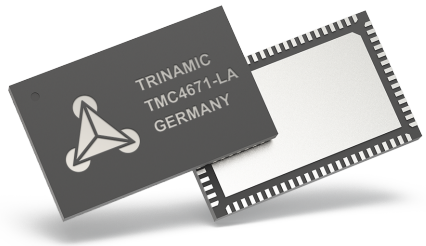
TMC4671 Data Sheet

IC Version V1.3 | 19-101065, Rev 1: 02/24

The TMC4671 is a fully integrated servo controller, providing field-oriented control (FOC) for brushless DC motor (BLDC)/permanent magnet synchronous motor (PMSM) and 2-phase stepper motors as well as DC motors and voice coils. All control functions are implemented in hardware. Integrated analog-to-digital converters (ADCs), position sensor interfaces, and position interpolators enable a fully functional servo controller for a wide range of servo applications.



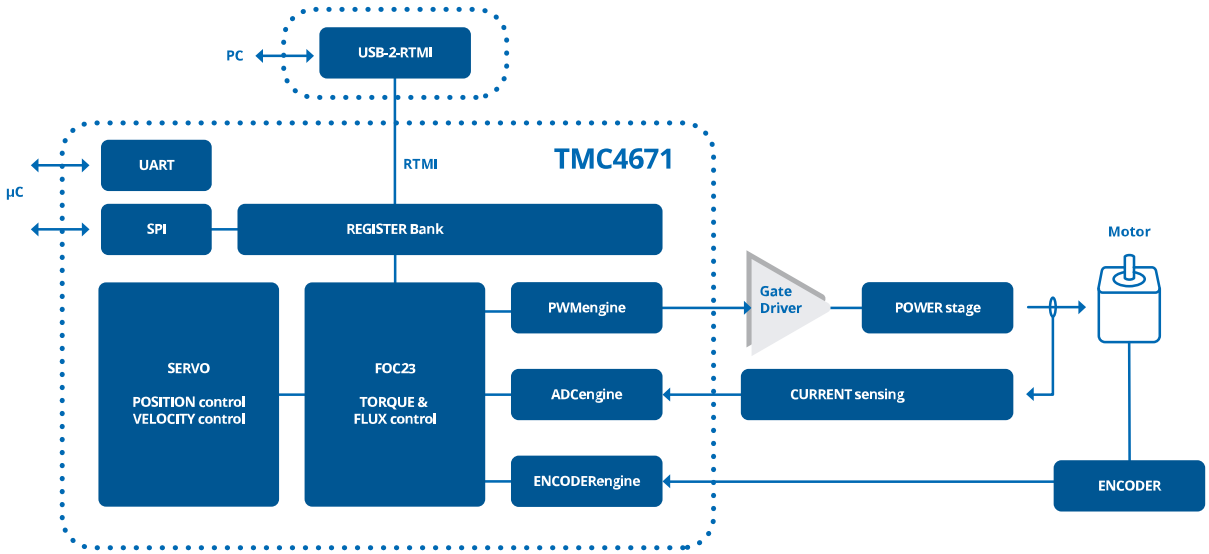
Features

- Servo Controller w/ Field Oriented Control (FOC)
- Torque Control (FOC), Velocity Control, Position Control
- Integrated ADCs, $\Delta\Sigma$-ADC Frontend
- Encoder Engine: Hall Analog/Digital, Encoder Analog/Digital
- Supports 3-Phase PMSM/BLDC, 2-Phase Stepper Motors, and 1-Phase DC Motors
- Fast PWM Engine (25kHz . . . 100kHz)
- Application SPI + Debug (UART, SPI)
- Step-Direction Interface (S/D)

Applications

- Robotics
- Pick and Place Machines
- Factory Automation
- E-Mobility
- Laboratory Automation
- Blowers
- Pumps

Simplified Block Diagram



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Read entire documentation.

Contents

1 Order Codes

Order Code	Description	Size
TMC4671-LA	TMC4671 FOC Servo Controller IC	10.5mm x 6.5mm
TMC4671-EVAL	TMC4671 Evaluation Board	55mm x 85mm
TMC4671-BOB	TMC4671 Breakout Board	38mm x 40mm
Landungsbruecke	MCU Board	85mm x 55mm
TMC6100-EVAL	Power Stage Board	85mm x 55mm
TMC-UPS-2A24V-A-EVAL	Power Stage Board	85mm x 55mm
TMC-UPS-10A70V-A-EVAL	Power Stage Board	85mm x 55mm
TMC4671-2A24V-EV-KIT	Evaluation Kit	—
TMC4671-10A70V-EV-KIT	Evaluation Kit	—
USB-2-RTMI	Interface Adapter to use RTMI	40mm x 20mm

Table 1: Order Codes

2 Functional Summary

- **Servo Controller with Field Oriented Control (FOC)**
  - Torque (and flux) control mode
  - Velocity control mode
  - Position control mode
- **Control Functions/PI Controllers**
  - Programmable clipping of inputs and outputs of interim results
  - Integrator windup protection for all controllers
  - Status output with programmable mask for internal status signal selection
- **Supported Motor Types**
  - FOC3 : 3-phase permanent magnet synchronous motors (PMSM)/brushless DC motor (BLDC)
  - FOC2 : 2-phase stepper motors
  - FOC1 : 1-phase brushed DC motors, or linear voice coil motors
- **ADC Engine with Offset Correction and Scaling**
  - Integrated Delta Sigma ADCs for current sense voltage, supply voltage, analog encoder, AGPIs
  - Interface for isolated external current sensing Delta Sigma modulators
- **Position Feedback**
  - Open loop position generator (programmable [rpm], [rpm/s]) for initial setup
  - Digital incremental encoder (ABN resp. ABZ, up to 2 MHz)
  - Secondary digital incremental encoder
  - Digital hall sensor interface (H1, H2, H3 resp. H_U, H_V, H_W) with interim position interpolation
  - Analog encoder/analog hall sensor interface (SinCos (0°, 90°) or 0°, 120°, 240°)
  - Position target, velocity, and target torque filters (Biquad)
  - Multi-turn position counter (32-bit)
- **PWM Engine Including SVPWM**
  - Programmable PWM frequency within the range of 25 kHz …100 kHz
  - PWM auto scaling for transparent change of PWM frequency during motion
  - Programmable brake-before-make (BBM) times (0 ns …2.5 μs) for digital gate control signals
  - Single bit SVPWM control (on/off) for space vector modulation (switchable during operation)

- **SPI Application Communication Interface**
  - 40 bit datagram length (1 readwrite bit + 7 address bits + 32 data bits)
  - Immediate SPI read response (register read access by single datagram)
  - SPI clock frequency fSCK up to 2 MHz (8 MHz write, 8 MHz read w/ 500 ns pause after address)
- **TRINAMIC Real Time Monitoring Interface**
  - High frequency sampling of real-time data through TRINAMIC's real-time monitoring system
  - Only single 10-pin high density connector on PCB needed
  - Enables frequency response identification and auto tuning options with TRINAMIC's integrated development environment (IDE)
- **UART Debug Interface**
  - Three pin (GND, RxD, TxD) 3.3 V UART interface (1N8; 9600 (default), 115200, 921600, 3M bps)
  - Available as port for external position sensors (example, absolute encoder together with processor)
  - Transparent register access parallel to embedded user application interface (SPI)
- **Supply Voltages**
  - 5V and 3.3V; V_{CCCORE} is internally generated
- **IO Voltage**
  - 3.3V for all digital IOs (choosable by V_{CCIO} supply)
  - 5V common mode analog input voltage range (1.25V ... 2.5V differential operating range)
- **Clock Frequency**
  - 25 MHz (from external oscillator)
- **Packages**
  - QFN76

3 FOC Basics

This section gives a short introduction into some basics of field-oriented control (FOC) of electric motors.

3.1 Why FOC?

The field-oriented control (FOC), alternatively named vector control (VC), is a method for the most energy-efficient way of turning an electric motor.

3.2 What is FOC?

The FOC was independently developed by K. Hasse, TU Darmstadt, 1968, and by Felix Blaschke, TU Braunschweig, 1973. The FOC is a current regulation scheme for electro motors that takes the orientation of the magnetic field and the position of the rotor of the motor into account, regulating the strength in such way that the motor gives that amount of torque requested as target torque. The FOC maximizes active power and minimizes idle power, that finally results in power dissipation, by intelligent closed-loop control as illustrated in Figure 1.

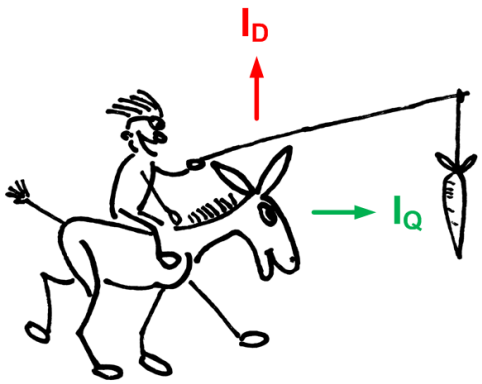


Figure 1: Illustration of the FOC Basic Principle by Cartoon: Maximize Active Power and Minimize Idle Power and Power Dissipation by Intelligent Closed-Loop Control.

3.3 Why FOC as Pure Hardware Solution?

The initial setup of the FOC is usually very time consuming and complex, although source code is freely available for various processors. This is because the FOC has many degrees of freedom that all need to fit together in a chain to work.

The hardware FOC as an existing standard building block drastically reduces the effort in system setup. With that off the shelf building block, the starting point of FOC is the setup of the parameters for the FOC. Setting up and implement the FOC itself and building and programming required interface blocks is no longer necessary. The real parallel processing of hardware blocks decouples the higher lever application software from high speed real-time tasks and simplifies the development of application software. With the TMC4671, the user is free to use its qualified CPU together with its qualified tool chain, freeing the user from fighting with processer-specific challenges concerning interrupt handling and direct memory access. There is no need for a dedicated tool chain to access the TMC4671 registers and to operate it - just SPI (or UART) communication needs to be enabled for any given CPU.

The hardware integration of the FOC drastically reduces the number of required components and reduces the required PCB space. This is in contrast to classical FOC servos formed by motor block and separate

controller box wired with motor cable and encoder cable. The high integration of FOC, together with velocity controller and position controller, enables the FOC as a standard peripheral component that transforms digital information into physical motion. Compact size together with high performance and energy efficiency, especially for battery powered mobile systems, are enabling factors when embedded goes autonomous.

3.4 How Does FOC Work?

Two force components act on the rotor of an electric motor. One component is just pulls in radial direction (ID), where the other component applies torque by pulling tangentially (IQ). The ideal FOC performs a closed loop current control that results in a pure torque generating current IQ – without direct current ID.

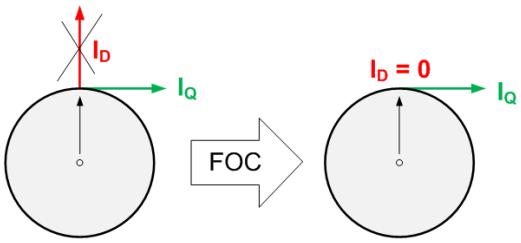


Figure 2: FOC Optimizes Torque by Closed-Loop Control While Maximizing IQ and Minimizing ID to 0

From top point of view, the FOC for 3-phase motors uses three phase currents of the stator interpreted as a current vector (Iu; Iv; Iw) and calculates three voltages interpreted as a voltage vector (Uu; Uv; Uw), taking the orientation of the rotor into account in a way that only a torque generating current IQ results.

From top point of view, the FOC for 2-phase motors uses two phase currents of the stator interpreted as a current vector (Ix; Iy) and calculates two voltages interpreted as a voltage vector (Ux; Uy), taking the orientation of the rotor into account in a way that only a torque generating current IQ results.

To do so, the knowledge of some static parameters (number of pole pairs of the motor, number of pulses per revolution of an used encoder, orientation of encoder relative to magnetic axis of the rotor, count direction of the encoder) is required together with some dynamic parameters (phase currents, orientation of the rotor).

The adjustment of P parameter and P and I parameters of two PI controllers for closed loop control of the phase currents depends on electrical parameters of the motor (resistance, inductance, back EMF constant of the motor that is also the torque constant of the motor, supply voltage).

3.5 What is Required for FOC?

The FOC needs to know the direction of the magnetic axis of the rotor of the motor in reference to the magnetic axis of the stator of the motor. The magnetic flux of the stator is calculated from the currents through the phases of the motor. The magnetic flux of the rotor is fixed to the rotor and thereby determined by an encoder device.

For the FOC, measure the currents through the coils of the stator and the angle of the rotor. The measured angle of the rotor needs to be adjusted to the magnetic axes.

The challenge of the FOC is the high number of degrees of freedom in all parameters.

3.5.1 Coordinate Transformations - Clarke, Park, iClarke, iPark

The FOC requires different coordinate transformations formulated as a set of matrix multiplications. These are the Clarke Transformation (Clarke), the Park Transformation (Park), the inverse Park Transformation (iPark), and the inverse Clarke Transformation (iClarke). The Park transformation is also known as DQ transformation, whereas the Clarke transformation is known as $\alpha\beta$ transformation.

The TMC4671 takes care of the required transformations so the user no longer has to fight with implementation details of these transformations.

3.5.2 Measurement of Stator Coil Currents

The measurement of the stator coil currents is required for the FOC to calculate a magnetic axis out of the stator field caused by the currents flowing through the stator coils.

Coil current stands for motor torque in context of FOC. This is because motor torque is proportional to motor current, defined by the torque constant of a motor. In addition, the torque depends on the orientation of the rotor of the motor relative to the magnetic field produced by the current through the coils of the stator of the motor.

3.5.3 Stator Coil Currents I_U, I_V, I_W and Association to Terminal Voltages U_U, U_V, U_W

The correct association between stator terminal voltages U_U, U_V, U_W and stator coil currents I_U, I_V, I_W is essential for the FOC. In addition to the association, the signs of each current channel must fit. Signs of the current can be adapted numerically by the ADC scaler. The mapping of ADC channels is programmable through configuration registers for the ADC selector. Initial setup is supported by the integrated open loop encoder block, that can support the user to turn a motor open loop.

3.5.4 IgainADC[A/LSB] - ADC Integer Current Value to Real World Unit

Together with ADC_I0_SCALE and ADC_I0_OFFSET and ADC_I1_SCALE and ADC_I1_OFFSET, measured ADC currents represented as 16-bit signed interger numbers (s16) represent real world currents. Multiplication of integer current value with gain scaling factor in unit Ampere per low significant bit (LSB) gives the real world value of current in unit Ampere.

$$\begin{aligned} I_0[A] &= \text{IgainADC}[A/LSB] \times \text{ADC\_I0} \\ I_1[A] &= \text{IgainADC}[A/LSB] \times \text{ADC\_I1} \end{aligned} \tag{1}$$

Different scalings between two associated current ADC channels can be trimmed by programing ADC_I0_SCALE and ADC_I1_SCALE. The IgainADC[A/LSB] must be determined from ADC gain factors, ADC reference voltage selection, and actual ADC scaling factor settings.

3.5.5 UgainADC[V/LSB] - ADC Integer Voltage Value to Real World Unit

Measured ADC voltages represented as 16-bit signed interger numbers (s16) represent real world voltages. Multiplication of integer voltage value with gain scaling factor in unit Volt per low significant bit (LSB) gives the real world value of voltage in unit Volt.

$$U[V] = \text{UgainADC}[V/LSB] \times \text{ADC\_U} \tag{2}$$

Determine the UgainADC[V/LSB] from ADC gain factors, actual ADC gains, and ADC reference voltage settings.

3.5.6 Measurement of Rotor Angle

Determination of the rotor angle is either done by sensors (digital encoder, analog encoder, digital hall sensors, analog hall sensors) or sensorless by a reconstruction of the rotor angle. Currently, there are no sensorless methods available for FOC that work in a general purpose way as a sensor down to velocity zero.

The TMC4671 does not support sensorless FOC.

3.5.7 Measured Rotor Angle vs. Magnetic Axis of Rotor vs. Magnetic Axis of Stator

The rotor angle, measured by an encoder, must be adjusted to the magnetic axis of the rotor. This is because an incremental encoder has an arbitrary orientation relative to the magnetic axis of the rotor, and the rotor has an arbitrary orientation to magnetic axis of the stator.

The direction of counting depends on the encoder, its mounting, and wiring and polarities of encoder signals and motor type. So, the direction of encoder counting is programmable for comfortable definition for a given combination of motor and encoder.

3.5.7.1 Direction of Motion - Magnetic Field vs. Position Sensor

For FOC, it is essential that the direction of revolution of the magnetic field is compatible with the direction of motion of the rotor position reconstructed from encoder signals: For revolution of magnetic field with positive direction, the decoder position must turn into the same positive direction. For revolution of magnetic field with negative direction, the decoder position must turn into the same negative direction.

With an absolute encoder, once adjusted to the relative orientation of the rotor and to the relative orientation of the stator, it is possible to start the FOC without initialization of the relative orientations.

3.5.7.2 Bang-Bang Initialization of the Encoder

A Bang-Bang initialization is an initialization where the motor is forced with high current into a specific position. For Bang-Bang initialization, set a current into direction D that is strong enough to move the rotor into the desired direction. Other initialization methods ramp up the current smoothly and adjust the current vector to rotor movement detected by the encoder.

3.5.7.3 Encoder Initialization Using Hall Sensors

The encoder can be initialized using digital hall sensor signals. Digital hall sensor signals give absolute positions within each electrical period with a resolution of sixty degrees. If the hall sensor signals are used to initialize the encoder position on the first change of a hall sensor signal, an absolute reference within the electrical period for commutation is given.

3.5.7.4 Minimum Movement Initialization of the Encoder

For minimal movement initialization of the encoder, slowly increase a current into direction D and adjust an offset of the measured angle in a way that the rotor of the motor does not move during initialization while the offset of the measured angle is determined.

3.5.8 Knowledge of Relevant Motor Parameters and Position Sensor (Encoder) Parameters

3.5.8.1 Number of Pole Pairs of a Motor

The number of pole pairs is an essential motor parameter. It defines the ratio between electrical revolutions and mechanical revolutions. For a motor with one pole pair, one mechanical revolution is equivalent to one electrical revolution. For a motor with number of pole pairs (NPP), one mechanical revolution is equivalent to npp electrical revolutions, with n = 1, 2, 3, 4, ...

Some define the number of poles (NP) instead of number of pole pairs (NPP) for a motor, which results in a factor of two that might cause confusion. For the TMC4671 NPP are used.

3.5.8.2 Number of Encoder Positions per Revolution

For the encoder, the number of positions per revolution (PPR) is an essential parameter. The number of positions per revolution is essential for the FOC.

Some encoder vendors give the number of lines per revolution (LPR) or just named line count (LC) as encoder parameter. Line count and positions per revolution might differ by a factor of four. This is because of the quadrature encoding (A signal and B signal with phase shift) that gives four positions per line, determining of the direction of revolution. Some encoder vendors associate counts per revolution (CPR) or pulses per revolution associated to PPR acronym.

The TMC4671 uses positions per revolution (PPR) as encoder parameter.

3.5.9 Proportional Integral (PI) Controllers for Closed Loop Current Control

Last but not least, two PI controllers are required for the FOC. The TMC4671 is equipped with two PI controllers, one for control of torque generating current I_Q and one to control current I_D to zero.

3.5.10 Pulse Width Modulation (PWM) and Space Vector Pulse Width Modulation (SVPWM)

The PWM power stage is a must-have for energy efficient motor control. The PWM engine of the TMC4671 just needs a couple of parameters to set PWM frequency fPWM and switching pauses for both high side switches tBBM_H and low side switches tBBM_L. Some control bits are for the programming of power switch polarities for maximum flexibility in the selection in gate drivers for the power MOS-FETs. An additional control bit selects SVPWM on or off. The TMC4671 allows the change of PWM frequency by a single parameter during operation.

With this, the TMC4671 is advanced compared to software solutions where PWM and SVPM configuration of CPU internal peripherals normally need settings of many parameters.

3.5.11 Orientations, Models of Motors, and Coordinate Transformations

The orientation of magnetic axes (U, V, W for FOC3 resp. X, Y for FOC2) is essential for the FOC together with the relative orientation of the rotor. Here, the rotor is modeled by a bar magnet with one pole pair (n_pole_pairs = 1) with magnetic axis in north-south direction.
The actual magnetic axis of the stator (formed by the motor coils) is determined by measurement of the coil currents.
The actual magnetic axis of the rotor is determined by incremental encoder or hall sensors. Incremental encoders need an initialization of orientation, whereas hall sensors give an absolute orientation, but with low resolution. A combination of hall sensor and incremental encoder is useful for start-up initialization.

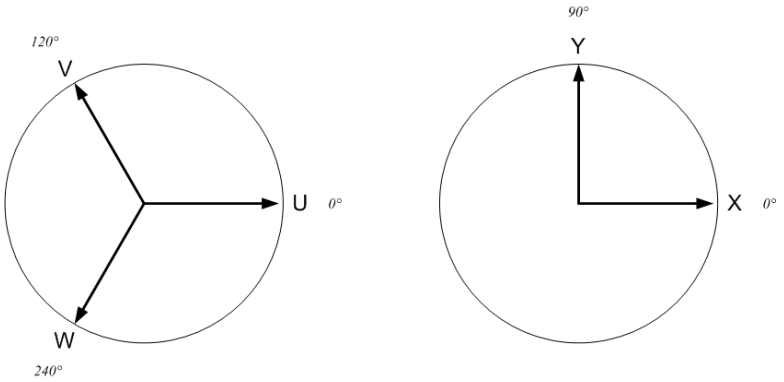


Figure 3: Orientations UVW (FOC3) and XY (FOC2)

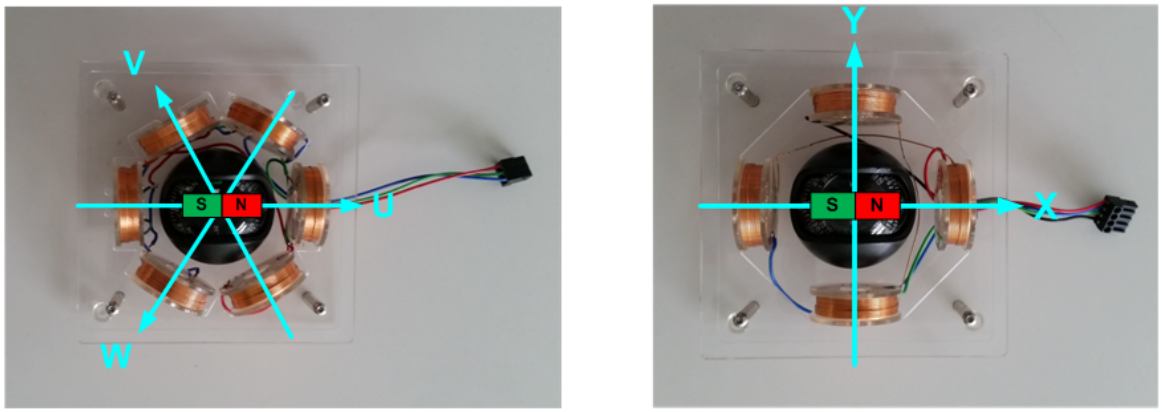


Figure 4: Compass Motor Model w/ Three Phases UVW (FOC3) and Compass Motor Model w/ Two Phases (FOC2)

4 Functional Description

The TMC4671 is a fully integrated controller for field-oriented control (FOC) of either one 3-phase brushless motor (FOC3) or one 2-phase stepper motor (FOC2) as well as 1-phase DC motor or voice coil actuator (FOC1). Containing the complete control loop core architecture (position, velocity, torque), the TMC4671 also has the required peripheral interfaces for communication with an application controller, for feedback (digital encoder, analog interpolator encoder, digital hall with interpolator, analog inputs for current and voltage measurement), and helpful additional IOs. The TMC4671 supports highest control loop speed and PWM frequencies.

The TMC4671 is the building block that takes care of all real-time critical tasks of field-oriented motor control. It decouples the real-time field-oriented motor control and its real-time sub-tasks such as current measurement, real-time sensor signal processing, and real-time PWM signal generation from the user application layer, as outlined by Figure 5.

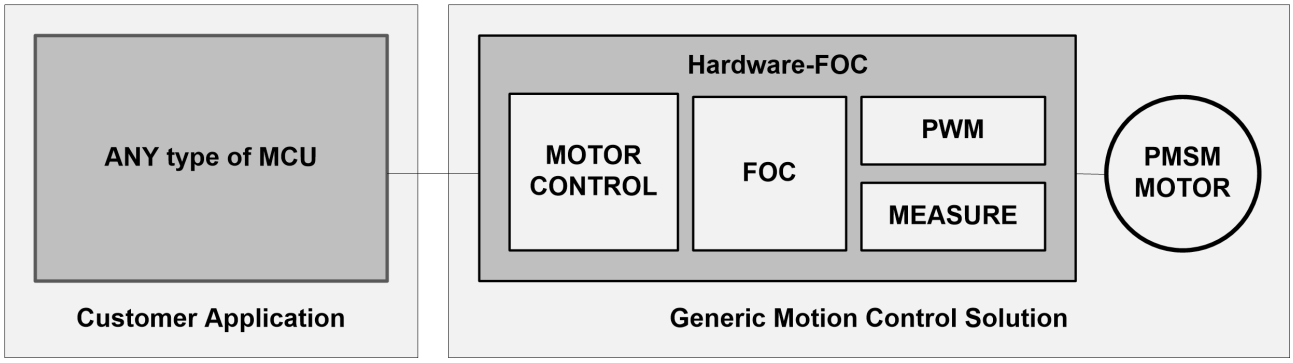


Figure 5: Hardware FOC Application Diagram

4.1 Functional Blocks

The application interface, register bank, ADC engine, encoder engine, FOC torque PI controller, velocity PI controller, position P controller, and PWM engine make up the TMC4671.

