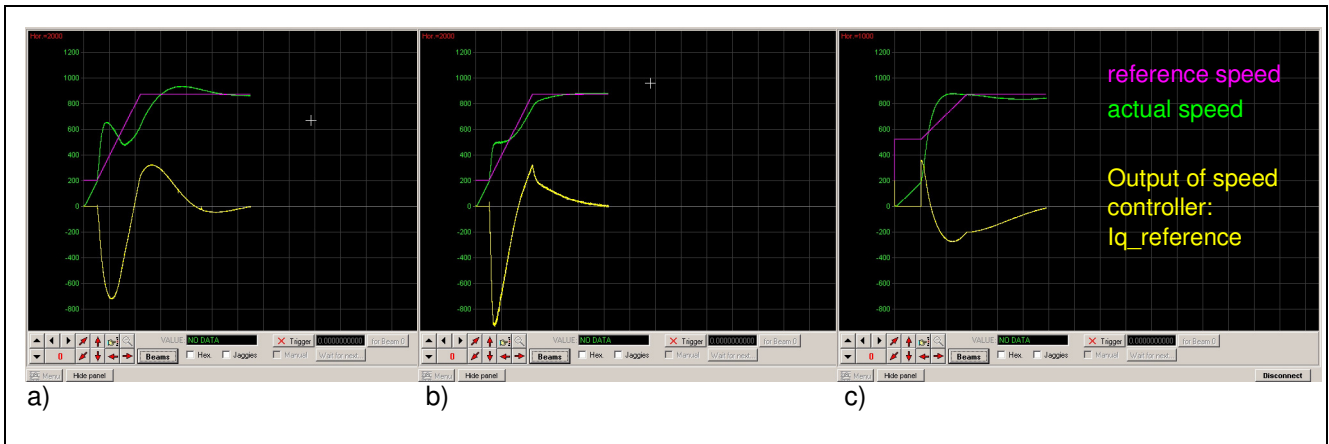


Figure 31 Switching Point from V/f to FOC with Different *Speed\_ref* Settings

In **Figure 31** different threshold conditions can be observed:

- a)  $Speed\_ref = \Delta\_ANGLE\_THRESHOLD$
- b)  $Speed\_ref > \Delta\_ANGLE\_THRESHOLD$
- c)  $Speed\_ref \gg \Delta\_ANGLE\_THRESHOLD$ .

#### 4.4.3 Monitoring the Runtime Behavior

The Runtime behavior can be influenced by the PI parameter settings of the speed and controllers. The DriveMonitor offers the window Setting 2 | Control, where the  $k_i$  and  $k_p$  parameters can be easily adjusted.

$I_{uvq}$  can be used for monitoring the *Scope Speed* or *Scope*.