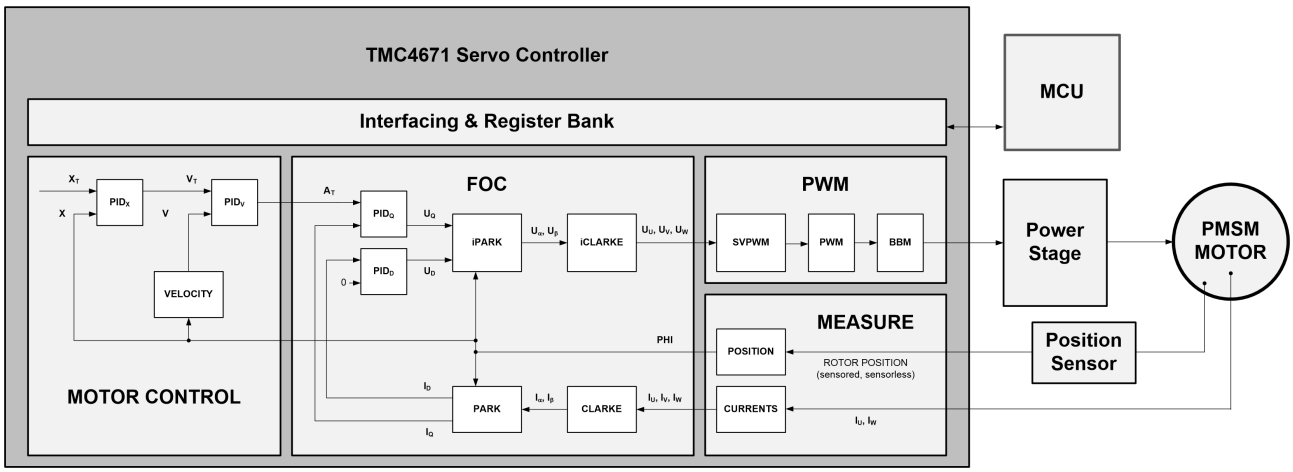


Figure 6: Hardware FOC Block Diagram

The ADC engine interfaces the integrated ADC channels and maps raw ADC values to signed 16-bit (s16) values for the inner FOC current control loop based on programmable offset and scaling factors. The FOC torque PI controller forms the inner base component including required transformations (Clark, Park, inverse Park, inverse Clark). All functional blocks are pure hardware.

4.2 Communication Interfaces

The TMC4671 is equipped with an SPI subnode user interface for access to all registers of the TMC4671. The SPI subnode user interface is the main application interface.
An additional UART interface is intended for system setup. With this interface, access all registers of the TMC4671 in parallel to the application accessing them through the SPI communication interface - through the user's firmware or through evaluation boards and the TMCL-IDE. The data format of the UART interface is similar to the SPI communication interface - SPI 40 bit datagrams sent to the TMC4671 and SPI 40 bit datagrams received by the MCU vs. five bytes sent through UART and five bytes received through UART. Sending a burst of different real-time data for visualization and analysis through the TMCL-IDE can be triggered using special datagrams. With this, set up an embedded application together with the TMCL-IDE, without having to write a complex set of visualization and analysis functions. It is possible to focus on the application.
The TMC4671 is also equipped with an additional SPI main node interface (TRINAMIC real-time monitoring interface, DBGSPI) for high-speed visualization of real-time data together with the TMCL-IDE.

4.2.1 SPI Subnode User Interface

The SPI of the TMC4671 for the user application has an easy command and control structure. The TMC4671 user SPI acts as a subnode. The SPI datagram length is 40 bit with a clock rate up to 8 MHz (1 MHz for the TMC4671-ES).

- The MSB (bit#39) is sent first. The LSB (bit#0) is sent last.
- The MSB (bit#39) is the WRITE_notREAD (WRnRD) bit.
- The bits (bit#39 to bit#32) are the address bits (ADDR).
- Bits (bit#31) to (bit#0) are 32 data bits.

The SPI of the TMC4671 immediately responds within the actual SPI datagram on read and write for ease-of-use communication and uses SPI mode 3 with CPOL = 1 and CPHA = 1. There is no support to daisy chain multiple ICs using MISO and MOSI. Datagrams longer or shorter than 40 bit should be avoided and might lead to unexpected behaviour.

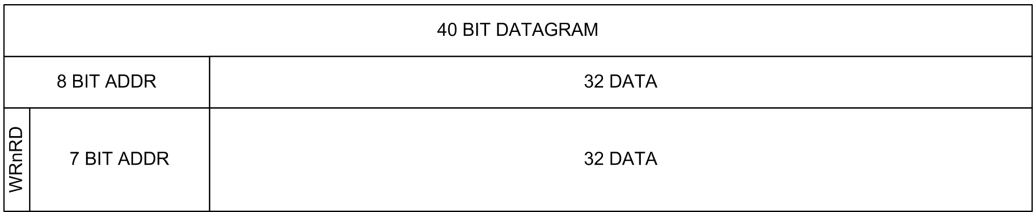


Figure 7: SPI Datagram Structure

A simple SPI datagram example:

```
0x8100000000 // 1st write 0x00000000 into address 0x01 (CHIPINFO_ADDR)
0x0000000000 // 2nd read register 0x00 (CHIPINFO_DATA), returns 0x34363731 <=> ACSII "4671"
```

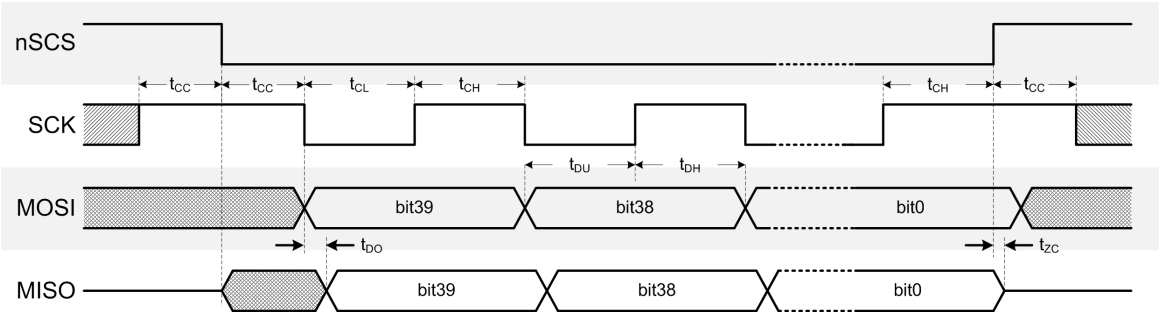


Figure 8: SPI Timing

SPI Interface Timing	Characteristics, fCLK = 25MHz				
Parameter	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
SCK valid before or after change of nSCS	t_{CC}	62.5			ns
nSCS high time	t_{CSH}	62.5			ns
nSCS low time	t_{CSL}	62.5			ns
SCK high time	t_{CH}	62.5			ns
SCK low time	t_{CL}	62.5			ns
tSCKpause time after read address byte	$t_{SCKpause}$	500			ns
SCK frequency with tSCKpause after write address	$f_{SCKpauseWR}$			8	MHz
SCK frequency for write access without pause	f_{SCKwr}			8	MHz
SCK frequency with tSCKpause after read address	$f_{SCKpauseRD}$			8	MHz
SCK frequency for read access without tSCKpause	f_{SCKrd}			2	MHz
MOSI setup time before rising edge of SCK	t_{DU}	62.5			ns
MOSI hold time after falling edge of SCK	t_{DH}	62.5			ns
MISO data valid time after falling edge of SCK	t_{DO}		20		ns

Table 2: SPI Timing Parameter

NOTE

SPI write access can be performed up to 8 MHz SPI clock frequency. SPI read access can be performed up to 8 MHz SPI clock frequency if a pause of at least 500 ns is inserted after transfer of the address byte of the SPI datagram. Without a pause of 500 ns after address byte, SPI read access can be performed up to 2 MHz SPI clock frequency.

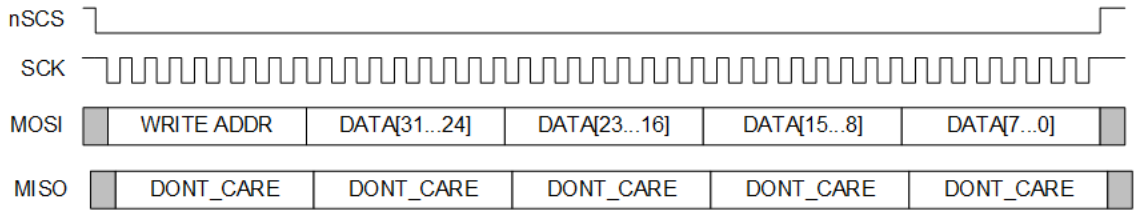


Figure 9: SPI Timing of Write Access without Pause with fSCK up to 8MHz

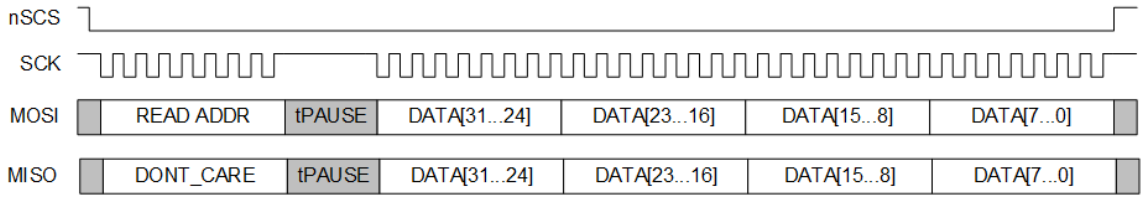


Figure 10: SPI Timing of Read Access with Pause (tPAUSE) of 500 ns with fSCK up to 8MHz

4.2.2 TRINAMIC Real-Time Monitoring Interface (SPI Main Node)

The TRINAMIC real-time monitoring interface (RTMI, SPI main node) is an additional fast interface enabling real-time identification of motor and system parameters. Check configuration and access registers in the TMC4671 through the TMCL-IDE with its build-in configuration wizards for FOC setup in parallel to the user firmware. TRINAMIC provides a monitoring adapter to access the interface, which connects easily to a single 10-pin high density connector (type: Hirose DF20F-10DP-1V) on an PCB or evaluation board. If the interface is not needed, pins can be left open or can be used as GPIOs according to the specification. The connector must be placed near the TMC4671. Its assignment is pictured in Figure 11.

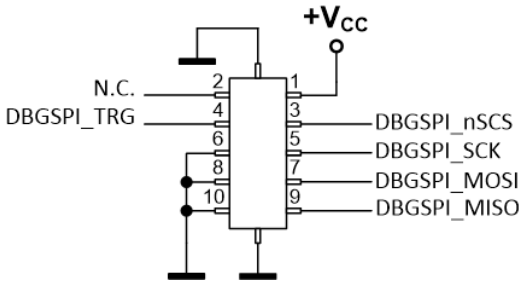


Figure 11: Connector for Real-Time Monitoring Interface (Connector Type: Hirose DF20F-10DP-1V)

4.2.3 UART Interface

The UART interface is a simple three pin (GND, RxD, TxD) 3.3V UART interface with up to 3 Mb/s transfer speed with one start bit, eight data bits, one stop bit, and no parity bits (1N8). The default speed is 9600 bps. Other supported speeds are 115200 bps, 921600 bps, and 3000000 bps. The speed must be changed manually in register 0x79 UART_BPS.

NOTE The baudrates must be entered as hexadecimal numbers.Table 3 lists the register value and its corresponding baudrate.

Value of Register 0x79	Selected Baudrate
0x00009600	9600 bps
0x00115200	115200 bps
0x00921600	921600 bps
0x03000000	3000000 bps

Table 3: Possible Baudrates and Corresponding Values for Register 0x79

With a 3.3V-UART-to-USB adapter cable (example, FTDI TTL-232R-RPi), use the full maximum data rate. The UART port enables in-system-setup-support by multiple-ported register access.

A UART datagram consists of five bytes - similar to the datagrams of the SPI. In contrast to SPI, the UART interface has a time out feature. So, the five bytes of a UART datagram must be sent within one second. A pause of more than one second causes a time out and sets the UART protocol handler back into IDLE state. In other words, waiting for more than one second in sending through UART ensures the UART protocol handler is in IDLE state.

A simple UART example (similar to the simple SPI example):

```
0x81 0x00 0x00 0x00 0x00 // 1st write 0x00000000 into address 0x01 (CHIPINFO_ADDR)
0x00 0x00 0x00 0x00 0x00 // 2nd read register 0x00 (CHIPINFO_DATA), returns 0x34363731
```

Why UART Interface? It might become necessary during the system setup phase to simply access some internal registers without disturbing the application, without changing the actual user application software, and without adding additional debugging code that might disturb the application software itself. The UART enables this supporting function. In addition, it also enables easy access for monitoring purposes with its very simple and direct five byte protocol. The UART interface is available to write periodically positions into the TMC4671 through an external CPU used as a protocol translator to enable absolute encoders for the TMC4671.

NOTE The UART-write response of the TMC4671 depicted in Figure 13 does not contain any useful information. It is only used to signal that the IC received and processed the write datagram sent by the controller.

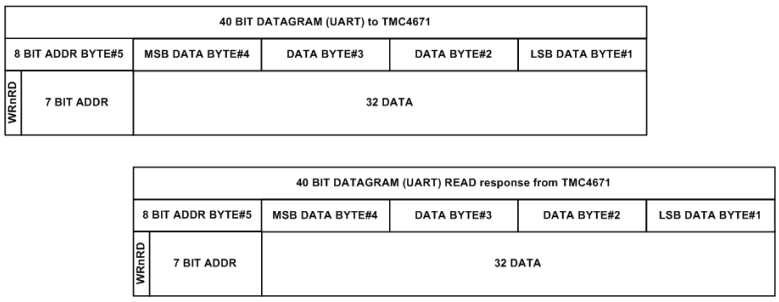


Figure 12: UART Read Datagram (TMC4671 Register Read through UART)

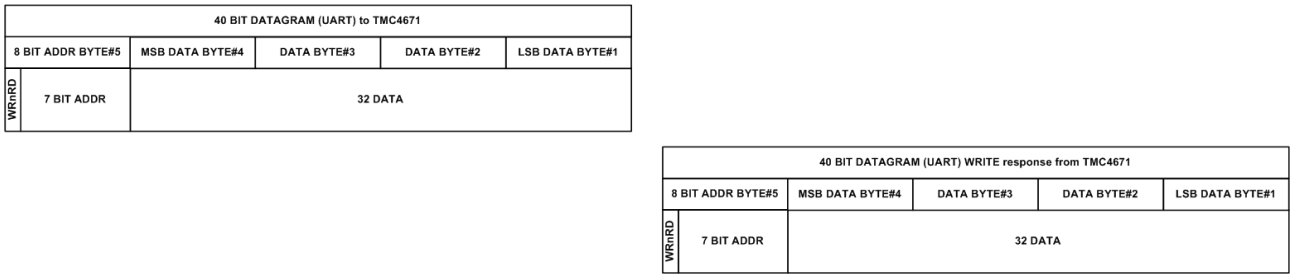


Figure 13: UART Write Datagram (TMC4671 Register Write through UART)

4.2.4 Step/Direction Interface

It is possible to manipulate the target position through the step direction interface. It can be enabled by setting the STEP_WIDTH (s32) register to a proper step width. The power-on default value of STEP_WIDTH is 0 that causes position target update with 0 step width that is no stepping. With STEP_WIDTH $\neq$ 0, each step pulse on STEP input causes incrementing or decrementing of target position depending on the polarity of DIR input. For positive STEP_WIDTH, DIR = 0 causes incrementing and the DIR = 1 causes decrementing of the target position. For negative STEP_WIDTH, DIR = 0 causes decrementing and DIR = 1 causes incrementing of the target position. This is because the STEP_WIDTH is represented as a signed number. The maximum STEP-pulse frequency is limited to half of the utilized oscillator frequency. With, for example, a 25 MHz oscillator, STEP-pulse frequency should not exceed 12.5 MHz.

4.2.5 Single Pin Interface

The TMC4671 can be operated in motion modes, in which the main target value is calculated from either a PWM input signal on PIN PWM_I or by analog input to AGPI_A.

Number	Motion Mode	Using PWM_I or AGPI_A
0	Stopped Mode	No
1	Torque Mode	No
2	Velocity Mode	No
3	Position Mode	No
4 to 7	Reserved	No
8	UQ UD Ext Mode	No

Number	Motion Mode	Using PWM_I or AGPI_A
9	(Reserved)	No
10	AGPI_A Torque Mode	AGPI_A
11	AGPI_A Velocity Mode	AGPI_A
12	AGPI_A Position Mode	AGPI_A
13	PWM_I Torque Mode	PWM_I
14	PWM_I Velocity Mode	PWM_I
15	PWM_I Position Mode	PWM_I

Table 4: Single Pin Interface Motion Modes

Registers SINGLE_PIN_IF_OFFSET and SINGLE_PIN_IF_SCALE can be used to scale to the value the desired range. In case of the PWM input, a permanent low input signal or permanent high signal is treated as input error and the chosen target value is set to zero.
Register SINGLE_PIN_IF_CFG configures the length of a digital filter for the PWM_I signal. Spikes on the signal can be thereby suppressed. Bit 0 in register SINGLE_PIN_IF_STATUS is set high when PWM_I is constant low, bit 1 is set high when the PWM_I is constant high. Writing to this register resets these flags. Maximum PWM period of the PWM signal must be 65000 x 40 ns. The calculation of the normalized duty cycle is started on the rising edge of PWM_I. The PWM frequency must be constant as big variations (tolerance of 4 µs in PWM period) in the PWM frequency are treated as error.
A duty cycle of 50% equals an input value of 32768. With the offset and scaling factors, it can be mapped to the desired range.

4.2.6 GPIO Interface

The TMC4671 has eight GPIO-pins arranged in group A (GPIO 0 to 3) and group B (GPIO 4 to 7). These pins can be configured using bits 0 to 6 of the register GPIO_dsADCI_CONFIG (0x7B). The configurations include RTMI, GPI, or GPO as well as clock signals, in and out, for external delta sigma modulators. Groups A and B can individually be configured as in or outputs. Single pins within these groups cannot be individually configured. Bits 16 to 19 set the GPO values for group A and bits 20 to 23 set the GPO values for group B. If configured as GPIs, bits 24 to 27 display the input on group A whereas bits 28 to 31 display the input on the group B GPIs.

GPIO_dsADCI_CONFIG (Bits 6 to 0)	Configured As	Group A	Group B
$xxxxxx0_b$	RTMI	0: Z 1: Z 2: Z 3: /CS (out)	4: SCK (out) 5: MOSI (out) 6: MISO (in) 7: TRG (in)
$xx11001_b$	GPIO	GPO	GPO
$xx00001_b$	GPIO	GPI	GPI
$xx01001_b$	GPIO	GPO	GPI
$xx10001_b$	GPIO	GPI	GPO
$11xx111_b$	Delta Sigma ADC	MCLK_out	MCLK_out
		0: ADCI0	4: ADCAGPI_B

GPIO_dsADCI_CONFIG (Bits 6 to 0)	Configured As	Group A	Group B
		1: ADCI1	5: AENC_UX
		2: ADCVM	6: AENC_VN
		3: ADCAGPI_A	7: AENC_WY
$00xx111_b$	Delta Sigma ADC	MCLK_in	MCLK_in
		0: ADCI0	4: ADCAGPI_B
		1: ADCI1	5: AENC_UX
		2: ADCVM	6: AENC_VN
		3: ADCAGPI_A	7: AENC_WY

Table 5: GPIO Configuration Overview with 'x' as Don't Care

NOTE

When the RTMI-option is selected, it is not possible to use the GPIOs and the other way around. On default, the RTMI-Mode is chosen and the unused GPIOs 0,1, and 2 are configured as inputs on high impedance Z.
In addition, the only possible Delta Sigma ADC configurations are the ones listed inTable 5.

4.3 Numerical Representation, Electrical Angle, Mechanical Angle, and Pole Pairs

The TMC4671 uses different numerical representations for different parameters, measured values, and interim results. The terms electrical angle PHI_E, mechanical angle PHI_M, and number of pole pairs (N_POLE_PAIRS) of the motor are important for the setup of FOC. This section describes the different numerical representations of parameters and terms.

4.3.1 Numerical Representation

The TMC4671 uses signed and unsigned values of different lengths and fixed point representations for parameters that require a non-integer granularity.

Symbol	Description	Min	Max
u16	Unsigned 16-bit value	0	65535
s16	Signed 16-bit values, 2'th complement	-32767	32767
u32	Unsigned 32-bit value	0	2^{32} = 4294967296
s32	Signed 32-bit values, 2'th complement	-2147483647	2^{31} - 1 = 2147483647
q8.8	Signed fix point value with 8-bit integer part and 8-bit fractional part	-32767/256	32767/256
q4.12	Signed fix point value with 4-bit integer part and 12-bit fractional part	-32767/4096	32767/4096

Table 6: Numerical Representations

NOTE Two's complement of n bit is $-2^{(n-1)} \ldots -2^{(n-1)} - 1$. To avoid unwanted overflow, the range is clipped to $-2^{(n-1)} + 1 \ldots -2^{(n-1)} - 1$.

Because the zero is interpreted as a positive number for 2'th complement representation of integer n bit number, the smallest negative number is $-2^{(n-1)}$, where the largest positive number is $2^{(n-1)} - 1$. Using the smallest negative number $-2^{(n-1)}$ might cause critical underflow or overflow. Internal clipping takes this into account by mapping $-2^{(n-1)}$ to $-2^{(n-1)} + 1$.

Hexadecimal Value	u16	s16	q8.8	q4.12
$0x0000_h$	0	0	0.0	0.0
$0x0001_h$	1	1	1/256	1/4096
$0x0002_h$	2	2	2/256	2/4096
$0x0080_h$	128	128	0.5	0.03125
$0x0100_h$	256	256	1.0	0.0625
$0x0200_h$	512	512	2.0	0.125
$0x3FFF_h$	16383	16383	16383/256	16383/4096
$0x5A81_h$	23169	23169	23169/256	23169/4096
$0x7FFF_h$	32767	32767	32767/256	32767/4096
$0x8000_h$	32768	-32768	-32768/256	-32768/4096
$0x8001_h$	32769	-32767	-32767/256	-32767/4096
$0x8002_h$	32770	-32766	-32766/256	-32766/4096
$0xC001_h$	49153	-16383	-16383/256	-16383/4096
$0xFFFE_h$	65534	-2	-2/256	-2/4096
$0xFFFF_h$	65535	-1	-1/256	-1/4096

Table 7: Examples of u16, s16, q8.8, q4.12

The q8.8 and q4.12 are used for P and I parameters, which are positive numbers. Note that q8.8 and q4.12 are used as signed numbers. This is because theses values are multiplied with signed error values resp. error integral values.

4.3.2 N_POLE_PAIRS, PHI_E, PHI_M

The parameter N_POLE_PAIRS defines the factor between electrical angle PHI_E and mechanical angle PHI_M of a motor (see Figure 14).
A motor with one (1) pole pair turns once for each electrical period. A motor with two (2) pole pairs turns once for every two electrical periods. A motor with three (3) pole pairs turns once for every three electrical periods. A motor with four pole (4) pairs turns once for every four electrical periods.

The electrical angle PHI_E is relevant for the commutation of the motor. It is relevant for the torque control of the inner FOC loop.

$$\text{PHI\_E} = \text{PHI\_M} \cdot \text{N\_POLE\_PAIRS} \tag{3}$$

The mechanical angle PHI_M is primarily relevant for velocity control and for positioning. This is to control the motor speed in terms of mechanical turns and not in terms of electrical turns.

$$\text{PHI\_M} = \text{PHI\_E}/\text{N\_POLE\_PAIRS} \tag{4}$$

Different encoders give different kinds of position angles. Digital hall sensors normally give the electrical position PHI_E, which can be used for commutation. Analog encoders give (depending on their resolution) angles that must be scaled first to mechanical angles PHI_M and to electrical angles PHI_E for commutation.

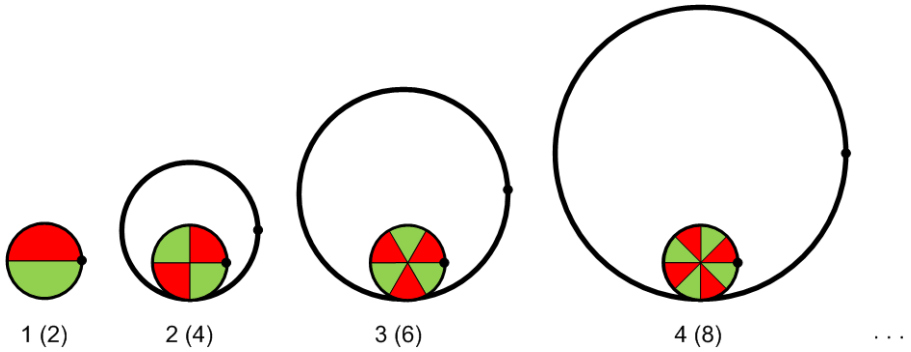


Figure 14: N_POLE_PAIRS - Number of Pole Pairs (Number of Poles)

4.3.3 Numerical Representation of Angles PHI

Electrical angles and mechanical angles are represented as 16-bit integer values. One full revolution of $360\,\mathrm{deg}$ is equivalent to $2^{16} = 65536$ steps. Any position coming from a sensor is mapped to this integer range. Adding an offset of PHI_OFFSET causes a rotation of an angle PHI_OFFSET$/2^{16}$. Subtraction of an offset causes a rotation of an angle PHI_OFFSET in opposite direction.

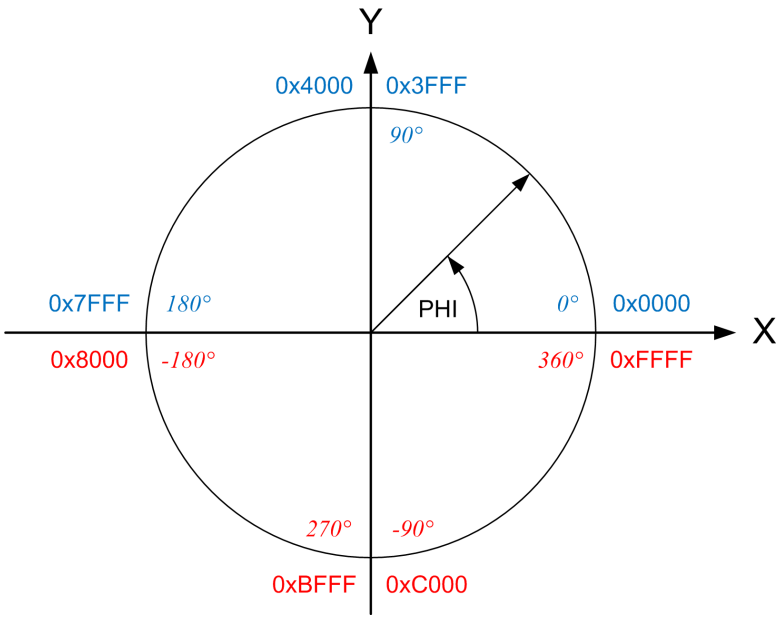


Figure 15: Integer Representation of Angles as 16-Bit Signed (s16) Resp. 16-Bit Unsigned (u16)

Hexadecimal Value	u16	s16	PHI[°]$_{u16}$	PHI[°]$_{s16}$
0x0000$_h$	0	0	0.0	0.0
0x1555$_h$	5461	5461	30.0	30.0
0x2AAA$_h$	10922	10922	60.0	60.0
0x4000$_h$	16384	16384	90.0	90.0
0x5555$_h$	21845	21845	120.0	120.0
0x6AAA$_h$	27306	27768	150.0	150.0
0x8000$_h$	32768	-32768	180.0	-180.0
0x9555$_h$	38229	-27307	210.0	-150.0
0xAAAA$_h$	43690	-21846	240.0	-120.0
0xC000$_h$	49152	-16384	270.0	-90.0
0xD555$_h$	54613	-10923	300.0	-60.0
0xEAAA$_h$	60074	-5462	330.0	-30.0

Table 8: Examples of u16, s16

The option of adding an offset is to adjust the angle shift between the motor and stator and the rotor and encoder. Finally, the relative orientations between the motor and stator and the rotor and encoder can be adjusted by just one offset. Alternatively, it is possible to set the counter position of an incremental encoder to zero on initial position. For absolute encoders, use the offset to set an initial position.

4.4 ADC Engine

The ADC engine controls the sampling, selection, scaling, and offset correction of different available ADC channels. Two ADC channels are for phase current measurement, three ADC channels are for analog hall signals or for analog sin-cos-encoder, one ADC channel is for optional measurement of the motor supply voltage, and two additional ADC channels are general purpose, where one general purpose analog input can be used as analog target value by the single pin interface.

4.4.1 ADC Current Sensing Channels ADC_I1 and ADC_I0

The ADC channels (ADC_I0_POS, ADC_I0_NEG, ADC_I1_POS, ADC_I1_NEG) are for current sensing in differential input configuration. In differential configuration, the ADC_I0_POS and ADC_I1_POS are the inputs for the sense amplifier output signals, where ADC_I1_NEG and ADC_I0_NEG are for the zero current sensing reference of the sense amplifiers. In single-ended configuration, the ADC_I0_POS and ADC_I1_POS are the inputs for the sense amplifier output signals, where ADC_I1_NEG and ADC_I0_NEG are internally connected to ground. The third current channel ADC_I2, as required for three-phase FOC, is calculated using Kirchhoff's law ADC_I2 = - (ADC_I1 + ADC_I0).

NOTE

ADC_I0_POS, ADC_I0_NEG, ADC_I1_POS, ADC_I1_NEG are low voltage analog inputs and must not be directly connected to inline sense resistors. The TMC4671 requires external dfferential motor supply common-mode range current-sensing amplifiers for inline current sensing.

4.4.2 ADC for Analog Hall Signals or Analog Sin-Cos-Encoders AENC_UX, AENC_VN, AENC_WY

For analog hall and analog encoder, the ADC engine has three differential input channels ((AENC_UX_POS, AENC_UX_NEG), (AENC_VN_POS, AENC_VN_NEG), and (AENC_WY_POS, AENC_WY_NEG)). The analog encoder ADC inputs can be configured single ended (AENC_UX_POS, AENC_VN_POS, AENC_WY_POS) with negative inputs (AENC_UX_NEG, AENC_VN_NEG, AENC_WY_NEG) internally connected to ground.

The three channels AENC_UX, AENC_VN, AENC_WY are for three-phase analog sine (with +/-120° phase shift) wave hall signals. The signals AENC_UX and AENC_WY are for two-phase analog sine wave and cosin wave hall signals. The signals AENC_UX and AENC_WY are for analog sin-cos-encoder. The AENC_VN is for an optional zero pulse channel of sin-cos-encoders. The AENC_VN is available for read out by the application software but it is not hardware handled by the TMC4671 for position zeroing.

For long analog signal lines, it might be necessary to use external differential receivers with twisted pair line termination resistors to drive the single-ended analog encoder inputs of the TMC4671.

4.4.3 ADC Supply Voltage Measurement ADC_VM

The ADC channel for measures of supply voltage (ADC_VM) and is associated with the brake chopper. The ADC_VM is available as raw value only without digital scaling because it is not directly processed by the FOC engine.

NOTE

ADC_VM must be scaled down electrically by the voltage divider to the allowed voltage range, and might require additional supply voltage spike protection.

4.4.4 ADC_VM for Brake Choppper

The ADC_VM is available as input for the optional brake chopper as raw value u16. The brake chopper thresholds must be set as absolute u16 values to activate and deactivate the brake chopper output depending on the ADC_VM value.

4.4.5 ADC EXT Register Option

It is possible to write ADC values into the ADC_EXT registers of the register bank from external sources or for evaluation purposes. These values can be selected as raw current ADC values. ADC_EXT registers are primarily intended for test purposes as optional inputs for external current measurement sources.

4.4.6 ADC General Purpose Analog Inputs AGPI_A and AGPI_B

Two general purpose ADC channels are single-ended analog inputs (AGPI_A, AGPI_B). The general purpose analog ADC inputs AGPI_A and AGPI_B are available as raw values only without digital scaling because these values are not directly processed by the FOC engine. These general purpose analog inputs (AGPI) are intended to monitor analog voltage signals representing MOSFET temperature or motor temperature. They are two additional ADC channels. Optionally, the AGPI_A is availabe as an analog target value signal.

4.4.7 ADC RAW Values

The sampled raw ADC values are available for read out in register 0x02 while the depicted ADC inputs are selected through register 0x03. This is important during the system setup phase to determine offset and scaling factors.

4.4.8 ADC_SCALE and ADC_OFFSET

The FOC engine expects offset corrected ADC current values scaled to the used 16-bit (s16) fixed point representation. The integrated scaler and offset compensator maps raw ADC samples of current measurement channels to 16-bit two's complement values (s16). While the offset is compensated by subtraction, it is represented as an unsigned value. The scaling value is signed to compensate wrong measurement direction. The s16 scaled ADC values are available for read out from the register (ADC_I1, ADC_I0) resp. (AENC_UX, AENC_VN, AENC_WY).

NOTE	Wrong scaling factors (ADC_SCALE) or wrong offsets (ADC_OFFSET) might cause damages when the FOC is active. Integrated hardware limiters allow protection - especially in the setup phase when using careful limits.

4.4.9 ADC Gain Factors for Real World Values

Each ADC channel of the TMC4671 has an individual gain factor determined by its associated chain of gain factors and by digital scaling factors, if available, for an ADC channel. ADC register values are either 16-bit unsigned values (u16) or 16-bit signed values (s16). With gain factors, calculate ADC values as real world values if required.

Gain factors IgainADC for ADC current values are typical in units [A/LSB] or [mA/LSB]. Gain factors UgainADC for ADC voltage values are typical in units [V/LSB] or [mV/LSB].

$$\text{ADCmeasuredCurrent[A]} = \text{IgainADC[A/LSB]} \times \text{ADC\_CURRENT\_S16} \quad (5)$$

$$\text{ADCmeasuredVoltage[V]} = \text{UgainADC[V/LSB]} \times \text{ADC\_VOLTAGE\_S16} \quad (6)$$

$$\text{ADCmeasuredVoltage[V]} = \text{UgainADC[V/LSB]} \times \text{ADC\_VOLTAGE\_U16} \quad (7)$$

4.4.10 Internal Delta Sigma ADCs

The TMC4671 is equipped with internal delta sigma ADCs for current measurement, supply voltage measurement, analog GPIs and analog encoder signal measurement. Delta sigma ADCs, as integrated within the TMC4671, together with programmable digital filters, are flexible in parameterizing concerning resolution vs. speed. The advantage of delta sigma ADCs is that the user can adjust measurement from lower speed with higher resolution to higher speed with lower resolution. This fits with motor control application. Higher resolution is required for low speed signals, while lower resolution satisfies the needs for high speed signals.

Due to high oversampling, the analog input front-end is easier to implement than for successive approximation register ADCs as anti-aliasing filters can be chosen to a much higher cutoff frequency. The ADC engine processes all ADC channels in parallel hardware - avoiding phase shifts between the channels compared to ADC channels integrated in MCUs.

4.4.11 Internal Delta Sigma ADC Input Stage Configuration

ADC channels can be configured either as differential-ended analog inputs (ADC_I0, ADC_I1, AENC_UX, AENC_VN, AENC_WY) or as single-ended analog inputs (ADC_VM, AGPI_A, AGPI_B). Additionally, the ADC all channels can be set to fixed voltages (0V, $V_{REF}/4$, $V_{REF}/2$, $3 \times V_{REF}/4$) for calibrations.

STAGE_CFG(n+2:n)	CONFIGURATION	DESCRIPTION	COMMENT
0	INP vs. INN	Differential mode	Default configuration
1	GND vs. INN	Single-ended negative INN vs. GND	(For test purposes only)
2	$V_{DD}/4$	25% ADC reference voltage	Calibration aid
3	$3 \times V_{DD}/4$	75% ADC reference voltage	Calibration aid
4	INP vs. GND	Single-ended mode INP vs. GND	(Half voltage range, offset)
5	$V_{DD}/2$	50% ADC reference voltage	Calibration aid
6	$V_{DD}/4$	25% ADC reference voltage	(Redundant configuration)
7	$3 \times V_{DD}/4$	75% ADC reference voltage	(Redundant configuration)

Table 9: Delta Sigma ($\Delta\Sigma$) ADC Input Stage Configurations

The three bit vector ADC_STAGES_CFG(n+2:n) is part of the DS_ANALOG_INPUT_STAGE_CFG(31:0) with n = 0, 4, 8, 12, 16, 20, 24, 28 for the eigth delta sigma ADC channels. DS_ANALOG_INPUT_STAGE_CFG configures the associated delta sigma ADC input stages according toTable 17. For association of the bit position (bit n+2 to bit n), see register bank section 7.2.

STAGE_CFG(n+2:n)	ADC channel	function
STAGE_CFG(2:0)	ADC_I0	Sense voltage of current I0
STAGE_CFG(6:4)	ADC_I1	Sense voltage of current I1
STAGE_CFG(9:8)	ADC_VM	Down divided supply voltage
STAGE_CFG(10)	'1'	Fixed for ADC_VM (STAGE_CFG=4,5,6,7)
STAGE_CFG(13:12)	ADC_AGPI_A	General purpose analog input A
STAGE_CFG(14)	'1'	Fixed for ADC_AGPI_A (STAGE_CFG=4,5,6,7)
STAGE_CFG(17:16)	ADC_AGPI_B	General purpose analog input B
STAGE_CFG(18)	'1'	Fixed for ADC_AGPI_B (STAGE_CFG=4,5,6,7)
STAGE_CFG(22:20)	ADC_AENC_UX	Analog hall UX/analog encoder COS
STAGE_CFG(26:24)	ADC_AENC_VN	Analog hall V/analog encoder N
STAGE_CFG(30:28)	ADC_AENC_WY	Analog hall WY/analog encoder SIN

Table 10: Delta Sigma (ΔΣ) ADC Input Stage Configurations

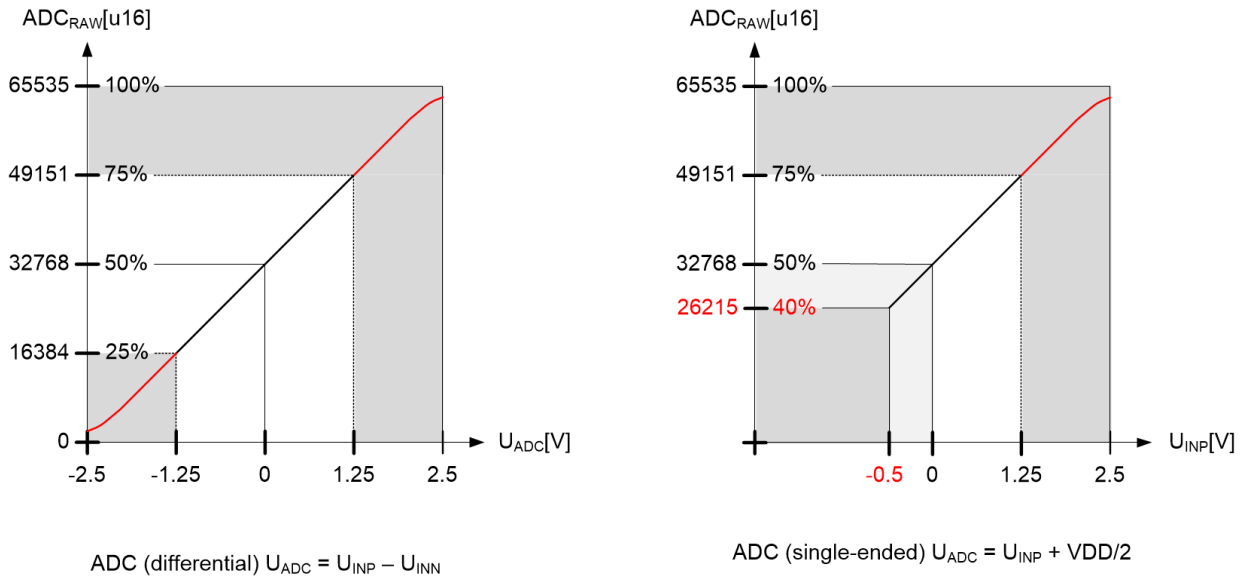


Figure 16: Input Voltage Ranges of Internal Delta Sigma ADC Channels

Figure 16 illustrates the typical relation between input voltage and corresponding raw ADC output. For differential operation, the input range between 25% and 75% corresponds to voltage values between 1.25V to 3.75V. This is the recommended operation area of the ADC. Below 25% and above 75%, the ADC shows significant nonlinearity due to the Delta Sigma measurement principle.

In single-ended operation, the recommended input range starts at 0V and ends at 1.25V. Measurement below GND might be distorted and is not recommended.

NOTE Due to manufacturing tolerances, calibration of offset and amplitude is always recommended. Also consider stability of the reference voltage.

4.4.12 External Delta Sigma ADCs

The delta sigma front-end of the ADC engine supports external delta sigma modulators to enable isolated delta sigma modulators for the TMC4671. Additionally, the delta sigma front-end supports low-cost comparators together with two resistors and one capacitor (R-C-R-CMP) forming first order delta sigma modulators, as generic analog front-end for pure digital variants of the TMC4671 core.

4.4.13 ADC Group A and ADC Group B

ADC channels of the TMC4671 are grouped into two groups, to enable different sample rates for two groups of analog signals, if needed. Running both ADC groups with the same sampling frequency is recommended for almost all applications. It might be necessary to run the ADC channels of analog encoder with a much higher frequency than the ADC channels for current measurement if using a high resolution analog encoder.

4.4.14 Delta Sigma Configuration and Timing Configuration

The delta sigma configuration is programmed through the MCFG register that selects the mode (internal/external delta sigma modulator with programmable MCLK; delta sigma modulator clock mode (MCLK output, MCLK input, MCLK used as MDAC output with external R-C-R-CMP configuration); delta sigma modulator clock and its polarity; and the polarity of the delta sigma modulator data signal MDAT).

NOTE The power-on delta sigma configuration should fit with most applications when using the integrated delta sigma ADCs of the TMC4671. Primarily, the default delta sigma configuration must be adapted when using external delta sigma modulators or selecting differential ADC input configurations, or in case of enhanced sampling requirements for high resolution analog encoders.

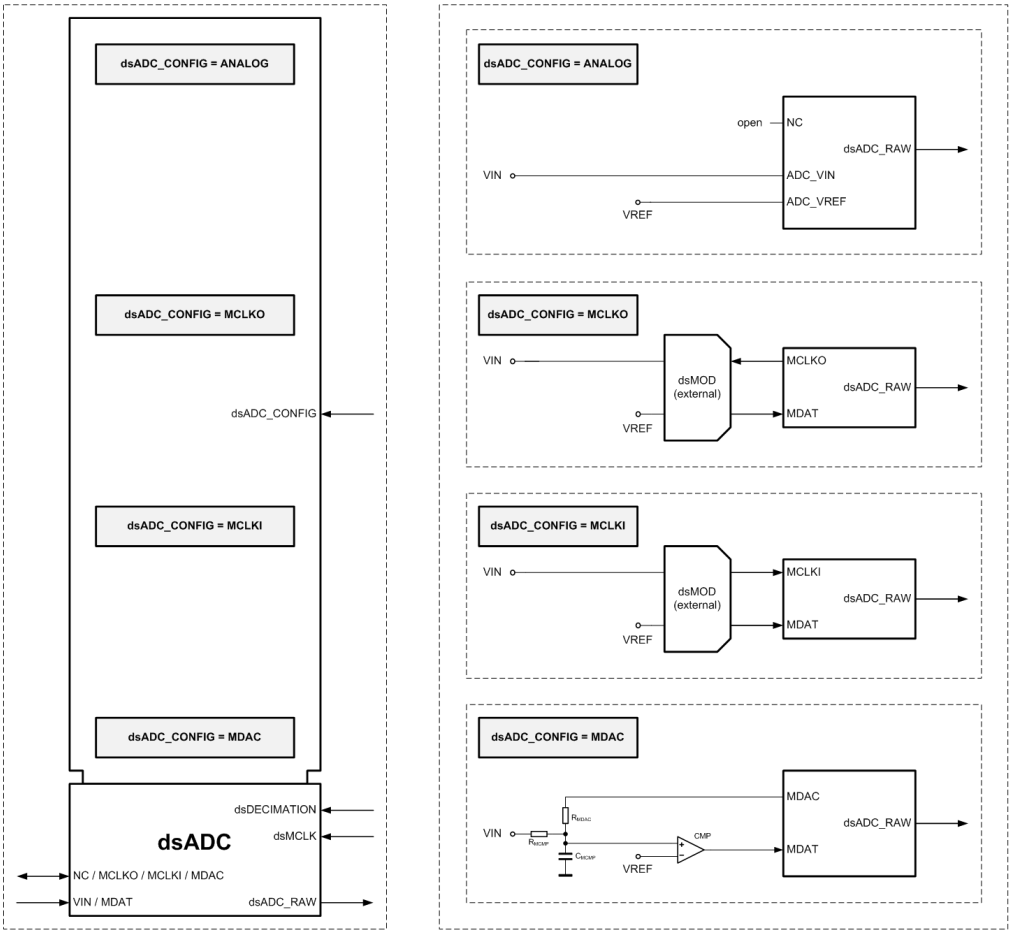


Figure 17: Delta Sigma ADC Configurations dsADC_CONFIG (Internal: ANALOG vs. External: MCLKO, MCLKI, MDAC)

dsADC_CONFIG	Description	NC_MCLKO_MCLKI_MDAC	VIN_MDAT
ANALOG	Integrated internal ADC mode, VIN_MDAT is analog input V_{IN}	MCLK not connected (NC)	V_{IN} (analog)
MCLKO	External dsModulator (example, AD7403) with MCLK input driven by MCLKO	MCLK output	MDAT input
MCLKI	External dsModulator (example, AD7400) with MCLK output that drives MCLKI	MCLK input	MDAT input
MDAC	External dsModulator (example, LM339) realized by external comparator CMP with two R and one C	MDAC output (= MCLK out)	MDAT input for CMP

Table 11: Delta Sigma ADC Configurations (Figure 17), Selected with dsADC_MCFG_A and dsADC_MCFG_B.

Register	Function
dsADC_MCFG_B	Delta sigma modulator configuration MCFG (ANALOG, MCLKI, MCLKO, MDAC), group B
dsADC_MCFG_A	Delta sigma modulator configuration MCFG (ANALOG, MCLKI, MCLKO, MDAC), group A
dsADC_MCLK_B	Delta sigma modulator clock MCLK, group B
dsADC_MCLK_A	Delta sigma modulator clock MCLK, group A
dsADC_MDEC_B	Delta sigma decimation parameter MDEC, group B
dsADC_MDEC_A	Delta sigma decimation parameter MDEC, group A

Table 12: Registers for Delta Sigma Configuration

4.4.14.1 Timing Configuration MCLK

When the programmable MCLK is selected, the MCLK_A and MCLK_B parameter registers define the programmable clock frequency fMCLK of the delta sigma modulator clock signal MCLK for delta sigma modulator group A and group B. For a given target delta sigma modulator frequency fMCLK, together with the internal clock frequency fCLK = 100MHz, the MCLK frequency parameter is calculated by:

$$\text{MCLK} = 2^{31} \times \text{fMCLK[Hz]}/\text{fCLK[Hz]} \tag{8}$$

Due to the 32 bit's length of the MCLK frequency parameter, the resulting frequency fMCLK might differ from the desired frequency fMCLK. The back calculation of the resulting frequency fMCLK for a calculated MCLK parameter with 32 bit length is defined by:

$$\text{fMCLK[Hz]} = \text{fCLK[Hz]} \times \text{MCLK}/2^{31} \tag{9}$$

The precise programming of the MCLK frequency is primarily intended for external delta sigma modulators to meet given EMI requirements. With this, it is possible to program frequencies fMCLK with a resolution better than 0.1 Hz. This advantage concerning EMI might cause trouble when using external delta sigma modulators if they are sensitive to slight frequency alternating. This is not an issue when using external first-order delta sigma modulators based on R-C-R-CMP (example, LM339). But for external second-order delta sigma modulators, it is recommended to configure the MCLK parameter for frequencies fMCLK with kHz quantization (example, 10,001,000 Hz instead of 10,000,001 Hz).

fMCLK_target	MCLK	fMCLK_resulting	Comment
25MHz	0x20000000	25MHz	W/o fMCLK frequency jitter, recommended
20MHz	0x19000000	20MHz -468750Hz	Recommended for ext. $\Delta\Sigma$ modulator
20MHz	0x19999999	20MHz -0.03Hz	Might be critical for ext. $\Delta\Sigma$ modulator
12.5MHz	0x10000000	12.5MHz	W/o fMCLK frequency jitter, recommended
10MHz	0x0CCCCCCC	10MHz -0.04Hz	Might be critical for ext. $\Delta\Sigma$ modulator
10MHz	0x0CC00000	10MHz -39062.5Hz	Recommended for ext. $\Delta\Sigma$ modulator

Table 13: Delta Sigma MCLK Configurations

4.4.14.2 Decimation Parameter MDEC

The high oversampled single bit delta sigma data stream (MDAT) is digitally filtered by Sinc3 filters. To get raw ADC data, the actual digitally filtered values must be sampled periodically with a lower rate called decimation ratio. The decimation is controlled by parameter MDEC_A for ADC group A and MDEC_B for ADC group B. A new ADC_RAW value is available after MDEC delta sigma pulses of MCLK. As such, the parameters MCLK and MDEC together define the sampling rate of the 16-bit ADC_RAW values.