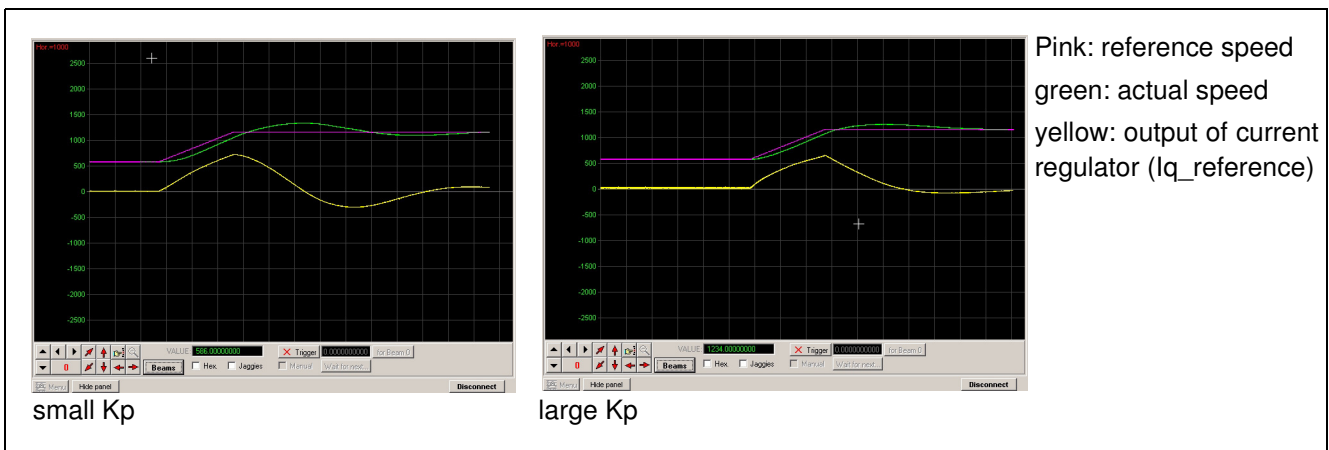


Figure 32 Speed Controller Adjustment, Reaction on a Reference Speed Step



#### 4.4.4 Special Tricks - Field Weakening

The nominal speed of a motor is reached when the phase amplitude is at its maximum value under zero load. This is because the BEMF (Back-Electro-Magnetic-Force) generated voltage increases with the speed and acts against the phase voltage. Therefore the speed saturates. In the FOC algorithm there are two components:

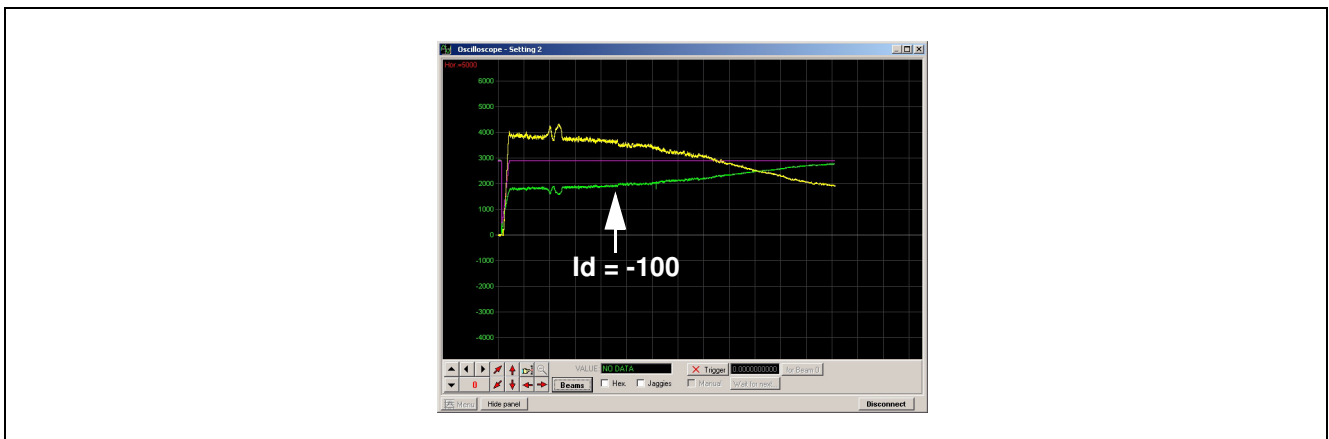
- The torque building current  $I_q$
- The field building current  $I_d$

The latter  $I_d$  is controlled to zero because a PMSM motor generates the field by its magnet. This field component can be easily controlled to a negative value which weakens the magnetic field. With this trick the motor must turn faster than its nominal rated speed.

DriveMonitor allows the field weakening to be applied easily in the Setting 2 | Control. The third Group Entry shows  $I_d$  Ref. A negative value here will let the motor turn faster.

Other interesting observations that can be made in this mode are:

- The overall current consumption will increase - watch the power supply current.
- The torque will decrease in field weakening - this is obvious as the resulting stator flux vector is no longer rectangular to the rotor flux vector.



**Figure 33 Field Weakening: With Negative Field Building Component  $I_d$  the Speed Increases Above the Nominal Speed.**