The delta sigma modulator with Sinc3 filter works with best noise reduction performance when the length of the step response time tSINC3 of the Sinc3 filter is equal to the length of the PWM period tPWM = (PWM_MAXCNT+1) / fPWMCLK = ((PWM_MAXCNT+1) $\times$ 10 ns) of the period. The length of the step function response of a Sinc3 filter is:

$$\text{tSINC3} = (3 \times (\text{MDEC} - 1) + 1) \times \text{tMCLK} \tag{10}$$

$$\text{MDEC}_{\text{recommended}} = \frac{\text{tPWM}}{3 \times \text{tMCLK}} - 2 \tag{11}$$

fMCLK	tMCLK	MDEC25 (25 kHz, 40μs)	MDEC50 (50 kHz, 20μs)	MDEC100 (100 kHz, 10μs)
50MHz	20ns	665	331	165
25MHz	40ns	331	165	81
20MHz	50ns	265	131	65
12.5MHz	80ns	165	81	40
10Hz	100ns	131	65	31

Table 14: Optimal Decimation Parameter MDEC (According to Equation (11) for Different PWM Frequencies fPWM (MDEC25 for fPWM=25kHz w/ PWM_MAXCNT=3999, MDEC50 for fPWM=50kHz w/ PWM_MAXCNT=1999, MDEC100 for fPWM=100kHz w/ PWM_MAXCNT=999).

NOTE

MDEC parameter can be changed during operation.
This enables adaptive adjustment of performance with respect to resolution versus speed on demand.

For most applications, the power-on decimation setting of MDEC should be sufficient.

4.4.15 Internal Delta Sigma Modulators - Mapping of V_RAW to ADC_RAW

Generally, delta sigma modulators work best for a typical input voltage range of 25% V_MAX ... 75% V_MAX (unsigned 0% ... 100%). For the integrated delta sigma modulators, this input voltage operation range is V_MAX = 5V.
The Table 15 defines the voltage ranges for 5V analog supply voltages.

V_SUPPLY[V]	(V_MIN[V])	V_MIN25%[V]	V_MAX50%[V]	V_MAX75%[V]	(V_MAX[V])
5.0	(0.0)	1.250	2.50	3.75	(5.0)

Table 15: Recommended Input Voltage Range from V_MIN25%[V] to V_MAX75%[V] for Internal Delta Sigma Modulators; V_SUPPLY[V] = 5V is Recommended for the Analog Part of the TMC4671.

$$\text{V\_RAW} = \begin{cases} \text{V\_MAX} & \text{for} \quad \text{V}_{IN} > \text{V\_MAX} \\ (\text{V}_{IN} - \text{V\_REF}) & \text{for} \quad \text{V\_MIN} < (\text{V}_{IN} - \text{V\_REF}) < \text{V\_MAX} \\ \text{V\_MIN} & \text{for} \quad \text{V}_{IN} < \text{V\_MIN} \end{cases} \tag{12}$$

The resulting raw ADC value is:

$$\text{ADC\_RAW} = (2^{16} - 1) \times \frac{\text{V\_RAW}}{\text{V\_MAX}} \quad \text{for} \quad \text{V\_MIN25\%[V]} < \text{V\_RAW} < \text{V\_MAX75\%[V]}. \tag{13}$$

The idealized expression (equation 12) is valid for recommended voltage ranges (Table 15) neglecting deviations in linearities. These deviations primarily depend on different impedance on the analog signal path, but also on digital parameterization. Finally, the deviation is quantified in terms of resulting ADC resolution. So, the Delta Sigma ADC engine maps the analog input voltages V_RAW = V_{IN} - V_REF of voltage range V_MIN < V_RAW < V_MAX to ADC_RAW values of range $\{0 \ldots (2^{16}) - 1\}$ <=> $\{0 \ldots 65535\}$ <=> 0x0000 ... 0xFFFF.

V_{MIN}[V]	V_{REF}[V]	V_{MAX}[V]	V_{IN}[V]	DUTY[%]	ADC_RAW
0.0	2.5	5.0	(0.0)	(0%)	(0x0000)
0.0	2.5	5.0	1.0	25%	0x4000
0.0	2.5	5.0	2.5	50%	0x7fff
0.0	2.5	5.0	3.75	75%	0xC000
0.0	2.5	5.0	(5.0)	(100%)	(0xffff)

Table 16: Delta Sigma Input Voltage Mapping of Internal Delta Sigma Modulators

NOTE For calibrating purposes, the input voltage of the delta sigma ADC inputs can be programmed to fixed voltages (25%, 50%, 75% of analog supply voltage) through the associated configuration register DS_ANALOG_INPUT_STAGE_CFG.

4.4.16 External Delta Sigma Modulator Interface

The TMC4671 is equipped with integrated digital filters for extraction of ADC raw values from delta sigma data stream for both internal and external delta sigma modulators. The interface for external delta sigma modulators is intended for external isolated sigma delta modulators, such as AD7401 (with MCLK input driven by TMC4671), or AD7402 (with MCLK output to drive TMC4671). In addition, the external delta sigma interface supports the use of simple comparator with a R-C-R network as external low cost delta sigma modulators (R-C-R-CMP, example, LM339).

NOTE When selecting the external delta sigma ADC Interface, the high-performance debug SPI (RTMI) it not available in parallel due to pin sharing. The UART is always available, but with less performance than the RTMI.

Each external delta sigma modulator channel (dsMOD) has two signals (see Figure 17), one dedicated input, and one programmable input/output. The configuration of the external delta sigma modulator interface is defined by programming associated registers. When selecting external delta sigma ADC, the associated analog ADC inputs are configured as digital inputs for the delta sigma signal data stream MDAT.

4.4.16.1 External Delta Sigma Modulator Interface - MDAC Configuration

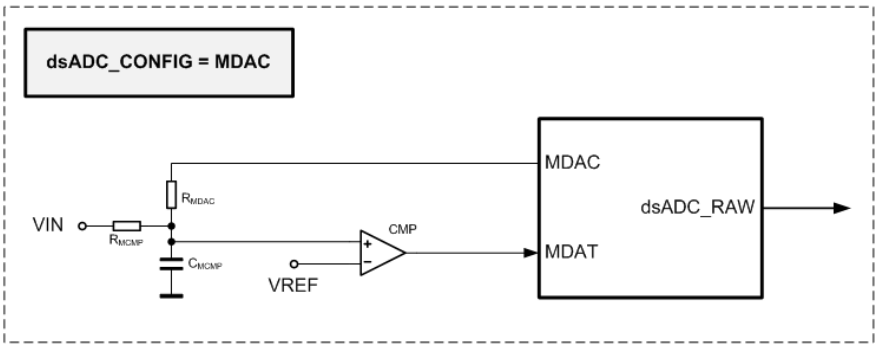


Figure 18: $\Delta\Sigma$ ADC Configuration - MDAC (Comparator-R-C-R as $\Delta\Sigma$-Modulator)

In the MDAC delta sigma modulator, the delay of the comparator CMP determines the MCLK of the comparator modulator. A capacitor C_{MCCMP} within a range of 100 pF ...1nF fits in most cases. The time constant τRC should be in a range of 0.1 tCMP ...tCMP of the comparator. The resistors should be in the range of 1K to 10K. The fMAXtyp depends also on the choice of the decimation ratio.

CMP	tCMPtyp [ns]	R_{MCMP} [kΩ]	R_{MDAC} [kΩ]	C_{MCMP} [pF]	fMCLKmaxTYP
LM339	1000	1	1	100	1 MHz
LM339	1000	10	10	100	100 kHz
LM339	1000	100	100	100	10 kHz

Table 17: Delta Sigma R-C-R-CMP Configurations (See Figure 17)

For external Delta Sigma R-C-R-CMP modulators, get the Delta Sigma input voltage mapping according toTable 18. The support of low-cost external comparators used as first order delta sigmal modulators is intended as a generic analog interface option for the compatibility of the TMC4671 core in case it is embedded within a pure digital technology environment.

V_{MIN}[V]	V_{REF}[V]	V_{MAX}[V]	V_{IN}[V]	DUTY[%]	ADC_RAW
0.0	1.65	3.3	0.0	0%	0x0000
0.0	1.65	3.3	0.825	25%	0x4000
0.0	1.65	3.3	1.65	50%	0x7fff
0.0	1.65	3.3	2.475	75%	0xC000
0.0	1.65	3.3	3.3	100%	0xffff
V_{MIN}[V]	**V_{REF}[V]**	**V_{MAX}[V]**	**V_{IN}[V]**	**DUTY[%]**	**ADC_RAW**
0.0	2.5	5.0	0.0	0%	0x0000
0.0	2.5	5.0	1.0	25%	0x4000
0.0	2.5	5.0	2.5	50%	0x7fff
0.0	2.5	5.0	3.75	75%	0xC000
0.0	2.5	5.0	5.0	100%	0xffff

Table 18: Delta Sigma Input Voltage Mapping of External Comparator (CMP)

4.5 Analog Signal Conditioning

The range of measured coil currents, resp. the measured voltages of sense resistors, must be mapped to the valid input voltage range of the delta sigma ADC inputs. This analog preprocessing is the task of the analog signal conditioning.

4.5.1 Chain of Gains

The current flowing through the sense resistor R_{Sense} generates a voltage U_{RSense} amplified by a sense amplifier with the gain $G_{SenseAmp}$, as depicted inFigure 19. The internal ADCs have a voltage range $U_{ADCRange}$ of +/-2.5 V. To calculate the A_{Peak}/LSB relation, use the following equation:

$$A_{Peak}/LSB = U_{ADCRange}/(R_{Sense} \times G_{SenseAmp} \times 32767) \quad (14)$$

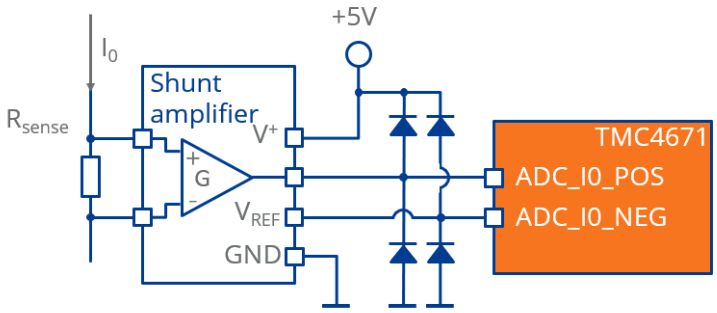


Figure 19: Principle of Shunt and Shunt Amplifier

To externally calculate the real measured currents, read the raw current values for ADC_I0_RAW and ADC_I1_RAW from register 0x02 ADC_RAW_DATA and multiply them with factor A_{Peak}/LSB.

To compensate, possible DC offsets of the chain of gains ADC_I1_OFFSET in register 0x08 and ADC_I0_OFFSET in register 0x09 can be used for each ADC channel, respectively. Furthermore, the raw ADC values can be scaled using ADC_I1_SCALE and ADC_I0_SCALE in the same registers. Using the default value of 256 for scaling is adviced and does not change the A_{Peak}/LSB factor, making it easy to calculate target currents, limits, and measured currents for the FOC-loop.

NOTE Due to the nonlinearity of the internal deltaSigma ADCs of the TMC4671, keep the actual voltage within [-1.25V; 1.25V] range for differential and [0; 1.25V] for single-ended mode. (see Figure 16)

4.5.2 Chain of Gains Example

The following equation is an example calculation using the TMC6100-EVAL-KIT (uses AD8418 senseamp) and the internal ADCs of the TMC4671.Table 19 lists the parameters utilized for the TMC6100-EVAL:

R_{Sense} [Ω]	$G_{SenseAmp}$ [V/V]	$U_{ADCRange}$ [V]
0.003	20	2.5

Table 19: Example ADC/LSB for TMC6100

$$A_{Peak}/LSB = 2.5\ V/(0.003\Omega \times 20\ V/V \times 32767) = 1.272\ mA/LSB \quad (15)$$

Table 20 depicts the conversion factor for Trinamic-EVAL-kits compatible with the TMC4671:

EVAL	[mA/LSB]
TMC-UPS-10/70-EVAL	0.509
TMC-UPS-2/24-EVAL	0.153
TMC6100-EVAL	1.272
TMC4671+TMC6100-BOB	1.272

Table 20: List of A/LSB-Factors for TMC-EVALs

4.5.3 FOC3 - Stator Coil Currents I_U, I_V, I_W and Associated Voltages U_U, U_V, U_W

The correct association between stator terminal voltages U_U, U_V, U_W and stator coil currents I_U, I_V, I_W is essential for the FOC.

For three-phase motors with three terminals U, V, W, the voltage U_U is in phase with the current I_U, U_V is in phase with I_V, and U_W is in phase with I_W according to equations (16) and (17) for FOC3.

$$\text{U\_UVW\_FOC3(U\_D, PHI\_E)} = \begin{cases} U_U(\phi_e) = U_D \times sin(\phi_e) \\ U_V(\phi_e) = U_D \times sin(\phi_e + 120^o) \\ U_W(\phi_e) = U_D \times sin(\phi_e - 120^o) \end{cases} \quad (16)$$

$$\text{I\_UVW\_FOC3(I\_D, PHI\_E)} = \begin{cases} I_U(\phi_e) = I_D \times sin(\phi_e) \\ I_V(\phi_e) = I_D \times sin(\phi_e + 120^o) \\ I_W(\phi_e) = I_D \times sin(\phi_e - 120^o) \end{cases} \quad (17)$$

4.5.4 FOC2 - Stepper Coil Currents I_X, I_Y and Associated Voltages U_X, U_Y

For two-phase motors (stepper) with four terminals UX1, VX2, and WY1, Y2, voltage U_Ux = U_X1 - U_X2 is in phase with the measured current I_X and U_Wy = U_Y1 - U_Y2 is in phase with the measured current I_Y according to equations (18) and (19) for FOC2.

$$\text{U\_XY\_FOC2} = \begin{cases} U_X(\phi_e) = U_X \times sin(\phi_e) \\ U_Y(\phi_e) = U_Y \times sin(\phi_e + 90^o) \end{cases} \quad (18)$$

$$\text{I\_XY\_FOC2} = \begin{cases} I_X(\phi_e) = I_D \times sin(\phi_e) \\ I_Y(\phi_e) = I_D \times sin(\phi_e + 90^o) \end{cases} \quad (19)$$

4.5.5 FOC1 - DC Motor Coil Current I_X1, I_X2, and Associated Voltage U_X1, U_X2

For DC motor with two terminals UX1, VX2, voltage U_X = U_X1 - U_X2 is in phase (same sign) with the measured current I_X. U_X is in phase (same sign) with the measured current I_X according to equations (20) and (21) for FOC1.

$$\text{U\_XY\_FOC1} = U_{X1} - V_{X2} \tag{20}$$

$$\text{I\_XY\_FOC1} = I_{X1} \tag{21}$$

4.5.6 ADC Selector and ADC Scaler w/ Offset Correction

The ADC selector selects ADC channels for FOC. The 3-phase FOC uses two ADC channels for measurement and calculates the third channel using Kirchhoff's Law using the scaled and offset-corrected ADC values. The 2-phase FOC just uses two ADC channels because for a 2-phase stepper motor, and the two phases are independent from each other.

NOTE The open-loop encoder is useful for setting up ADC channel selection, scaling, and offset by running a motor open-loop.

The FOC23 engine processes currents as 16-bit signed (s16) values. Raw ADC values are expanded to 16-bit width, regardless of their resolution. With this, each ADC is available for read out as a 16-bit number. The ADC scaler w/ offset correction is for the preprocessing of measured raw current values. It might be used to map to own units (example, A or mA). For scaling, gains of current amplifiers, reference voltages, and offsets must be taken into account.

NOTE Raw ADC values generally are of 16-bit width, regardless of their real resolution.

NOTE The ADC scaler maps raw ADC values to the 16-bit signed (s16) range and centers the values to zero by removing offsets.

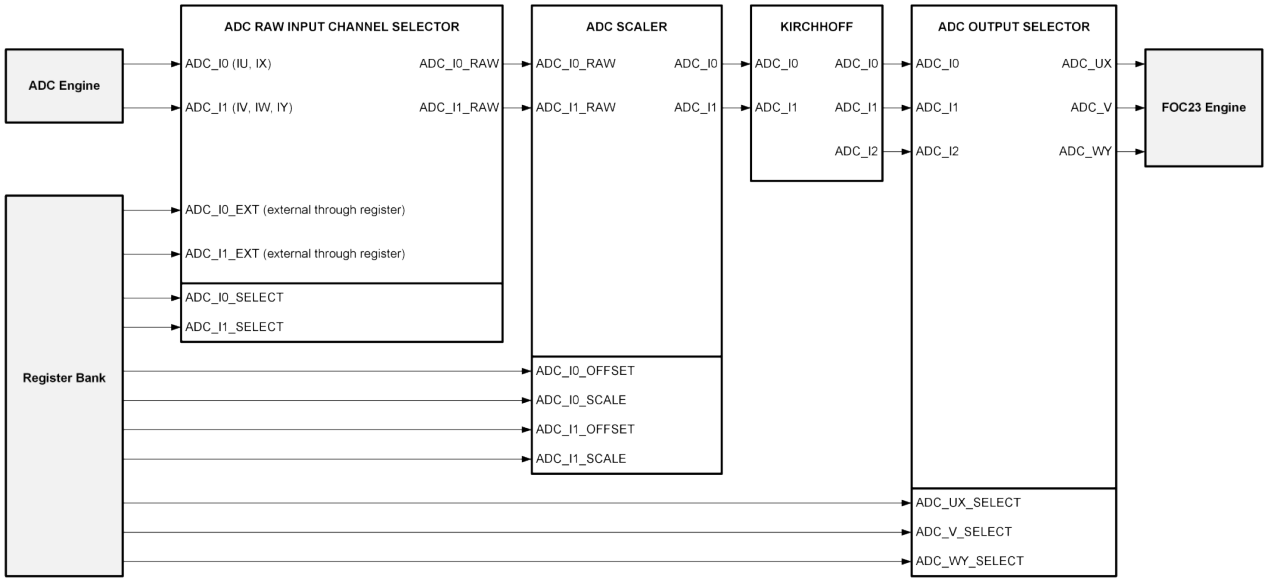


Figure 20: ADC Selector and Scaler w/ Offset Correction

ADC offsets and ADC scalers for the analog current measurement input channels must be programmed into the associated registers. Each ADC_I_U, ADC_I_V, ADC_I0_EXT, and ADC_I1_EXT are mapped either to ADC_I0_RAW or to ADC_I1_RAW by ADC_I0_SELECT and ADC_I1_SELECT.

In addition, the ADC_OFFSET is for conversion of unsigned ADC values into signed ADC values as required for the FOC. For FOC3, the third current ADC_I2 is calculated using Kirchhoff's Law. This requires the correct scaling and offset correction beforehand. For FOC2, there is no calculation of a third current. The scaling factors ADC_I0_SCALE and ADC_I1_SCALE are displayed in a Q8.8 format which results in the following equations:

$$\text{ADC\_I0} = (\text{ADC\_I0\_RAW} - \text{ADC\_I0\_OFFSET}) \times \text{ADC\_I0\_SCALE}/256 \tag{22}$$

$$\text{ADC\_I1} = (\text{ADC\_I1\_RAW} - \text{ADC\_I1\_OFFSET}) \times \text{ADC\_I1\_SCALE}/256 \tag{23}$$

The ADC_UX_SELECT selects one of the three ADC channels ADC_I0, ADC_I1, or ADC_I2 for ADC_UX.

The ADC_V_SELECT selects one of the three ADC channels ADC_I0, ADC_I1, or ADC_I2 for ADC_V.

The ADC_WY_SELECT selects one of the three ADC channels ADC_I0, ADC_I1, or ADC_I2 for ADC_WY.

The ADC_UX, ADC_V, and ADC_WY are for the FOC3 (U, V, W).

The ADC_UX and ADC_WY (X, Y) are for the FOC2 (UX, WY).

NOTE The open-loop encoder is useful to run a motor open loop for setting up the ADC channel selection with correct association between phase currents I_U, I_V, I_W and phase voltages U_U, U_V, U_W.

4.6 Encoder Engine

The encoder engine is a unified position sensor interface. It maps the selected encoder position information to electrical position (phi_e) and to mechanical position (phi_m). Both are 16-bit values. The encoder engine maps single turn positions from position sensors to multi-turn positions. It is possible to overwrite the multi-turn position for initialization.

The different position sensors are the position sources for torque and flux control through FOC, for velocity control, and for position control. The PHI_E_SELECTION selects the source of the electrical angle phi_e for the inner FOC control loop. VELOCITY_SELECTION selects the source for velocity measurement. With phi_e selected as source for velocity measurement, one gets the electrical velocity. With the mechanical angle phi_m selected as source for velocity measurement, one gets the mechanical velocity taking the set number of pole pairs (N_POLE_PAIRS) of the motor into account. Nevertheless, for a highly precise positioning, it might be useful to do positioning based on the electrical angle phi_e.

4.6.1 Open-Loop Encoder

For initial system setup, the encoder engine is equipped with an open-loop position generator. This allows turning the motor open-loop by specifying speed in RPM and acceleration in RPM/s, together with a voltage UD_EXT in D direction. As such, the open-loop encoder is not a real encoder. It simply gives positions as an encoder does. The open-loop decoder has a direction bit to define the direction of motion for the application.

NOTE The open-loop encoder is useful for initial ADC setup, encoder setup, hall signal validation, and for validation of the number of pole pairs of a motor. The open-loop encoder turns a motor open with programmable velocity in unit [RPM] with programmable acceleration in unit [RPM/s].

With the open-loop encoder, turn a motor without any position sensor and without any current measurement as a first step of doing the system setup. With the turning motor, it is posisble to adjust the ADC scales and offsets and set up positions sensors (hall, incremental encoder, ...) according to resolution, orientation, and direction of rotation.

4.6.2 Incremental ABN Encoder

The incremental encoders give two phase shifted incremental pulse signals A and B. Some incremental encoders have an additional null position signal N or zero pulse signal Z. An incremental encoder (called ABN encoder or ABZ encoder) has an individual number of incremental pulses per revolution. The number of incremental pulses define the number of positions per revolution (PPR). The PPR might mean pulses per revolution or periods per revolution. Instead of positions per revolution, some incremental encoder vendors call these counts per revolution (CPR).

The PPR parameter is the most important parameter of the incremental encoder interface. With this, it forms a modulo (PPR) counter, counting from 0 to (PPR-1). Depending on the direction, it counts up or down. The modulo PPR counter is mapped into the register bank as a dual ported register. It is possible to overwrite it with an initial position. The ABN encoder interface provides both the electrical position and the multi-turn position, which are accessible through dual-ported read-write registers.

NOTE The PPR parameter must be set exactly according to the used encoder.

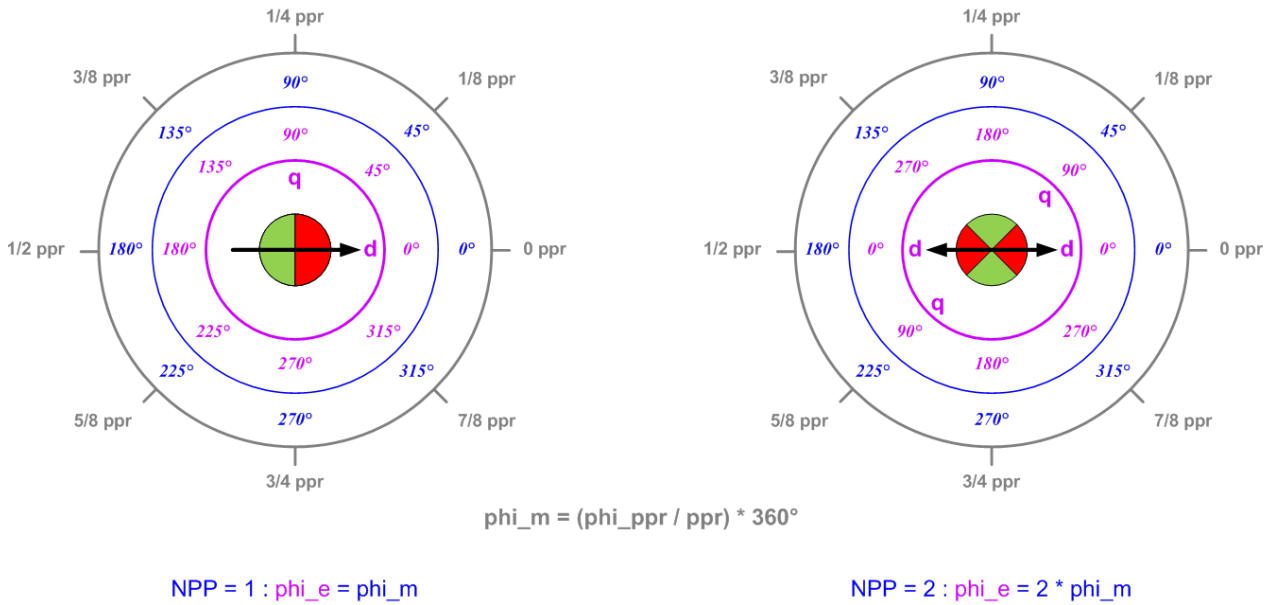


Figure 21: Number of Pole Pairs NPP vs. Mechanical Angle phi_m and Electrical Angle phi_e

The goal of the initialization of an incremental encoder is to set it up so that the magnetic axis of the rotor fits with the electrical angle phi_e with the angle zero on D axis. For this, it is necessary to know the number of pole pairs NPP, the resolution of the incremental encoder in pulses per revolution PPR, and the orientation between measured encoder angle of the rotor and the electrical angle of the field orientation. An encoder measures mechanical angle phi_m, where the FOC needs the electrical angle phi_e for commutation. The number of pole pairs NPP determines the ratio between mechanical angle phi_m and electrical angle phi_e. The parameters phi_m_offset and phi_e_offset are for compensation of differences in orientation angle by adjustments.

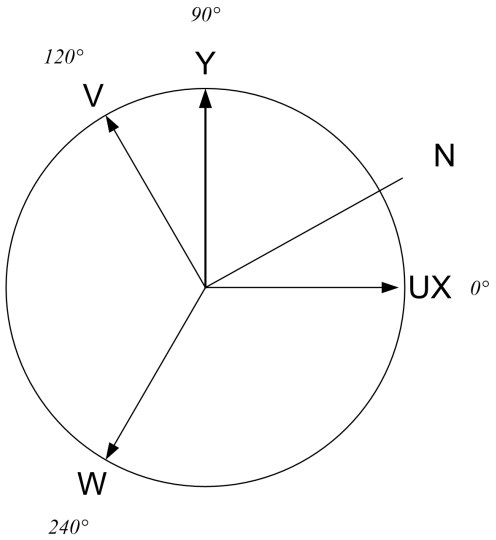


Figure 22: ABN Incremental Encoder N Pulse

The N pulse from an encoder triggers either sampling of the actual encoder count to fetch the position at the N pulse or it rewrites the fetched n position on an N pulse. The N pulse can either be used as a standalone pulse or and-ed with NAB = N and A and B. It depends on the decoder what kind of N pulse has to be used - either N or NAB. For those encoders with precise N pulse within one AB quadrant, the N pulse must be used. For those encoders with N pulse over four AB quadrants, enhance the precision of the N pulse position detection by using NAB instead of N.

NOTE	Incremental encoders are available with N pulse and without N pulse.

The polarity of N pulse, A pulse, and B pulse are programmable. The N pulse is for reinitialization with each turn of the motor. Once fetched, the ABN decoder can be configured to write back the fetched N pulse position with each N pulse.

NOTE	The ABN encoder interface has a direction bit to set to match the wiring of the motor to the direction of the encoder.

Logical ABN = A and B and N might be useful for incremental encoders with low resolution N pulse to enhance the resolution. On the other hand, for incremental encoders with high resolution N pulse, a logical ABN = A and B and N might totally suppress the resulting N pulse.

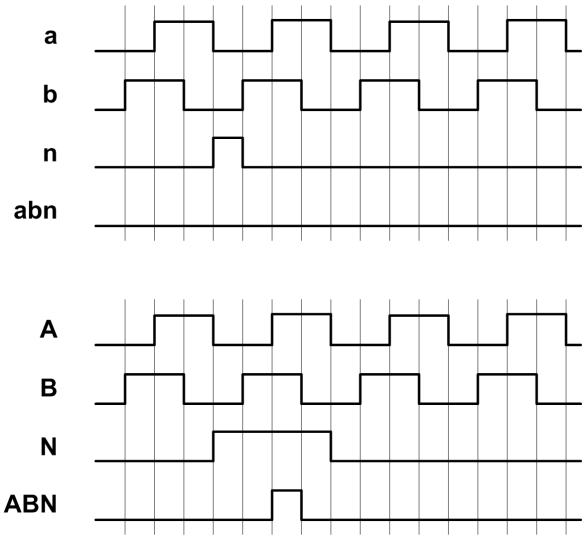


Figure 23: Encoder ABN Timing - High Precise N Pulse and Less Precise N Pulse

4.6.3 Secondary Incremental ABN Encoder

For commutating a motor with FOC, select a position sensor source (digital incremental encoder, digital hall, analog hall, analog incremental encoder, ...) mounted close to the motor. The inner FOC loop controls torque and flux of the motor based on the measured phase currents and the electrical angle of the rotor.

The TMC4671 is equipped with a secondary incremental encoder interface. This secondary encoder interface is available as the source for velocity control or position control. This is for applications where a motor with a gearing positions an object.

NOTE The secondary incremental encoder is not available for commutation (phi_e) for the inner FOC. In others words, there is no electrical angle phi_e selectable from the secondary encoder.

4.6.4 Digital Hall Sensor Interface with Optional Interim Position Interpolation

The digital hall interface is the position sensor interface for digital hall signals. The digital hall signal interface first maps the digital hall signals to an electrical position PHI_E_RAW using the contents of the HALL_POSITION_XXX_YYY registers (0x34, 0x35, and 0x36). An offset PHI_E_OFFSET can be used to rotate the orientation of the hall signal angle. The electrical angle PHI_E is for commutation. Optionally, the default electrical positions of the hall sensors can be adjusted by writes into the associated registers. The default values for decode are listed in Table 21.

Digital Hall signals (U,V,W)	Polarity = 1 (0x33)	Registerfield	Corresponding PHI_E (default)
1 1 0	0 0 1	HALL_POSITION_000	0x0000
0 1 0	1 0 1	HALL_POSITION_060	0x2AAA
0 1 1	1 0 0	HALL_POSITION_120	0x5555

0 0 1	1 1 0	HALL_POSITION_180	0x7FFF
1 0 1	0 1 0	HALL_POSITION_240	0xAAAA
1 0 0	0 1 1	HALL_POSITION_300	0xD555

Table 21: Digital Hall Decode

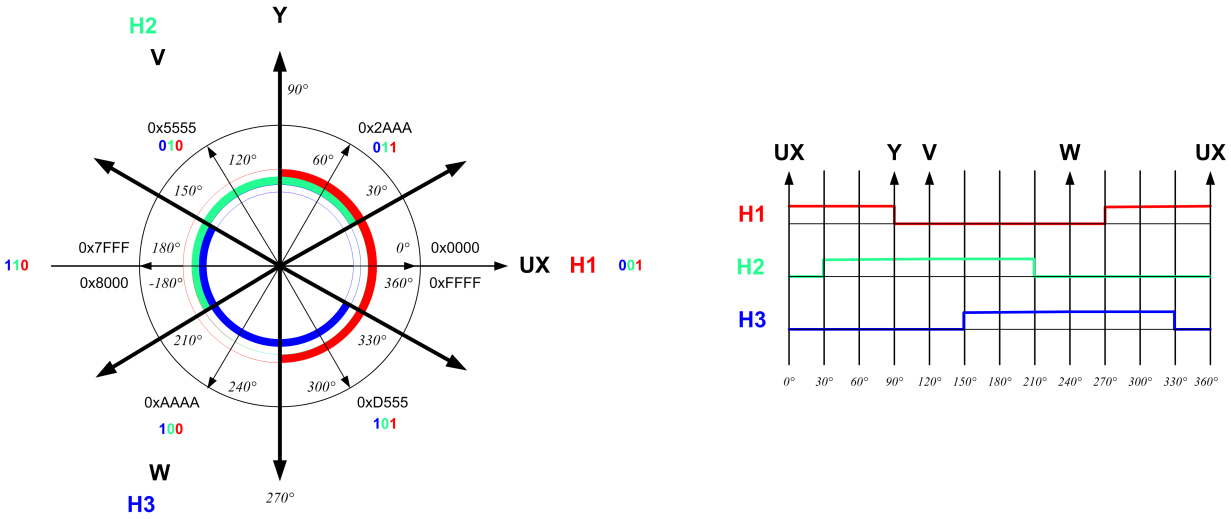


Figure 24: Principle of Hall Sensor Angles

Hall sensors give absolute positions within an electrical period with a resolution of 60° as 16-bit positions (s16 resp. u16) PHI. With activated interim hall position interpolation, gets high resolution interim positions when the motor is running at a speed above 60 electrical RPM.

4.6.5 Digital Hall Sensor - Interim Position Interpolation

For lower torque ripple, switch on the position interpolation of interim hall positions. This function is useful for motors compatible with sine wave commutation, but equipped with digital hall sensors. When the position interpolation is switched on, it becomes active on speeds above 60rpm. For lower speeds, it automatically disables itself. This is especially important when the motor has to be at rest. Hall sensor position interpolation might fail when hall sensors are not properly placed in the motor. Adjust hall sensor positions for this case.

NOTE

Hall interpolation is not intended for positioning applications, especially not with changes of direction. Check the errata section when using hall interpolation and positioning. When using interpolated angles, PID_POSITION_ACTUAL might glitch when the direction is changed or the motor operates at very low velocity.

4.6.6 Digital Hall Sensors - Masking, Filtering, and PWM Center Sampling

Sometimes digital hall sensor signals get disturbed by switching events in the power stage. The TMC4671 can automatically mask switching distortions by correct setting of the HALL_MASKING register. When a

switching event occurs, the hall sensor signals are held for HALL_MASKING value times 10 ns. This way, hall sensor distortions are eliminated.

Uncorrelated distortions can be filtered using a digital filter of configurable length. If the input signal to the filter does not change for HALL_DIG_FILTER_LENGTH times 5µs, the signal can pass the filter. This filter eliminates issues with bouncing hall signals. Naming with Elliot: Masking is better then Blanking.

Spikes on hall signals (Hx that stands for H1, H2, H3) disturb the FOC loop when hall signals are used for commutation or for initialization of incremental encoders. Spikes on hall signal lines might occur when hall signals are fed on singled-ended signal lines in parallel to motor power lines due to electromagnetic crosstalk in a single cable. Long hall signal lines might cause digital hall signal crosstalk even in separate fed cables. Cables that provide hall signals without spikes should be preferred. A good ground for digital hall signals is important for clean hall signals. A good ground shield of the motor might help for clean hall signals. In the best case, hall signals are fed within separate shielded signal lines together with differential line drivers.

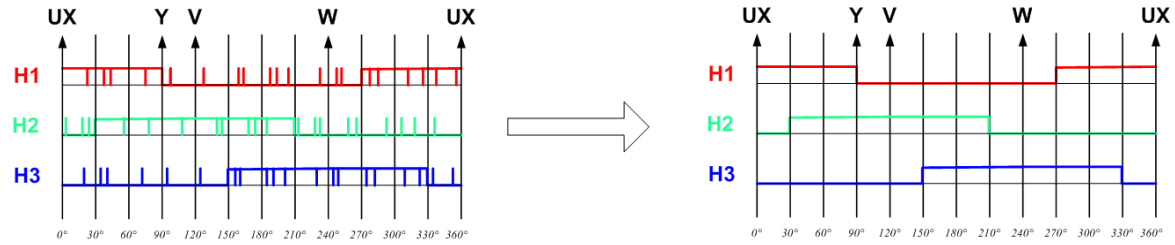


Figure 25: Outline of Noisy Hall Signals (Left) due to Electromagnetic Interference with PWM Switching and Noise Cleaned Hall Signals (Right) by PWM Center Synced Sampling of Hall Signal Vector (H1, H2, H3)

The best option is to avoid spikes on digital hall signals. Nevertheless, to enable lower cost motors with lower performance hall signal shielding, the TMC4671 is equipped with sall Signal spike suppression and PWM centered hall signal vector sampling.

To reduce possible current ripple that might be caused by noisy hall signals, the sampling of the hall signal vector can be programmed for sampling once per PWM period at its center for the desired noise reduction. The PWM centered hall signal sampling is programmable by HALL_MODE(4) control bit. Continuous sampling is default. This function is not available for TMC4671-ES engineering samples.

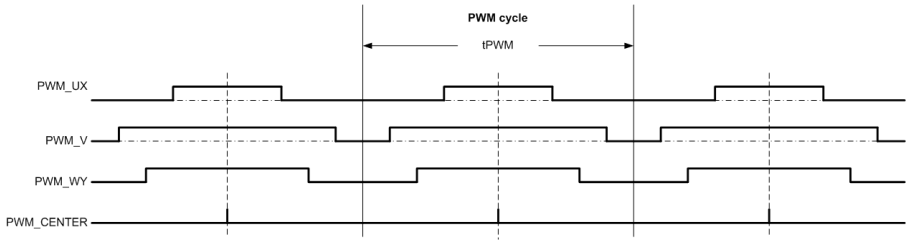


Figure 26: Hall Signal PWM Center Sampling on PWM_CENTER

The PWM center synchronization must be qualified for high speed applications due to reduction of hall signals for PWM frequency. The PWM center might have an influence on hall signal interpolation and must be qualified if hall signal interpolation is enabled.

For additional spike suppression, the TMC4671 is equipped with a digital hall signal blanking, to support lower performance cabling environments. The blank time for the hall signals is programmable (HALL_BLANK) in steps of 10ns from 0ns up to 4095ns. The hall signal blanking time should be programmed as long as necessary for safe suppression of spikes of maximum duration. On the other side, the hall signal blanking should be programmed as short as possible to avoid disturbance by too strong filtering that might also disturb the FOC.

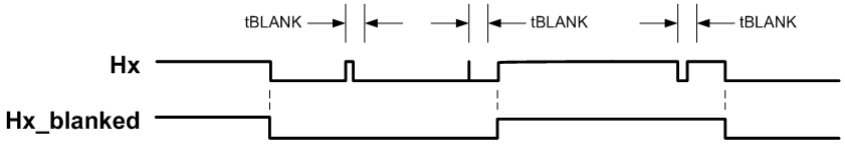


Figure 27: Hall Signal Blanking

4.6.7 Digital Hall Sensors Together with Incremental Encoder

If a motor is equipped with both hall sensors and incremental encoder, the hall sensors can be used for the initialization as a low resolution absolute position sensor. Later on, the incremental encoder can be used as a high resolution sensor for commutation.

4.6.8 Analog Hall and Analog Encoder Interface (SinCos of 0° 90° or 0° 120° 240°)

An analog encoder interface is part of the decoder engine. It is able to handle analog position signals of 0° and 90° and of 0° 120° 240°. The analog decoder engine adds offsets and scales the raw analog encoder signals, while also calculating the electrical angle PHI_E from these analog position signals by an ATAN2 algorithm.

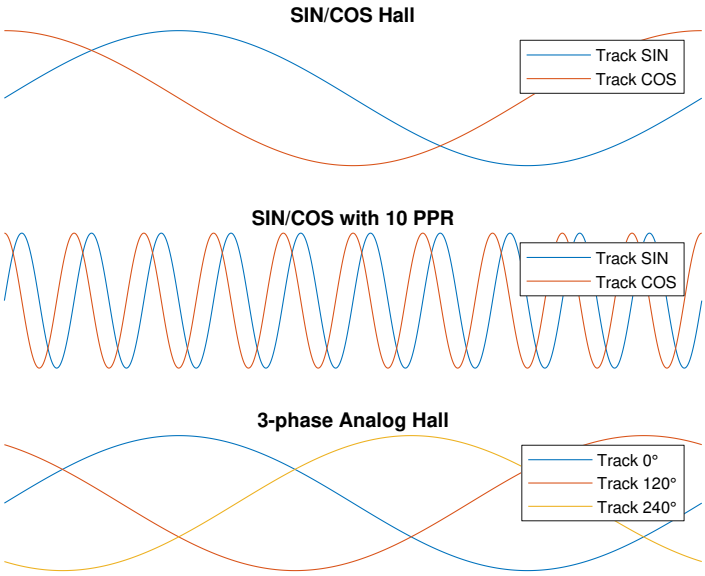


Figure 28: Analog Encoder (AENC) Signal Waveforms

An individual signed offset is added to each associated raw ADC channel and scaled by its associated scaling factors according to:

$$\text{AENC\_VALUE} = (\text{AENC\_RAW} + \text{AENC\_OFFSET}) \times \text{AENC\_SCALE} \tag{24}$$

In addition, the AENC_OFFSET converts unsigned ADC values into signed ADC values as required for the FOC.

NOTE The control bit 0 in register AENC_DECODER_MODE (0x3B) selects either processing of analog position signals of 0° and 90° (0b0) or analog signals of 0° 120° 240° on (0b1).

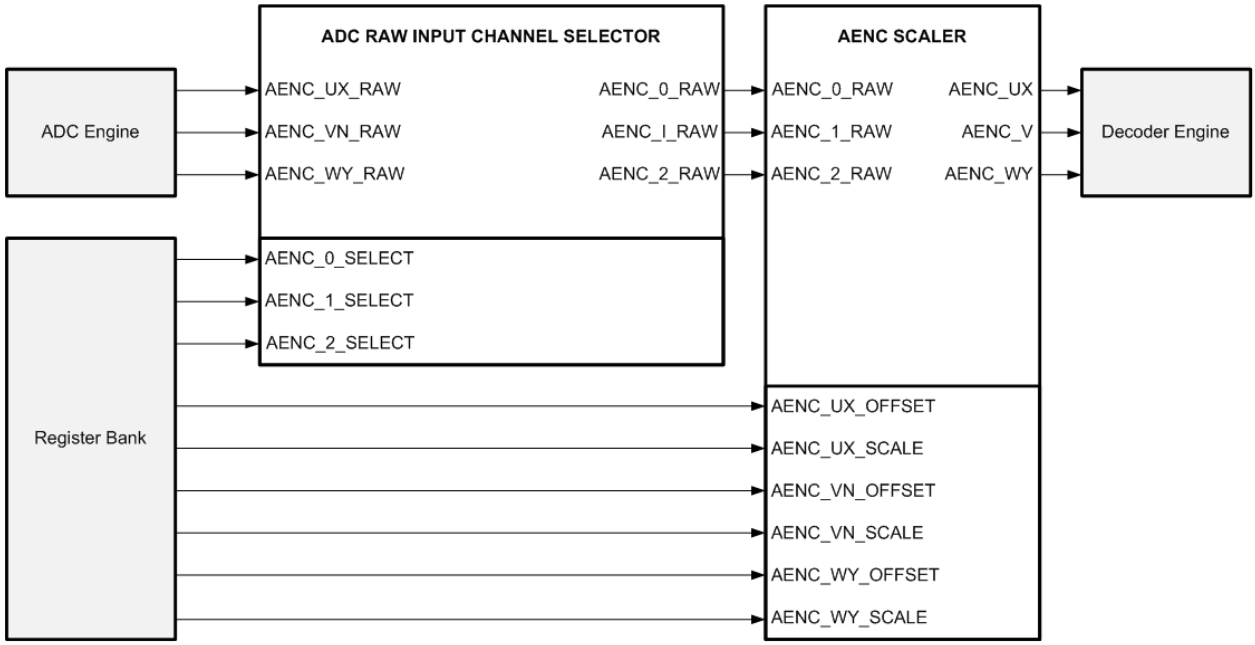


Figure 29: Analog Encoder (AENC) Selector & Scaler w/ Offset Correction

Figure 28 shows the possible waveforms. The graphs show usual SIN/COS track signals with one and multiple periods per revolution as well as typical waveforms of three-phase analog hall signals for one electrical revolution. The number of periods per revolution can be configured by register AENC_DECODER_PPR. The position in one period (AENC_DECODER_PHI_A) is calculated by an ATAN2 algorithm. The periods are counted with respect to the number of periods per revolution to calculate AENC_DECODER_PHI_E and AENC_DECODER_PHI_M. If PPR is the same as the number of pole pairs, AENC_DECODER_PHI_E and AENC_DECODER_PHI_A are identical. This is usually the case for analog hall signals.

NOTE The analog N pulse is just a raw ADC value. Handling of analog N pulse similar to N pulse handling of digital encoder N pulse is not implemented for analog encoder.

NOTE The used PPR in 0x40 AENC_DECODER_PPR must be 1 or a power of 2. When using different PPR, AENC_DECODER_COUNT must stay within $[-2^{31}; -2^{16}]$ and $[2^{16}; 2^{31}]$. For this case, 0x41 AENC_DECODER_COUNT must be overwritten during encoder initialization.

4.6.9 Analog Position Decoder (SinCos of 0°90° or 0°120°240°)

The extracted positions from the analog decoder are available for read out from registers.

4.6.9.1 Multi-Turn Counter

Electrical angles are mapped to a multi-turn position counter. It is possible to overwrite this multi-turn position for initialization purposes.

4.6.9.2 Encoder Engine Phi Selector

The angle selector selects the source for the commutation angle PHI_E. That electrical angle is available for commutation.

4.6.9.3 External Position Register

A register value written into the register bank through the application interface is available for commutation as well. With this, it is possible to interface to any encoder by just writing positions extracted from the external encoder into this regulator. From the decoder engine point of view, this is just one more selectable encoder source.

4.6.10 Encoder Initialization Support

The TMC4671 needs proper feedback for correct and stable operation. One main parameter is the commutation angle offset PHI_E_OFFSET. This offset must not be calculated when an absolute sensor system like analog or digital hall sensors is used. All other supported feedback systems must be initialized - their PHI_E_OFFSETs must be identified. There are several options to determine PHI_E_OFFSET with support of the TMC4671.

4.6.10.1 Encoder Initialization in Open-Loop Mode

In the case of a free driving motor, the motor can be switched to open-loop mode. In this mode, the used commutation angle (PHI_OPEN_LOOP) can be used to match the measured PHI_E. This method is supported by the TMCL-IDE.

4.6.10.2 Encoder Initialization by N Pulse Detection

After determination of a correct offset, the value can be used again after the power cycle. The encoder's N pulse can be used as reference for this. For starters, it is possible to drive the motor in open-loop mode or using digital hall sensor signals. After passing the encoder's N pulse, the ABN encoder is initialized and can be used for operation.

4.6.11 Velocity Measurement

Servo control comprises position, velocity, and current control. The position and the current are measured by separate sensors. The actual velocity has to be calculated by time discrete differentiation from the position signal. It is possible to choose a calculated position from the various encoder interfaces for velocity measurement by parameter VELOCITY_SELECTION.
It is possible to switch between two different velocity calculation algorithms with the parameter VELOCITY_METER_SELECTION. Default setting (VELOCITY_METER_SELECTION = 0) is the standard velocity meter, which calculates the velocity at a sampling rate of about 4369.067Hz by differentiation. Output value is displayed in revolutions per minute (RPM). This option is recommended for usage with the standard PI controller structure.
By choosing the second option (VELOCITY_METER_SELECTION = 1), the sampling frequency is synchronized to the PWM frequency. This option is recommended for usage with the advanced PI controller structure. Otherwise, the controller structure might tend to be unstable due to non-matched sampling.
Velocity filters can be applied to reduce noise on velocity signals. Section 4.8 describes filtering opportunities in detail.

4.6.12 Reference Switches

The TMC4671 is equipped with three input pins for reference switches (REF_SW_L, REF_SW_H and REF_SW_R). These pins can be used to determine three reference positions. The TMC4671 displays the status of the reference switches in the register TMC4671_INPUTS_RAW and is able to store the actual position at rising edge of the corresponding signal. The signal polarities are programmable and can be filtered with a configurable digital filter, which suppresses spike errors.
With the STATUS_FLAGS and STATUS_MASK registers, the STATUS output can be configured as an IRQ for passing a reference switch.
The actual position can be latched when passing a reference switch. The latched positions are displayed in register INTERIM_DATA (0x6E). Register INTERIM_ADDR (0x6F) selects the data displayed in INTERIM_DATA with HOME_POSITION at address 31, LEFT_POSITION at adress 32, and RIGHT_POSITION at adress 33.
The position latching can be enabled through register CONFIG_DATA (0x4D) with CONFIG_ADDR (0x4E) set to 51 (ref_switch_config). Position latching is enabled by setting bit 0 of ref_switch_config to 1.If a reference switch is passed, the corresponding status bit (HOME_SWITCH_PASSED, LEFT_SWITCH_PASSED, and RIGHT_SWITCH_PASSED) in REF_SWITCH_STATUS (INTERIM_DATA with INTERIM_ADDRESS = 30) is enabled. The bits can only be cleared by toggling/disabling the enable bit in ref_switch_config.

NOTE	The polarity registers do not affect the status registers. The status flag only represents the current logical state of the switch.