### 5 FOC Building Blocks Overview Drawing

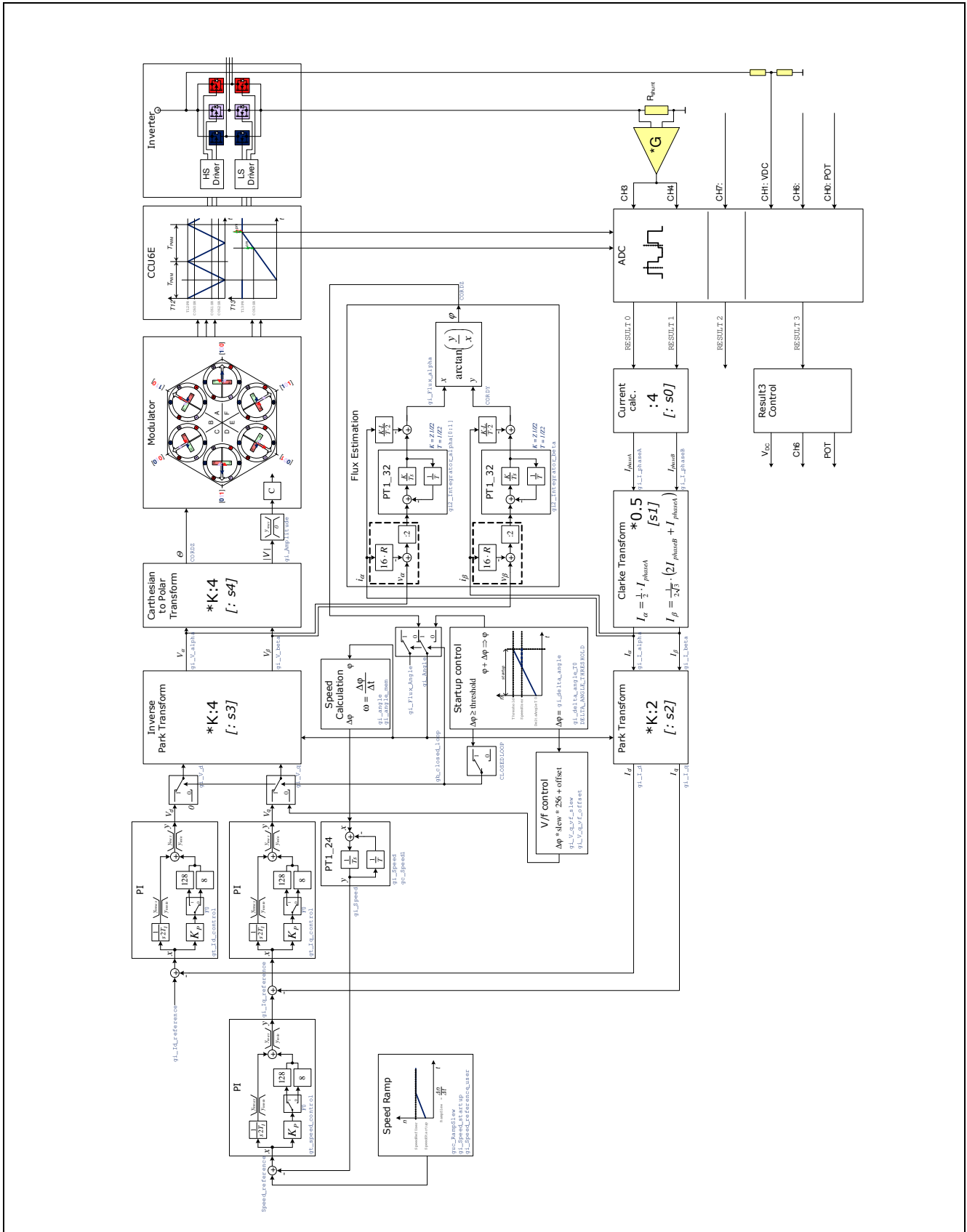


Figure 34 FOC Building Blocks

## 6 References

- [1] Application note - AP0807120\_DriveMonitor
- [2] Application note - AP0808810\_XC878DriveCard
- [3] Application note - AP9000110\_LVinverter
- [4] Application note - AP1616010\_DriveCard\_XE164
- [5] Application note - AP0805910\_Sensorless\_FOC
- [6] Application note - AP16165\_XE164\_Sensorless\_FOC
- [7] Link to XC878 Starter-kit CD download for getting started with XC800 toolchain - [www.infineon.com/XC878](http://www.infineon.com/XC878)
- [8] Link to free toolchain program - [www.infineon.com/Freetools](http://www.infineon.com/Freetools)
- [9] Link to FOC Drive page - [www.infineon.com/FOCDRIVE](http://www.infineon.com/FOCDRIVE)

[www.infineon.com](http://www.infineon.com)

Published by Infineon Technologies AG