4.7 FOC23 Engine

The FOC23 engine performs the inner current control loop for the torque current I_Q and the flux current I_D including the required transformations. Programmable limiters take care of clipping of interim results. Per default, the programmable circular limiter clips U_D and U_Q to U_D_R = $\sqrt{(2)}\times$ U_Q and U_R_R = $\sqrt{(2)}\times$ U_D. PI controllers perform the regulation tasks. Please make sure to enable controllers by pulling ENI pin to high level.

4.7.1 ENI and ENO pins

The enable input (ENI) can be used to start and stop control action. During reset, enable output (ENO) is low and afterwards it forwards ENI signal. Thereby, it can be used to enable the power stage. When ENI is low, all controllers are deactivated and PWM operates at 50% duty cycle. ENI input value can be read through the TMC4671_INPUTS_RAW register.

4.7.2 PI Controllers

PI controllers are used for current control and velocity control. A P controller is used for position control. The derivative part is not yet supported but might be added in the future. It is possible to choose between two PI controller structures: the classic PI controller structure, which is also used in the TMC4670, and the advanced PI controller structure. The advanced PI controller structure shows better performance in dynamics and is recommended for high performance applications. Switch between controllers by setting register MODE_PID_TYPE. Controller type can not be switched individually for each cascade level.

4.7.3 PI Controller Calculations - Classic Structure

The PI controllers in the classic structure perform the following calculation

$$Y = P \times e + I \times \int_0^t e(t)\,dt \tag{25}$$

with

$$e = X\_TARGET - X \tag{26}$$

where X_TARGET stands for target flux (s16), target torque (s16), target velocity (s32), or target position (s32) with error e, which is the difference between target value and actual values. The Y stands for the output of the PI controller feed as target input to the successive PI controller of the FOC servo controller cascade (position $\rightarrow$ PI $\rightarrow$ velocity $\rightarrow$ PI $\rightarrow$ current $\rightarrow$ PI $\rightarrow$ voltage).

Y_PID_FLUX	=	PID_FLUX_P	$\times$	ERROR_FLUX/256
Y_PID_FLUX_RATE	=	PID_FLUX_I	$\times$	ERROR_FLUX/65536/(32µs)
Y_PID_TORQUE	=	PID_FLUX_P	$\times$	ERROR_TORQUE/256
Y_PID_TORQUE_RATE	=	PID_TORQUE_I	$\times$	ERROR_TORQUE/65536/(32µs)
Y_PID_VELOCITY	=	PID_VELOCITY_P	$\times$	ERROR_VELOCITY/256
Y_PID_VELOCITY_RATE	=	PID_VELOCITY_I	$\times$	ERROR_VELOCITY/65536/(256µs)

Y_PID_POSITION	=	PID_POSITION_P	×	ERROR_POSITION/65536
Y_PID_POSITION_RATE	=	PID_POSITION_I	×	ERROR_POSITION/65536/(256µs)

Table 22: Scalings and Change Rate Timings of PID Controllers (Classic Structure) for Currents, Velocity, and Position for Clock Frequency fCLK = 25MHz

Number	Motion Mode	Description
0	Stopped Mode	Disabling all controllers
1	Torque Mode	Standard Torque Control Mode
2	Velocity Mode	Standard Velocity Control Mode
3	Position Mode	Standard Position Control Mode
4 to 7	Reserved	Reserved
8	UQ UD Ext Mode	Voltage Control Mode (Software Mode)
9	Reserved	reserved
10	AGPI_A Torque Mode	AGPI_A Used as Target Torque Value
11	AGPI_A Velocity Mode	AGPI_A Used as Target Velocity Value
12	AGPI_A Position Mode	AGPI_A Used as Target Position Value
13	PWM_I Torque Mode	PWM_I Used as Target Torque Value
14	PWM_I Velocity Mode	PWM_I Used as Target Velocity Value
15	PWM_I Position Mode	PWM_I Used as Target Position Value

Table 23: Motion Modes

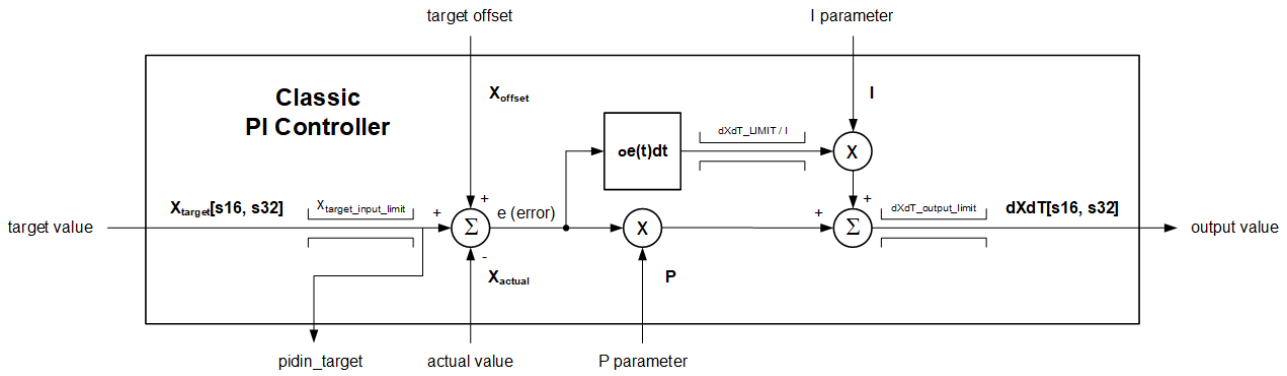


Figure 30: Classic PI Controller

NOTE

Changing the I-parameter of the classic PI controller during operation causes the controller output to jump, as the control error is first integrated and then gained by the I parameter. Jumps can be avoided by incremental changes of I-parameter.

NOTE

Support for the TMC4671 is integrated into the TMCL-IDE, including wizards for set up and configuration. With the TMCL-IDE, configuration and operation can be done in a few steps with direct access to all registers of the TMC4671.

4.7.4 PI Controller Calculations - Advanced Structure

The PI controllers in the advanced controller structure perform the calculation:

$$\mathrm{dXdT} = \mathrm{P} \times \mathrm{e} + \int_0^t \mathrm{P} \times \mathrm{I} \times \mathrm{e(t)}\,\mathrm{d}t \tag{27}$$

with:

$$\mathrm{e} = \mathrm{X\_TARGET} - \mathrm{X} \tag{28}$$

where X_TARGET represents target flux, target torque, target velocity, or target position with control error e, which is the difference between target value and actual values. The time constant $\mathrm{d}t$ is set according to the PWM period but can be downsampled for the position controller by register MODE_PID_SMPL. Position controller evaluation can be downsampled by a constant factor, when needed.

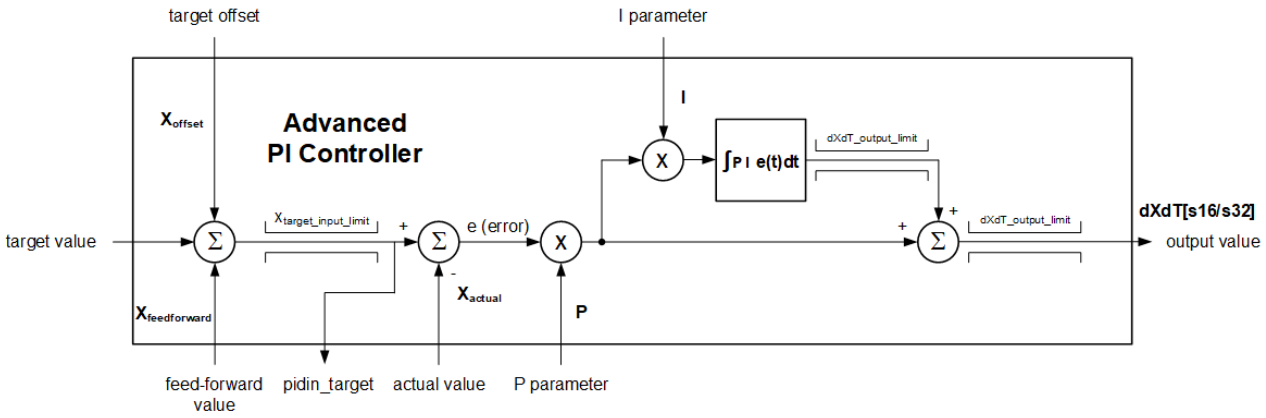


Figure 31: Advanced PI Controller

NOTE

The P Factor normalization as Q8.8 of the advanced PI controller of the TMC4671-ES is selectable for the TMC4671-LA as either Q8.8 or Q4.12. This can be configured in register 0x4D CONFIG_DATA when register 0x4E CONFIG_ADDR is set to 0x3E. For more information, see section 7.2. Using Q4.12 needs changes in the user's application controller software when using the advanced PI position controller.

The transfer function of the advanced PI controller can be described by the following pseudo code:

$$\begin{aligned} \text{dXdT} &= \text{e} \times \text{P} + \text{integrator} \\ \text{integrator} &= \text{integrator} + (\text{P} \times \text{I} \times \text{e})/256 \end{aligned} \tag{29}$$

NOTE For the advanced control structure, the integrator input value is additionally divided by 256. (see equation 29)

P and I are either displayed as Q8.8 (P/256) or Q4.12 (P/4096). This is individually configurable for each controller parameter in the controller cascade.Table 24 gives an overview on how the representation affects the integrator and the output of the PI controller.

PI representation	**I 8.8**	**I 4.12**
P 8.8	$\text{dXdT} = \text{e} \times \frac{\text{P}}{256} + \text{integrator}$ $\text{integrator} = \text{integrator} + \left(\frac{\text{P}}{256} \times \frac{\text{I}}{256} \times \text{e}\right)/256$	$\text{dXdT} = \text{e} \times \frac{\text{P}}{256} + \text{integrator}$ $\text{integrator} = \text{integrator} + \left(\frac{\text{P}}{256} \times \frac{\text{I}}{4096} \times \text{e}\right)/256$
P 4.12	$\text{dXdT} = \text{e} \times \frac{\text{P}}{4096} + \text{integrator}$ $\text{integrator} = \text{integrator} + \left(\frac{\text{P}}{4096} \times \frac{\text{I}}{256} \times \text{e}\right)/256$	$\text{dXdT} = \text{e} \times \frac{\text{P}}{4096} + \text{integrator}$ $\text{integrator} = \text{integrator} + \left(\frac{\text{P}}{4096} \times \frac{\text{I}}{4096} \times \text{e}\right)/256$

Table 24: PI Normalization Overview

Downsampling of the advanced position controller can be configured by register MODE_PID_SMPL. When the register is 0, the controllers sample on the PWM-frequency f_{PWM}. The new sample rate is derived from f_{PWM} and the downsampling value assigned to register MODE_PID_SMPL (range: 0 to 127). A reasonable setting is in the range of 0 to 25. The derived sampling frequency is calculated as follows:

$$\text{Samplerate}_{\text{new}} = \frac{f_{PWM}}{\text{downsampling} + 1} \tag{30}$$

4.7.5 PI Controller - Clipping

The limiting of target values for PI controllers and output values of PI controllers is programmable. Per power on default, these limits are set to maximum values. During initialization, these limits should be set properly for correct operation and clipping.
The target input is clipped to X_TARGET_LIMIT. The output of a PI controller is named dXdT because it gives the desired derivative d/dt as a target value to the following stage: the position (x) controller gives velocity (dx/dt). The output of the PI Controller is clipped to dXdT_LIMIT. The error integral of (25) is clipped to dXdT_LIMIT / I in the classic controller structure, and the integrator output is clipped to dXdT_output_limit in the advanced controller structure.

The output of the torque and flux controller is limited by register 0x5D PIDOUT_UQ_UD_LIMITS.
The input of the torque and flux controller is limited by register 0x5E PID_TORQUE_FLUX_LIMITS
The input of the velocity controller is limited by register 0x60 PID_VELOCITY_LIMIT.
The minimum input of the position controller is limited by register 0x61 PID_POSITION_LIMIT_LOW.
The maximum input of the position controller is limited by register 0x62 PID_POSITION_LIMIT_HIGH.

The clipped controller targets can be read from register 0x6E INTERIM_DATA and its subregisters PIDIN_TARGET_TORQUE, PIDIN_TARGET_FLUX, PIDIN_TARGET_VELOCITY, and PIDIN_TARGET_POSITION.

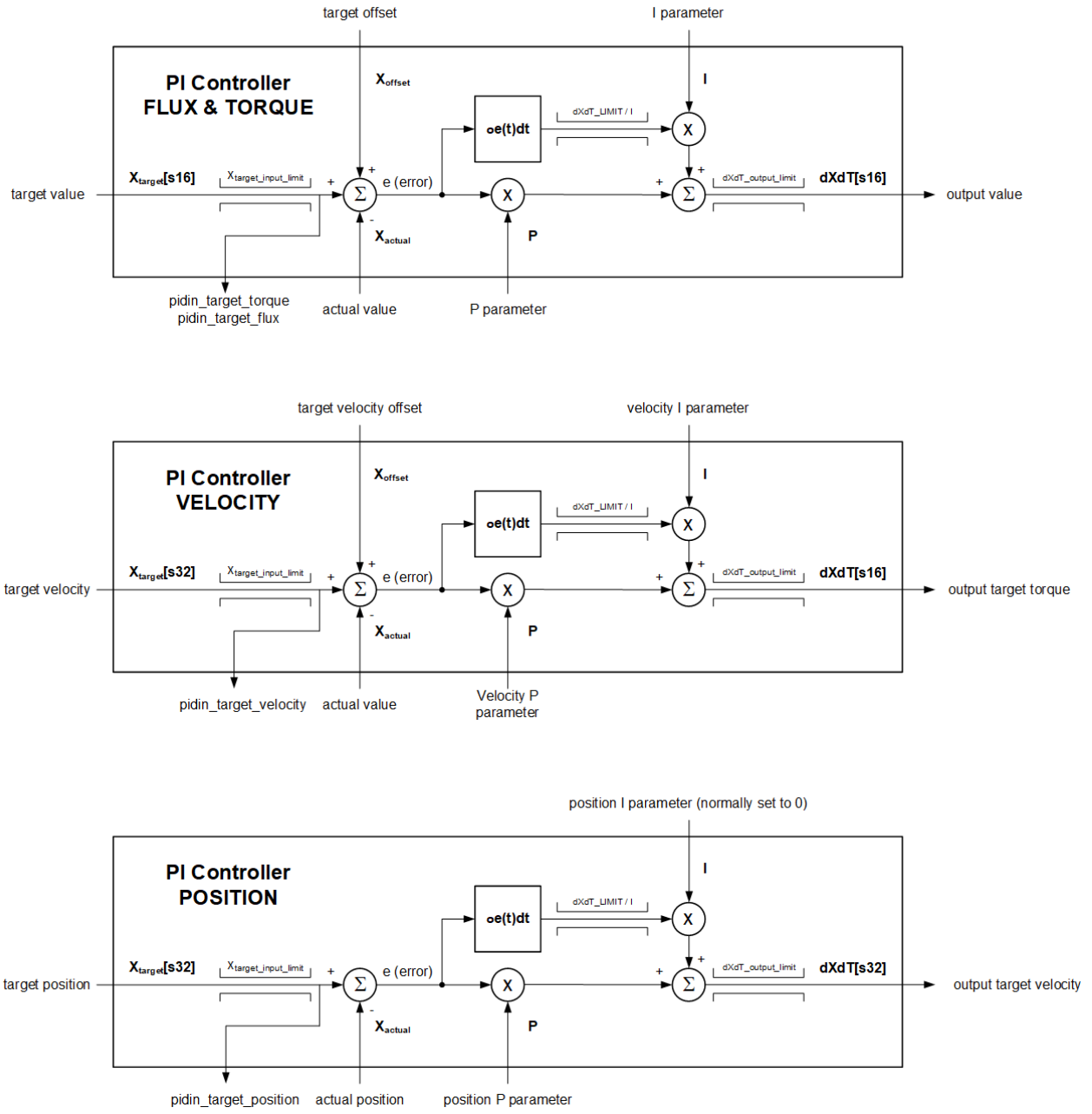


Figure 32: PI Controllers for Position, Velocity, and Current

4.7.6 PI Flux and PI Torque Controller

For the classic control structure, the P part is represented as q8.8 and the I part represented as q0.15. For the advanced control structure, the P and I part are selectable as q8.8 or q4.12. CheckTable 24 for more information on the advanced structure.

NOTE

Applying a negative value to PID_FLUX_TARGET in register 0x64 results in field weakening. A negative flux target induces a magnetic field that is opposing the magnetic field of the rotor, which in turn reduces the generated backEMF. This might result in higher maximum velocity at the cost of reduced maximum torque and increased current consumption. -1000 is a good starting value for the flux target. Increment/decrement this parameter to achieve the needed increase in velocity.

4.7.7 PI Velocity Controller

For the classic control structure, the P part is represented as q8.8 and the I part represented as q0.15. For the advanced control structure, the P and I part are selectable as q8.8 or q4.12. CheckTable 24 for more information on the advanced structure.

4.7.8 P Position Controller

For the classic position regulator, the P part is represented as q4.12 to be compatible with the high resolution positions - one single rotation is handled as an s16. For the advanced controller structure, the P part is selectable as q8.8 or q4.12. CheckTable 24 for more information on the advanced structure.

4.7.9 Inner FOC Control Loop - Flux and Torque

The inner FOC loop (Figure 33) controls the flux current to the flux target value and the torque current to the desired torque target. The inner FOC loop performs the desired transformations according to Figure 34 for 3-phase motors (FOC3). For 2-phase motors (FOC2), both Clarke (CLARKE) transformation and inverse Clarke (iCLARKE) are bypassed. For control of DC motors, transformations are bypassed and only the first full bridge (connected to X1 and X2) is used.

The inner FOC control loop gets a target torque value (I_Q_TARGET), which represents acceleration, the rotor position, and the measured currents as input data. Together with the programmed P and I parameters, the inner FOC loop calculates the target voltage values as input for the PWM engine.

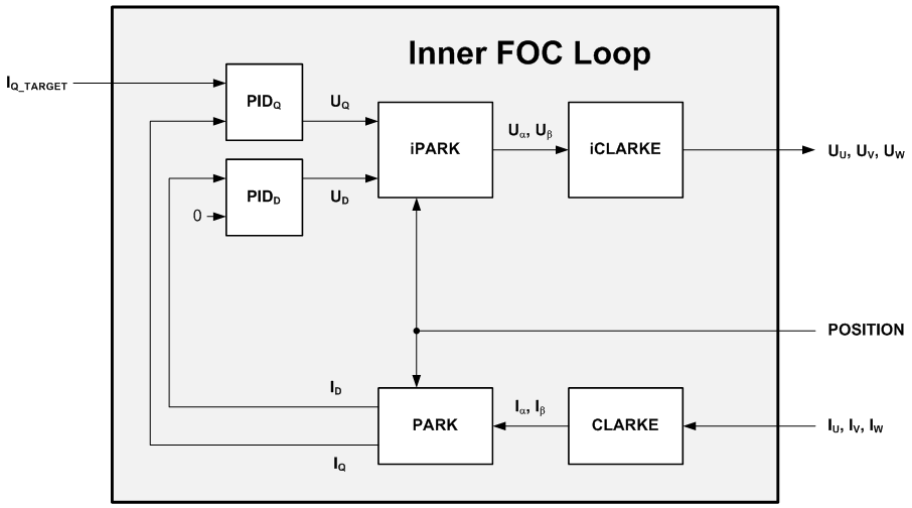


Figure 33: Inner FOC Control Loop

4.7.10 FOC Transformations and PI(D) for Control of Flux and Torque

The Clarke transformation (CLARKE) maps three motor phase currents (I_U, I_V, I_W) to a two-dimensional coordinate system with two currents (I_α, I_β). Based on the actual rotor angle determined by an encoder or through sensorless techniques, the Park transformation (PARK) maps these two currents to a quasi-static coordinate system with two currents (I_D, I_Q). The current I_D represents flux and the current I_Q represents torque. The flux just pulls on the rotor but does not affect torque. The torque is affected by I_Q. Two PI controllers determine two voltages (U_D, U_Q) to drive desired currents for a target torque and a target flux. The determined voltages (U_D, U_Q) are retransformed into the stator system by the inverse Park transformation (iPARK). The inverse Clarke Transformation (iCLARKE) transforms these two currents into three voltages (U_U, U_V, U_W). These three voltage are the input of the PWM engine to drive the power stage.

In case of the FOC2, Clarke transformation CLARKE and inverse Clarke Transformation iCLARKE are skipped.

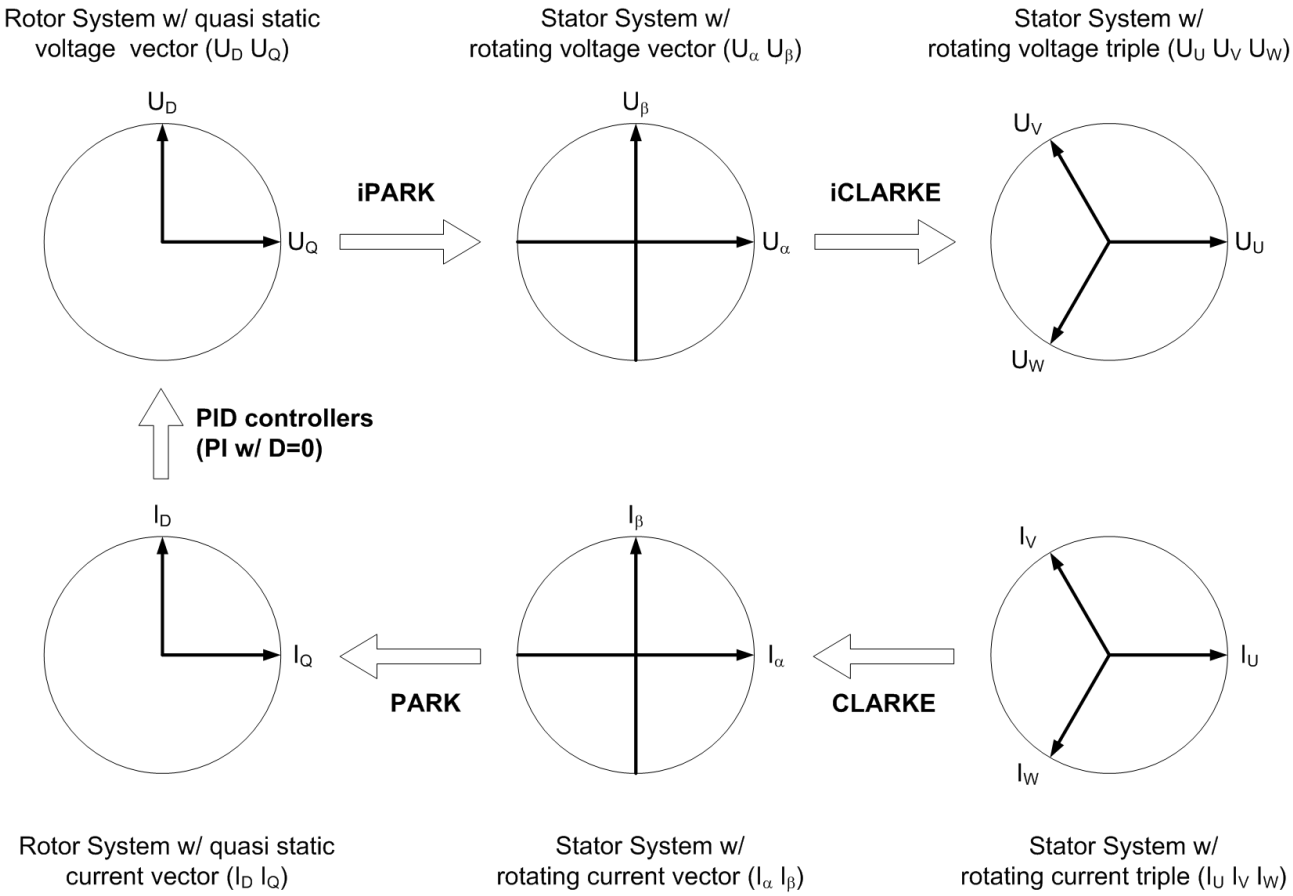


Figure 34: FOC3 Transformations (FOC2 Just Skips CLARKE and iCLARKE)

4.7.11 Motion Modes

It is possible to operate the TMC4671 in several motion modes. Standard motion modes are position control, velocity control and torque control, where target values are fed into the controllers through register access. The motion mode UD_UQ_EXTERN allows to set voltages for open-loop operation and for tests during setup.

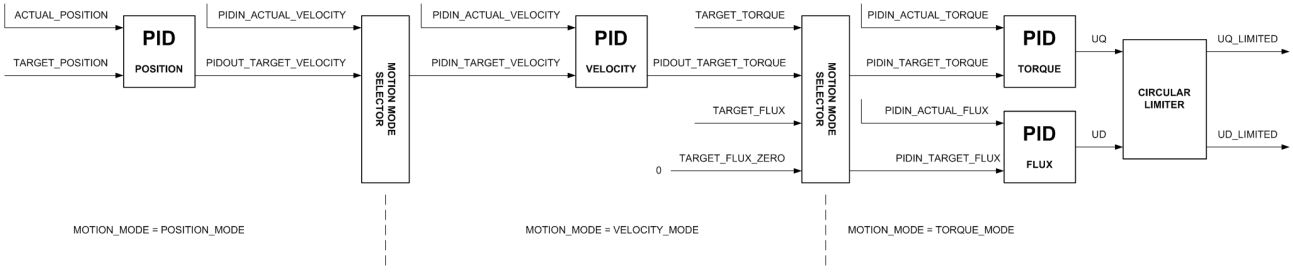


Figure 35: Standard Motion Modes

In position control mode, it is possible to feed the step and direction interface to generate a position target value for the controller cascade. In additional motion modes, target values are fed into the TMC4671 through PWM interface (pin: PWM_IN) or analog input through pin AGPI_A.

There are additional motion modes, which use input from the PWM_I input or the AGPI_A input. Input signals can be scaled through a standard scaler providing offset and gain correction. The interface can be configured through the registers SINGLE_PIN_IF_OFFSET_SCALE and SINGLE_PIN_IF_STATUS_CFG, where the status of the interface can be monitored as well. PWM input signals that are out of frequency range can be neglected. In case of wrong input data, last correct position is used or velocity and torque are set to zero.

Number	Motion Mode	Description
0	Stopped Mode	Disabling all Controllers
1	Torque Mode	Standard Torque Control Mode
2	Velocity Mode	Standard Velocity Control Mode
3	Position Mode	Standard Position Control Mode
4 to 7	Reserved	Reserved
8	UQ UD Ext Mode	Voltage Control Mode (Software Mode)
9	Reserved	Reserved
10	AGPI_A Torque Mode	AGPI_A Used as Target Torque value
11	AGPI_A Velocity Mode	AGPI_A Used as Target Velocity value
12	AGPI_A Position Mode	AGPI_A Used as Target Position value
13	PWM_I Torque Mode	PWM_I Used as Target Torque value
14	PWM_I Velocity Mode	PWM_I Used as Target Velocity value
15	PWM_I Position Mode	PWM_I Used as Target Position value

Table 25: Motion Modes

4.7.12 Brake Chopper

During regenerative braking of the motor, current is driven into the DC link. If the power frontend is not actively controlled, the DC link voltage rises. The brake chopper output pin (BRAKE) can be used for control of an external brake chopper, which burns energy over a brake resistor. The BRAKE pin is set to high for a complete PWM cycle if measured voltage is higher then ADC_VM_LIMIT_HIGH. Once active it is deactivated when voltage drops below ADC_VM_LIMIT_LOW. This acts like a hysteresis. BRAKE can be deactivated by setting both registers to zero. By setting proper values in the registers, it is automatically enabled.

4.8 Filtering and Feed-Forward Control

The TMC4671 uses different filters for certain target and actual values. When using standard velocity meter, a standard velocity filter is used, which is optimized for velocity signals from hall sensors. Additional biquad filters can be used to suppress measurement noise or damp resonances.

4.8.1 Biquad Filters

The TMC4671 uses standard biquad filters (standard IIR filter of second order, Wikipedia Article) in the following structure.

$$\text{Y(n)} = \text{X(n)} \times \text{b\_0} + \text{X(n-1)} \times \text{b\_1} + \text{X(n-2)} \times \text{b\_2} + \text{Y(n-1)} \times \text{a\_1} + \text{Y(n-2)} \times \text{a\_2} \tag{31}$$

In this equation, X(n) is the actual input sample, while Y(n-1) is the filter output of the last cycle. All coefficients are S32 values and are normalized to a Q3.29 format. Take care of correct parametrization of the filter. There is no built-in plausibility or stability check. All filters can be disabled or enabled through register access. Biquad state variables are reset when parameters are changed. The TRINAMIC IDE supports parametrization with wizards.
A standard biquad filter has the following transfer function in the Laplace-Domain:

$$\text{G(s)} = \frac{\text{b\_2\_cont} \times s^2 + \text{b\_1\_cont} \times \text{s} + \text{b\_0\_cont}}{\text{a\_2\_cont} \times s^2 + \text{a\_1\_cont} \times \text{s} + \text{a\_0\_cont}} \tag{32}$$

The transfer function must be transformed to time discrete domain by Z-Transformation and coefficients must be normalized. This is done by the following equations.

$$b\_2\_z = (b\_0\_cont \times T^2 + 2 \times b\_1\_cont \times T + 4 \times b\_2\_cont)/(T^2 - 2 \times a\_1\_cont \times T + 4 \times a\_2\_cont) \tag{33}$$
$$b\_1\_z = (2 \times b\_0\_cont \times T^2 - 8 \times b\_2\_cont)/(T^2 - 2 \times a\_1\_cont \times T + 4 \times a\_2\_cont) \tag{34}$$
$$b\_0\_z = (b\_0\_cont \times T^2 - 2 \times b\_1\_cont \times T + 4 \times b\_2\_cont)/(T^2 - 2 \times a\_1\_cont \times T + 4 \times a\_2\_cont) \tag{35}$$
$$a\_2\_z = (T^2 + 2 \times a\_1\_cont \times T + 4 \times a\_2\_cont)/(T^2 - 2 \times a\_1\_cont \times T + 4 \times a\_2\_cont) \tag{36}$$
$$a\_1\_z = (2 \times T^2 - 8 \times a\_2\_cont)/(T^2 - 2 \times a\_1\_cont \times T + 4 \times a\_2\_cont) \tag{37}$$
$$b\_0 = round(b\_0\_z \times 2^{29}) \tag{38}$$
$$b\_1 = round(b\_1\_z \times 2^{29}) \tag{39}$$
$$b\_2 = round(b\_2\_z \times 2^{29}) \tag{40}$$
$$a\_1 = round(-a\_1\_z \times 2^{29}) \tag{41}$$
$$a\_2 = round(-a\_2\_z \times 2^{29}) \tag{42}$$

while T is the sampling time according to PWM_MAX_COUNT $\times$ 10 ns and variables with index z are auxiliary variables.
A standard second order lowpass filter with given cutoff frequency ω_c and damping factor D has the following transfer function in the Laplace-Domain:

$$G_{LP}(s) = \frac{1}{\frac{1}{\omega_c^2} \times s^2 + \frac{2\,D}{\omega_c} \times s + 1} \tag{43}$$

Determine filter coefficients with the upper equations by comparing coefficients of both transfer functions. The TMCL-IDE also provides a dimensioning tool.

There are four biquad filters in the control structure.Figure 36 illustrates their placement in the control structure.

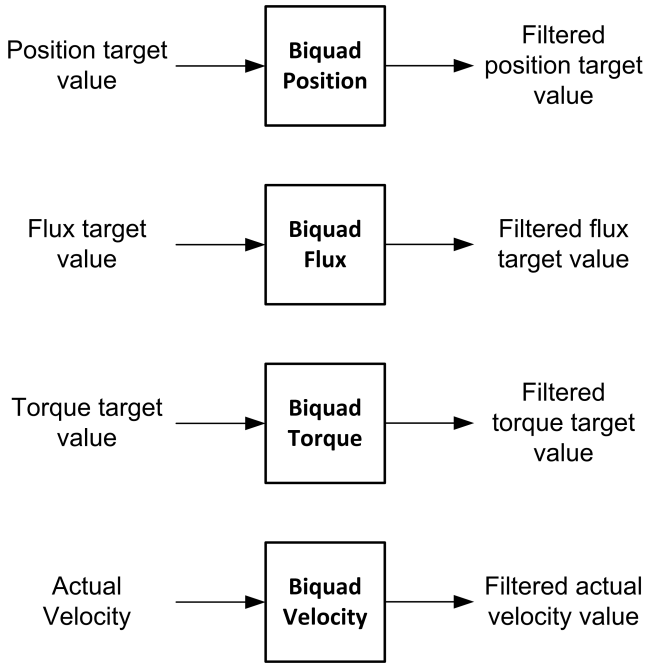


Figure 36: Biquad Filters

The biquad filter for the position target value is intended to be used as a low-pass filter for smoothening position input to the control structure. It is evaluated in every PWM cycle, or down-sampled according to the down-sampling factor for the position controller. After powering on, it is disabled.
The biquad filter for the flux target value is also intended to be used as a low-pass filter for input values from the microcontroller. Sampling frequency is fixed to the PWM frequency.
The biquad filter for the torque target value can be used as a low-pass filter for bandwidth limitation and noise suppression. Moreover, it can be designed to suppress a resonance or anti-resonance. Same statements are correct for the velocity biquad filter. Both filters' sampling times are fixed to the PWM period.
The velocity actual value biquad is configured as a second order low-pass with a cutoff frequency at 200 Hz - by default at a sampling frequency of 25 kHz. This can be used to smoothen the measured velocity, for example, when using digital hall with lower velocities. Biquad filters can be activated separately.

4.8.2 Standard Velocity Filter

By using the standard velocity measurement algorithm, the default velocity filter is enabled and cannot be switched off. The standard velocity filter is a low-pass filter with a cutoff frequency of 20Hz (slope of -20 dB/decade). In this configuration, a new velocity is calculated at a sample rate of approx. 4369.067Hz. This configuration is intended to be used in low-performance applications with a simple position feedback system like digital hall sensors.

4.8.3 Feed-Forward Control Structure

NOTE

Software feed forward control through offset registers is recommended, due to missing amplification possibility. Utilize feedforward to actively increase the target value of a controller besides the normal target input. For torque/flux, use register 0x65 PID_TORQUE_FLUX_OFFSET and for the velocity use register 0x67 PID_VELOCITY_OFFSET.

4.9 PWM Engine

The PWM engine converts voltage vectors to pulse width modulated (PWM) control signals. These digital PWM signals control the gate drivers of the power stage. For a detailed description of the PWM control registers and PWM register control bits, see section 7 page 74.

The ease-of-use PWM engine requires just a couple of parameter settings. Primarily, the polarities for the gate control signal of high-side and low-side must be set. The power on default PWM mode is 0, meaning PWM = OFF. For operation, the centered PWM mode must be switched on by setting the PWM mode to 7. A single bit switches the space vector PWM (SVPWM) on. For 3-phase PMSM, the SVPWM = ON gives more effective voltage. Nevertheless, for some applications it makes sense to switch the SVPWM = OFF to keep the star point voltage of a motor almost at rest.

4.9.1 PWM Polarities

The PWM polarities register (PWM_POLARITIES) controls the polarities of the logic level gate control signals. The polarities of the gate control signals are individually programmable for low-side gate control and for high-side gate control. The PWM polarities register controls the polarity of other control signals as well. PWM_POLARITIES[1] controls the polarity of the logic level high-side gate control signal. PWM_POLARITIES[0] controls the polarity of the logic level low-side gate control signal.

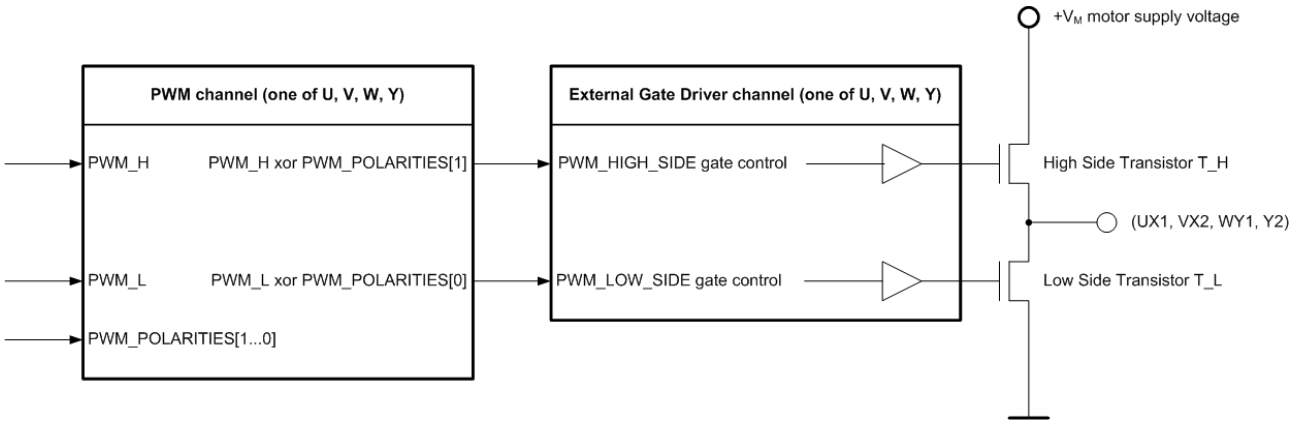


Figure 37: PWM Gate Driver Control Polarities

PWM_POLARITIES[1...0]	PWM_HIGH_SIDE	PWM_LOW_SIDE
0 0	PWM_H	PWM_L
0 1	PWM_H	Not PWM_L

1 0	Not PWM_H	PWM_L
1 1	Not PWM_H	Not PWM_L

Table 26: Status Flags Register

4.9.2 PWM Engine and associated Motor Connectors

The PWM engine of the TMC4671 has eight gate control outputs to control up to four power MOS half bridges. For three-phase motors, three half bridges are used (U, V, W). For two-phase stepper motors, four half bridges are used for (U, V, W, Y). For DC motor control, the first two half bridges (U, V) are used.

Gate Control Signals	FOC3: 3 Phase Motor	FOC2: 2 Phase Stepper	FOC1: Single Phase DC Motor
PWM_UX1_H PWM_UX1_L	U	X1	X1
PWM_VX2_H PWM_VX2_L	V	X2	X2
PWM_WY1_H PWM_WY1_L	W	Y1	-
PWM_Y2_H PWM_Y2_L	-	Y2	-

Table 27: FOC321 Gate Control Signal Configurations

For the DC motor current control (here named FOC1), the number of pole pairs is not relevant - in contrast to closed loop current control of two-phase stepper motors (FOC2) and three-phase permanent magnet motors (FOC3). For DC motor control, the number of pole pairs should be set to 1 to equal mechanical angle and electrical angle for velocity control and for position control.

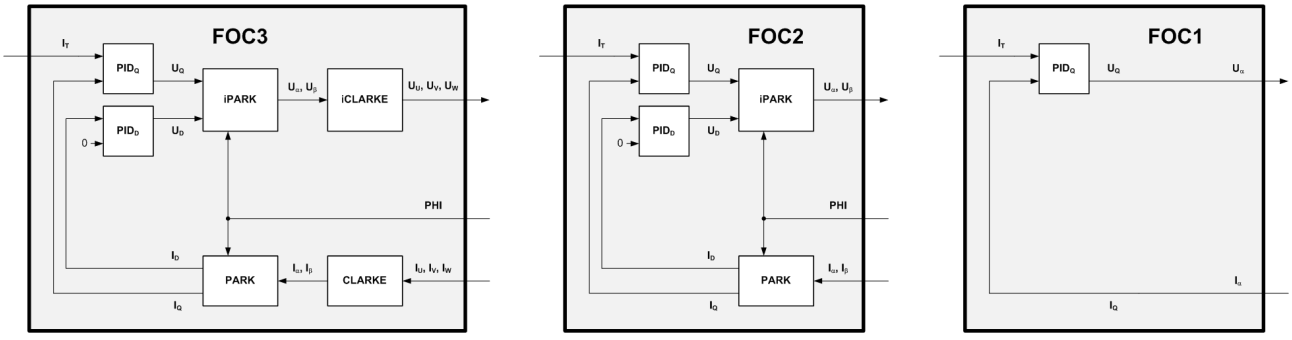


Figure 38: FOC3 (Three-Phase Motor), FOC2 (Two-Phase Stepper Motor), FOC1 (Single-Phase DC Motor)

4.9.3 PWM Frequency

The PWM counter maximum length register PWM_MAXCNT controls the PWM frequency. For a clock frequency fCLK = 25 MHz, the PWM frequency fPWM[Hz] = (4.0 × fCLK[Hz])/(PWM_MAXCNT + 1). With fCLK = 25 MHz and power-on reset (POR) default of PWM_MAXCNT=3999, the PWM frequency fPWM = 25 kHz.

NOTE

The PWM frequency is the fundamental frequency of the control system. It can be changed at any time, also during motion for the classic PI controller structure. The advanced PI controller structure is tied to the PWM frequency and integrator gains must be changed. Make sure to set current measurement decimation rates to fit PWM period in high performance applications.

4.9.4 PWM Resolution

The base resolution of the PWM is 12-bit internally mapped to 16-bit range. The minimal PWM increment is 20ns due to the symmetrical PWM with 100 MHz counter frequency. MAX_PWMCNT = 4095 gives the full resolution of 12-bit with ≈ 25 kHz w/ fCLK = 25 MHz. MAX_PWMCNT = 2047 results in 11-bit resolution, but with ≈ 50kHz w/ fCLK = 25 MHz. So, the PWM_MAXCNT defines the PWM frequency, but also affects the resolution of the PWM.

4.9.5 PWM Modes

The power-on reset (POR) default of the PWM is OFF. The standard PWM scheme is the centered PWM. Passive braking and freewheeling modes are available on demand. See section 7 for the settings.

4.9.6 Break-Before-Make (BBM)

One register controls BBM time for the high-side, another register controls BBM time for the low-side. The BBM times are programmable in 10 ns steps. The BBM time can be set to zero for gate drivers that have their own integrated BBM timers.

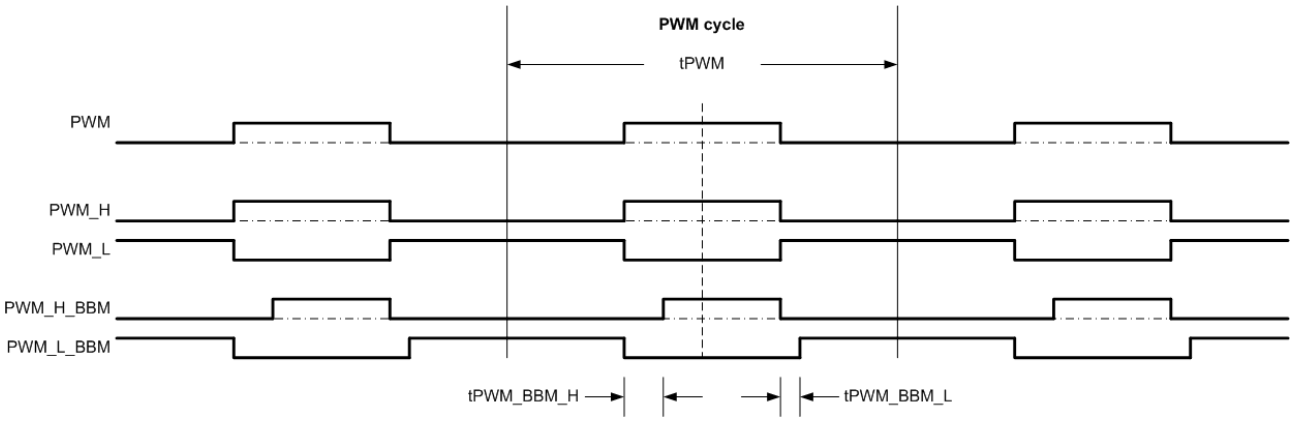


Figure 39: BBM Timing

NOTE Measured BBM times at MOSFET gates differ from programmed BBM times due to driver delays and possible additional gate driver BBM times. The programmed BBM times are for the digital control signals.

NOTE Too short BBM times cause electrical shortcuts of the MOSFET bridges - so called shoot through - that short the power supply and might damage the power stage and power supply.

4.9.7 Space Vector PWM (SVPWM)

A single bit enables the space vector PWM (SVPWM). No further settings are required for the space vector PWM - just ON or OFF. The power on default for the SVPWM is OFF. Space vector PWM can be enabled to maximize voltage utilization in the case of an isolated star point of the motor. If the star point is not isolated, SVPWM might cause unintended current flows through the star point. Space vector PWM is only used for three-phase motors. For other motors, the SVPWM must be switched off.

NOTE For engineering samples TMC4671-ES, the space vector PWM does not allow higher voltage utilization. This is fixed for the release version TMC4671-LA.

4.9.8 Real- and Integer-Conversions

The TMC4671 displays voltages and currents as integer values. The following tables show how to convert integer values to real values (Table 28), and the other way round (Table 29). Equation 14 in section 4.5.1 describes the chain of gains and introduces A_{Peak}/LSB. This variable depends on resistance of the shunt resistor as well as the properties of the sense amplifier. It is needed for the current conversions. The voltage conversion depends on the supply voltage V_M.
Senseamps and their respective shunt resistors can deviate in their properties due to part tolerances or aging. However, their values must still be comparable. This is done by using a scaling factor for both ADCs to harmonize their signals.

$$\mathrm{ADC\_GAIN_{scaled}} = (\mathrm{ADC\_GAIN} \times \frac{\mathrm{ADC\_SCALE}}{256}) \tag{44}$$

	Integer to Real
$I_{uvw,real}$	$\frac{I_{uvw,s16}}{ADC\_GAIN_{scaled}}$
$I_{\alpha\beta,real}$	$\frac{I_{\alpha\beta,s16}}{ADC\_GAIN_{scaled}}$
$I_{dq,real}$	$\frac{I_{dq,s16}}{ADC\_GAIN_{scaled}}$
$U_{dq,real}$	$U_{dq,s16} \times \frac{V_M}{2^{15}}$
$U_{\alpha\beta,real}$	$U_{\alpha\beta,s16} \times \frac{V_M}{2^{15}}$
$FOC_{uvw,real}$	$FOC_{uvw,s16} \times \frac{V_M}{2^{15}}$
$PWM_{uvw,real}$	$PWM_{uvw,s16} \times \frac{V_M}{2^{15}}$

Table 28: Factors for Integer to Real Conversion

	real to integer
$I_{uvw,s16}$	$I_{uvw,real} \times ADC\_GAIN_{scaled}$
$I_{\alpha\beta,s16}$	$I_{\alpha\beta,real} \times ADC\_GAIN_{scaled}$
$I_{dq,s16}$	$I_{dq,real} \times ADC\_GAIN_{scaled}$
$U_{dq,s16}$	$U_{dq,real} \times \frac{2^{15}}{V_M}$
$U_{\alpha\beta,s16}$	$U_{\alpha\beta,real} \times \frac{2^{15}}{V_M}$
$FOC_{uvw,s16}$	$FOC_{uvw,real} \times \frac{2^{15}}{V_M}$
$PWM_{uvw,s16}$	$PWM_{dq,real} \times \frac{2^{15}}{V_M}$

Table 29: Factors for Real to Integer Conversion

The PWM value defines the output voltage. It is calculated using the content of register INTERIM_DATA while INTERIM_ADDR is 0x11 or 0x12. The s16 PWM value is converted to a u16 value by adding 0x8000. Equation 45 applies for the high-side PWM when connected to a DC- or stepper-motor as well as the three phases of a BLDC-motor when space vector PWM is inactive:

$$U_{clamp} = (PWM_{uvw,s16} + 0x8000) \times \frac{V_M}{2^{16}} \tag{45}$$

Equation 46 describes the output voltage on the clamps for the low-side PWM when connected to a DC- or stepper-motor:

$$U_{clamp} = (\text{-}PWM_{uxwy,s16} + 0x8000) \times \frac{V_M}{2^{16}} \tag{46}$$

The following equation describes the integer to real transformation for three-phase space vector PWM:

$$\begin{aligned} FOC_{MIN} &= \min(FOC_u, FOC_v, FOC_w) \\ FOC_{MAX} &= \max(FOC_u, FOC_v, FOC_w) \\ U_{clamp,uvw} &= (\frac{2}{\sqrt{3}} \times (PWM_{uvw,s16} - \frac{FOC_{MAX} + FOC_{MIN}}{2}) + 0x8000) \times \frac{V_M}{2^{16}} \end{aligned} \tag{47}$$

5 Safety Functions

Different safety functions are integrated and mapped to status bits. A programmable mask register selects bits for activation of the STATUS output.

Internal hardware limiters for real time clipping and monitoring of interim values are available. LIMIT or LIMITS is part of register names of registers associated to internal limiters.Table 30 lists the available status flags.

Bit	Source	Info
0	pid_x_target_limit	1: Position controller input clipped (s. 0x61, 0x62)
1	Reserved	-
2	pid_x_errsum_limit	1: Position controller integrator clipped (s. 0x60)
3	pid_x_output_limit	1: Position controller output clipped (s. 0x60)

4	pid_v_target_limit	1: Velocity controller input clipped (s. 0x60)
5	Reserved	-
6	pid_v_errsum_limit	1: Velocity controller integrator clipped (s. 0x5E)
7	pid_v_output_limit	1: Velocity controller output clipped (s. 0x5E)
8	pid_id_target_limit	1: Flux controller target clipped (s. 0x5E)
9	Reserved	-
10	pid_id_errsum_limit	1: Flux controller integrator clipped (s. 0x5D)
11	pid_id_output_limit	1: Flux controller output clipped (s. 0x5D)
12	pid_iq_target_limit	1: Torque controller target clipped (s. 0x5E)
13	Reserved	-
14	pid_iq_errsum_limit	1: Torque controller integrator clipped (s. 0x5D)
15	pid_iq_output_limit	1: Torque controller output clipped (s. 0x5D)
16	ipark_cirlim_limit_u_d	1: ud in circular limiter clipped (s. 0x5D)
17	ipark_cirlim_limit_u_q	1: uq in circular limiter clipped (s. 0x5D)
18	ipark_cirlim_limit_u_r	1: uqud vector clipped (s. 0x5D)
19	Reserved	-
20	ref_sw_r	1: REF_R transition from 0 -> 1 (polarity ignored)
21	ref_sw_h	1: REF_H transition from 0 -> 1 (polarity ignored)
22	ref_sw_l	1: REF_L transition from 0 -> 1 (polarity ignored)
23	Reserved	-
24	pwm_min	PWM duty cycle reached 0%
25	pwm_max	PWM duty cycle reached 100%
26	adc_i_clipped	At least one of the current measurement ADCs clipped
27	adc_aenc_clipped	Analog encoder ADCs clipped
28	ENC_N	An N-pulse is detected on ABN-Encoder interface 1
29	ENC2_N	An N-pulse is detected on ABN-Encoder interface 2
30	AENC_N	An analog-encoder N-pulse was detected
31	Reserved	-

Table 30: Status Flags Register

All controllers have input limiters as offsets can be added to target values and they can be limited to remain in certain ranges. Also, all controller outputs can be limited and the integrating parts (error sum) of the PI controllers are also limited to controller outputs.
If one of these limiters gets active, the flag goes to high state. This is usually a normal operation, when controllers are working on the borders of their working area. With STATUS_MASK register, corresponding flags can be activated.

Other status flags go to high state whether the voltage limitation is reached (circular limiter in iPark transformation) or PWM is saturated (pwm_min and pwm_max). This is also the usual operation as the current controller has to deal with voltage limitation at high velocity operation.
It is also possible to use the status output to generate an IRQ on reference switch or N-channel of encoder.
Also ADC clipping can be monitored, which is a good indicator of wrong or faulty behavior.
Status flags register can be written directly. It is not possible to clear individual bits.

6 FOC Setup - How to Turn a Motor

This section summarizes the basic steps to turn a motor with TMC4671. The wizard of the TMCL-IDE guides through theses basic steps. Schematics and layout of the TMC4671 evaluation kit are open source and available for download from www.trinamic.com

NOTE	TRINAMIC recommends to use a TMC4671 evaluation kit together with the TMCL-IDE with its integrated wizards for initial evaluation and setup.

To create own application software, refer to TRINAMIC's API to reduce software development efforts.

6.1 Select Motor Type

The TMC4671 supports closed loop control of single-phase DC motors, stepper motors, and three-phase motors. The selection of the motor type defines the configuration of the gate control channels for the power stage and either the usage or bypass of FOC transformations (Clarke, Park, iPark, iClark).

6.1.1 FOC1 Setup - How to Turn a Single-Phase DC Motor

For the DC motor, the mechanical commutator of the DC motor realizes something like mechanical field oriented control, where the TMC4671 just realizes closed loop current control of the DC motor. From FOC point of view, the FOC converts a brushless motor (BLDC) resp. permanent magnet synchronous motor (PMSM) into a closed loop current controlled DC motor.

From closed loop velocity control point of view and from closed loop position control point of view, there is no difference between electronically FOC controlled BLDC motor or PMSM motor and a mechanical commutated DC motor with electronic closed loop current control.

6.1.2 FOC2 Setup - How to Turn a Two-Phase Motor (Stepper)

The TMC4671 is able to turn a two-phase stepper motor with FOC by internal skip of Clarke transformation and iClarke transformation. A special feature of stepper motors is the high number of pole pairs (NPP) that are typical 50. For stepper motors, it is usual to give the number of full steps (FS) per revolution, with NPP = (FS/revolution)/4. A stepper with 200 full steps per revolution has 50 pole pairs.

6.1.3 FOC3 Setup - How to Turn a Three-Phase Motor (PMSM or BLDC)

A three-phase motor is the classical FOC controlled brushless motor. Take care on the number of pole pairs (NPP) and the number of poles (NP) with NPP = NP/2.

6.2 Set Number of Pole Pairs (NPP)

The number of (magnetic) pole pairs (NPP) is characteristic for each motor and it is essential for commutation of two-phase motors and three-phase motors with FOC. For DC motor, the NPP is not important for commutation itself, but should be set to one to have the same scaling for electrical angle and mechanical angle.

6.3 Run Motor Open Loop

Initial turning of a motor open loop is useful to determine the association between phase voltage, phase currents, and for position sensor setup. Position sensors mounted on a motor might have an opposite direction of rotation compared to the motor. The same direction of rotation is essential for the FOC. In addition, the phase shift between the rotor angle and angle measured by a position sensor must be zero in the best case. Otherwise, the motor is operated at lower efficiency or turns in wrong direction, which causes instability.

6.3.1 Determination of Association Between Phase Voltage and Phase Currents

For starters, the motor should be turned open loop to measure ADC offsets and set ADC scaler offset. Additionally, the open loop turn is useful to validate (or to determine) the association between motor phase currents and motor phase terminal voltages. This association is essential for the FOC. With proper ADC channel selection setup, voltage U_UX1 is in phase with current I_UX1, voltage U_VX2 is in phase with current I_VX2, and voltage U_WY1 is in phase with I_WY1. For two-phase stepper motor, the voltage U_Y2 is in phase with current I_Y2. Only two currents are measured and the other current is calculated by TMC4671. For DC motor, only one current is measured.

6.3.2 Determination of Direction of Rotation and Phase Shift of Angles

For absolute position sensors like hall sensors, the phase shift and the direction of rotation only must be determined once initially. For relative position sensors, like incremental encoders, the direction of turning must be determined every time after the power cycle. The relative orientation between measured incremental encoder angle and rotor angle must be determined on each power-up.

6.4 Selection of Position Sensors

For closed loop operation, the type of encoder (digital hall, ABN encoder, analog hall, SinCos) must be set. For analog hall signals or analog incremental encoders, adjust the analog ADC channels for the analog encoders - similar to ADC offset and ADC scaling as for current measuring ADC channels. The TMC4671 allows the selection of different types of position sensors for different tasks. One position sensor is for the inner FOC closed loop current control loop.

6.4.1 Selection of FOC Sensor for PHI_E

Select one sensor for the FOC to measure the electrical angle PHI_E. This sensor is used for the inner closed loop control loop for closed loop current control.

6.4.2 Selection of Sensor for VELOCITY

Select one sensor for measurement of velocity. This can be the sensor selected for measurement of PHI_E but it is more common to use the mechanical angle PHI_M for measurement of velocity. Using electrical angles can give advantages for applications with slow motion for NPP more than one because the minimum velocity in revolutions per minute (RPM) is one and the electrical angles have higher speed than mechanical angles.

6.4.3 Selection of sensor for POSITION

Select one sensor for measurement of position of the rotor, the angle of the rotor. This can be the sensor selected for measurement PHI_E but it is more usual to use the mechanical angle PHI_M for measurement of position. For stepper motors, it might make sense to select the electrical angle PHI_E for positioning

to have a benefit from higher resolution using electrical angles. This is because each period (electrical or mechanical) is normalized to 2^{16} = 65536 positions.

6.5 Modes of Operation - (Open Loop), Torque, Velocity, Positioning

The TMC4671 can operate in torque mode, velocity mode, or position mode. The control loops (current, velocity, position) are cascaded. Thus the outer loops depend on the tuning of the inner loops. So, the current loop must be adjusted first. The velocity loop must be adjusted after the current control loop is adjusted. The position control loop must be adjusted last.

6.6 Controller Tuning

PI controller tuning is described throughout the control theory literature. In general, there are two main strategies to tune the controllers. First strategy is to observe the controller step response for different parameter sets and tune the parameters to fit dynamics and settling time. With this approach, sampling target and actual value as well as controller output (check for saturation) at fixed frequency is recommended. The USB-2-RTMI adapter in combination with the TMCL-IDE provide tuning tools to support this strategy.
Another approach is to identify controller plant parameters and calculate controller parameters from these parameters. This is also supported by the TMCL-IDE for the current control loop. For the other control loops, the first strategy is recommended.

7 Register Map

The TMC4671 has a register address range of 128 addresses with registers up to 32-bit data width. Some registers hold 32-bit data fields, some hold 2 x 16-bit data fields and other hold combinations of different data fields with individual data types. Data fields must be extracted by masking and shifting after read from a TMC4671 register within the application. Data fields must be composed by masking and shifting by the application before writing into a TMC4671 register. Refer to TRINAMIC's API to reduce software development efforts. This section describes the register bank of the TMC4671.

Section 7.1 gives an overview of all registers. It is is intended to give an initial over view of all registers.

Section 7.2 is the detailed reference of all registers and the register fields.

Section 7.3 gives the description of power-on-reset default values of all registers.

7.1 Register Map - Overview

Registers in TMC4671 have different purposes. Some registers are used for test only, other can be used to monitor internal states (example, ADC values). Most registers are only accessed during initialization (example, calibration or control parameters). Control registers are used for the input of target values to controllers and should be updated regularly according to chosen motion modes (example, PID_VELOCITY_TARGET should be updated in velocity mode). If not using a certain functional block, no need to parametrize it.

The TMC4671 has an address space of 128 addresses. To display more then 128 registers, so called stacked registers are added. These are CHIPINFO_DATA, ADC_RAW_DATA, PID_ERROR_DATA, CONFIG_DATA, and INTERIM_DATA. These data registers display or give access to different subregisters according to their corresponding address registers (CHIPINFO_ADDR, ADC_RAW_ADDR, PID_ERROR_ADDR, CONFIG_ADDR, and INTERIM_ADDR). Read access to a subregister requires a write access to the address register and a read access to the data register.

Address	Register Name	Access	Usage
$0x00_h$	CHIPINFO_DATA	R	Test
$0x01_h$	CHIPINFO_ADDR	RW	Test
$0x02_h$	ADC_RAW_DATA	R	Monitor
$0x03_h$	ADC_RAW_ADDR	RW	Monitor
$0x04_h$	dsADC_MCFG_B_MCFG_A	RW	Init
$0x05_h$	dsADC_MCLK_A	RW	Init
$0x06_h$	dsADC_MCLK_B	RW	Init
$0x07_h$	dsADC_MDEC_B_MDEC_A	RW	Init
$0x08_h$	ADC_I1_SCALE_OFFSET	RW	Init
$0x09_h$	ADC_I0_SCALE_OFFSET	RW	Init
$0x0A_h$	ADC_I_SELECT	RW	Init
$0x0B_h$	ADC_I1_I0_EXT	RW	Test
$0x0C_h$	DS_ANALOG_INPUT_STAGE_CFG	RW	Test
$0x0D_h$	AENC_0_SCALE_OFFSET	RW	Init
$0x0E_h$	AENC_1_SCALE_OFFSET	RW	Init
$0x0F_h$	AENC_2_SCALE_OFFSET	RW	Init
$0x11_h$	AENC_SELECT	RW	Init
$0x12_h$	ADC_IWY_IUX	R	Monitor
$0x13_h$	ADC_IV	R	Monitor
$0x15_h$	AENC_WY_UX	R	Monitor
$0x16_h$	AENC_VN	R	Monitor
$0x17_h$	PWM_POLARITIES	RW	Init
$0x18_h$	PWM_MAXCNT	RW	Init
$0x19_h$	PWM_BBM_H_BBM_L	RW	Init
$0x1A_h$	PWM_SV_CHOP	RW	Init

Address	Register Name	Access	Usage
$0x1B_h$	MOTOR_TYPE_N_POLE_PAIRS	RW	Init
$0x1C_h$	PHI_E_EXT	RW	Test
$0x1F_h$	OPENLOOP_MODE	RW	Init
$0x20_h$	OPENLOOP_ACCELERATION	RW	Init
$0x21_h$	OPENLOOP_VELOCITY_TARGET	RW	Init
$0x22_h$	OPENLOOP_VELOCITY_ACTUAL	RW	Monitor
$0x23_h$	OPENLOOP_PHI	RW	Monitor/Test
$0x24_h$	UQ_UD_EXT	RW	Init/Test
$0x25_h$	ABN_DECODER_MODE	RW	Init
$0x26_h$	ABN_DECODER_PPR	RW	Init
$0x27_h$	ABN_DECODER_COUNT	RW	Init/Test/Monitor
$0x28_h$	ABN_DECODER_COUNT_N	RW	Init/Test/Monitor
$0x29_h$	ABN_DECODER_PHI_E_PHI_M_OFFSET	RW	Init
$0x2A_h$	ABN_DECODER_PHI_E_PHI_M	R	Monitor
$0x2C_h$	ABN_2_DECODER_MODE	RW	Init
$0x2D_h$	ABN_2_DECODER_PPR	RW	Init
$0x2E_h$	ABN_2_DECODER_COUNT	RW	Init/Test/Monitor
$0x2F_h$	ABN_2_DECODER_COUNT_N	RW	Init/Test/Monitor
$0x30_h$	ABN_2_DECODER_PHI_M_OFFSET	RW	Init
$0x31_h$	ABN_2_DECODER_PHI_M	R	Monitor
$0x33_h$	HALL_MODE	RW	Init
$0x34_h$	HALL_POSITION_060_000	RW	Init
$0x35_h$	HALL_POSITION_180_120	RW	Init
$0x36_h$	HALL_POSITION_300_240	RW	Init
$0x37_h$	HALL_PHI_E_PHI_M_OFFSET	RW	Init
$0x38_h$	HALL_DPHI_MAX	RW	Init
$0x39_h$	HALL_PHI_E_INTERPOLATED_PHI_E	R	Monitor
$0x3A_h$	HALL_PHI_M	R	Monitor
$0x3B_h$	AENC_DECODER_MODE	RW	Init
$0x3C_h$	AENC_DECODER_N_THRESHOLD	RW	Init
$0x3D_h$	AENC_DECODER_PHI_A_RAW	R	Monitor
$0x3E_h$	AENC_DECODER_PHI_A_OFFSET	RW	Init
$0x3F_h$	AENC_DECODER_PHI_A	R	Monitor
$0x40_h$	AENC_DECODER_PPR	RW	Init

Address	Register Name	Access	Usage
$0x41_h$	AENC_DECODER_COUNT	RW	Monitor
$0x42_h$	AENC_DECODER_COUNT_N	RW	Monitor/Init
$0x45_h$	AENC_DECODER_PHI_E_PHI_M_OFFSET	RW	Init
$0x46_h$	AENC_DECODER_PHI_E_PHI_M	R	Monitor
$0x4D_h$	CONFIG_DATA	RW	Init
$0x4E_h$	CONFIG_ADDR	RW	Init
$0x50_h$	VELOCITY_SELECTION	RW	Init
$0x51_h$	POSITION_SELECTION	RW	Init
$0x52_h$	PHI_E_SELECTION	RW	Init
$0x53_h$	PHI_E	R	Monitor
$0x54_h$	PID_FLUX_P_FLUX_I	RW	Init
$0x56_h$	PID_TORQUE_P_TORQUE_I	RW	Init
$0x58_h$	PID_VELOCITY_P_VELOCITY_I	RW	Init
$0x5A_h$	PID_POSITION_P_POSITION_I	RW	Init
$0x5D_h$	PIDOUT_UQ_UD_LIMITS	RW	Init
$0x5E_h$	PID_TORQUE_FLUX_LIMITS	RW	Init
$0x60_h$	PID_VELOCITY_LIMIT	RW	Init
$0x61_h$	PID_POSITION_LIMIT_LOW	RW	Init
$0x62_h$	PID_POSITION_LIMIT_HIGH	RW	Init
$0x63_h$	MODE_RAMP_MODE_MOTION	RW	Init
$0x64_h$	PID_TORQUE_FLUX_TARGET	RW	Control
$0x65_h$	PID_TORQUE_FLUX_OFFSET	RW	Control
$0x66_h$	PID_VELOCITY_TARGET	RW	Control
$0x67_h$	PID_VELOCITY_OFFSET	RW	Control
$0x68_h$	PID_POSITION_TARGET	RW	Control
$0x69_h$	PID_TORQUE_FLUX_ACTUAL	R	Monitor
$0x6A_h$	PID_VELOCITY_ACTUAL	R	Monitor
$0x6B_h$	PID_POSITION_ACTUAL	RW	Monitor/Init
$0x6C_h$	PID_ERROR_DATA	R	Test
$0x6D_h$	PID_ERROR_ADDR	RW	Test
$0x6E_h$	INTERIM_DATA	RW	Monitor
$0x6F_h$	INTERIM_ADDR	RW	Monitor
$0x75_h$	ADC_VM_LIMITS	RW	Init
$0x76_h$	TMC4671_INPUTS_RAW	R	Test/Monitor

Address	Register Name	Access	Usage
$0x77_h$	TMC4671_OUTPUTS_RAW	R	Test/Monitor
$0x78_h$	STEP_WIDTH	RW	Init
$0x79_h$	UART_BPS	RW	Init
$0x7B_h$	GPIO_dsADCI_CONFIG	RW	Init
$0x7C_h$	STATUS_FLAGS	RW	Monitor
$0x7D_h$	STATUS_MASK	RW	Monitor

Table 31: TMC4671 Registers

7.2 Register Map - Functional Description

ADDR	NAME	DATA TYPE (BIT MASK)	FUNCTION
$0x00_h$	CHIPINFO_DATA		This register displays name and version information of the accessed IC. It can be used for test of communication.
	SI_TYPE	u32(31:0)	0: Hardware type (ASCII)
	SI_VERSION	u32(31:0)	0: Hardware version (u16.u16)
	SI_DATE	u32(31:0)	0: Hardware date (nibble wise date stamp yyyymmdd)
	SI_TIME	u32(31:0)	0: Hardware time (nibble wise time stamp –hhmmss)
	SI_VARIANT	u32(31:0)	
	SI_BUILD	u32(31:0)	
$0x01_h$	CHIPINFO_ADDR		This register is used to change the displayed information in register CHIPINFO_DATA.
	CHIP_INFO_ADDRESS	u8(7:0)	0: SI_TYPE 1: SI_VERSION 2: SI_DATE 3: SI_TIME 4: SI_VARIANT 5: SI_BUILD
$0x02_h$	ADC_RAW_DATA		This register displays ADC values. The displayed registers can be switched by register ADC_RAW_ADDR.
	ADC_I0_RAW	u16(15:0)	Raw phase current I0
	ADC_I1_RAW	u16(31:16)	Raw phase current I1
	ADC_VM_RAW	u16(15:0)	Raw supply voltage value.
	ADC_AGPI_A_RAW	u16(31:16)	Raw analog GPI A value
	ADC_AGPI_B_RAW	u16(15:0)	Raw analog GPI B value
	ADC_AENC_UX_RAW	u16(31:16)	Raw analog encoder signal
	ADC_AENC_VN_RAW	u16(15:0)	Raw analog encoder signal
	ADC_AENC_WY_RAW	u16(31:16)	Raw analog encoder signal
$0x03_h$	ADC_RAW_ADDR		This register is used to change the displayed information in register ADC_RAW_DATA.

	ADC_RAW_ADDR	u8(7:0)	0: ADC_I1_RAW & ADC_I0_RAW 1: ADC_AGPI_A_RAW & ADC_VM_RAW 2: ADC_AENC_UX_RAW & ADC_AGPI_B_RAW 3: ADC_AENC_WY_RAW & ADC_AENC_VN_RAW
$0x04_h$	dsADC_MCFG_B_MCFG_A		This register is used to configure internal ADCs (delta sigma modulators). Do not modify when using internal Delta Sigma modulators (standard use case).
	cfg_dsmodulator_a	u2(1:0)	0: int. dsMOD 1: ext. MCLK input 2: ext. MCLK output 3: ext. CMP
	mclk_polarity_a	bit(2)	0: off 1: on
	mdat_polarity_a	bit(3)	0: off 1: on
	sel_nclk_mclk_i_a	bit(4)	0: off 1: on
	cfg_dsmodulator_b	u2(17:16)	0: int. dsMOD 1: ext. MCLK input 2: ext. MCLK output 3: ext. CMP
	mclk_polarity_b	bit(18)	0: off 1: on
	mdat_polarity_b	bit(19)	0: off 1: on
	sel_nclk_mclk_i_b	bit(20)	0: off 1: on
$0x05_h$	dsADC_MCLK_A		This register is used to modify Delta Sigma modulator clock. Do not modify when using internal delta sigma modulators (standard use case).
	dsADC_MCLK_A	u32(31:0)	dsADC_MCLK_A = $\frac{(2^{31} \times f_{MCLK})}{f_{CLK}}$

0x06$_h$	dsADC_MCLK_B		This register is used to modify Delta Sigma modulator clock. Do not modify when using internal delta sigma modulators (standard use case).
	dsADC_MCLK_B	u32(31:0)	dsADC_MCLK_B = $\frac{(2^{31} \times f_{MCLK})}{f_{CLK}}$
0x07$_h$	dsADC_MDEC_B_MDEC_A		This register is used to change decimation rates of SINC3 filters for Delta Sigma modulators. Set values according to actual PWM frequency. See functional description of ADC engine 1.
	dsADC_MDEC_A	u16(15:0)	0: PWM synchronous, others according to register content
	dsADC_MDEC_B	u16(31:16)	0: PWM synchronous, others according to register content
0x08$_h$	ADC_I1_SCALE_OFFSET		This register is used to set calibration data for ADC channel I1 (offset and amplitude correction).
	ADC_I1_OFFSET	u16(15:0)	Offset for current ADC channel 1.
	ADC_I1_SCALE	s16(31:16)	Scaling factor for current ADC channel 1.
0x09$_h$	ADC_I0_SCALE_OFFSET		This register is used to set calibration data for ADC channel I0 (offset and amplitude correction).
	ADC_I0_OFFSET	u16(15:0)	Offset for current ADC channel 0.
	ADC_I0_SCALE	s16(31:16)	Scaling factor for current ADC channel 0.
0x0A$_h$	ADC_I_SELECT		This register is used to assign correct ADC channel to PWM output channel. For each FOC input current, either an ADC value or the calculated sum of the currents (I2) can be assigned to match internal data processing to power stage design.
	ADC_I0_SELECT	u8(7:0)	Select input for raw current ADC_I0_RAW. 0: ADCSD_I0_RAW (sigma delta ADC) 1: ADCSD_I1_RAW (sigma delta ADC) 2: ADC_I0_EXT (from register) 3: ADC_I1_EXT (from register)

	ADC_I1_SELECT	u8(15:8)	Select input for raw current ADC_I1_RAW. 0: ADCSD_I0_RAW (sigma delta ADC) 1: ADCSD_I1_RAW (sigma delta ADC) 2: ADC_I0_EXT (from register) 3: ADC_I1_EXT (from register)
	ADC_I_UX_SELECT	u2(25:24)	0: UX = ADC_I0 (default) 1: UX = ADC_I1 2: UX = ADC_I2
	ADC_I_V_SELECT	u2(27:26)	0: V = ADC_I0 1: V = ADC_I1 (default) 2: V = ADC_I2
	ADC_I_WY_SELECT	u2(29:28)	0: WY = ADC_I0 1: WY = ADC_I1 2: WY = ADC_I2 (default)
$0x0B_h$	ADC_I1_I0_EXT		This register can be used to write ADC values through SPI in case external ADCs are used or controller cascade function are tested. Using external ADCs probably effect control performance and are not recommended.
	ADC_I0_EXT	u16(15:0)	Register for write of ADC_I0 value from external source (eg. CPU).
	ADC_I1_EXT	u16(31:16)	Register for write of ADC_I1 value from external source (eg. CPU).
$0x0C_h$	DS_ANALOG_INPUT_STAGE_CFG		This register is used to configure ADC channels for different input configurations and test modes.
	ADC_I0	u4(3:0)	0: INP vs. INN 1: GND vs. INN 2: $V_{DD}/4$ 3: $3\times V_{DD}/4$ 4: INP vs. GND 5: $V_{DD}/2$ 6: $V_{DD}/4$ 7: $3\times V_{DD}/4$
	ADC_I1	u4(7:4)	0: INP vs. INN

		1: GND vs. INN
		2: $V_{DD}/4$
		3: $3\times V_{DD}/4$
		4: INP vs. GND
		5: $V_{DD}/2$
		6: $V_{DD}/4$
		7: $3\times V_{DD}/4$
ADC_VM	u4(11:8)	0: INP vs. INN
		1: GND vs. INN
		2: $V_{DD}/4$
		3: $3\times V_{DD}/4$
		4: INP vs. GND
		5: $V_{DD}/2$
		6: $V_{DD}/4$
		7: $3\times V_{DD}/4$
ADC_AGPI_A	u4(15:12)	0: INP vs. INN
		1: GND vs. INN
		2: $V_{DD}/4$
		3: $3\times V_{DD}/4$
		4: INP vs. GND
		5: $V_{DD}/2$
		6: $V_{DD}/4$
		7: $3\times V_{DD}/4$
ADC_AGPI_B	u4(19:16)	0: INP vs. INN
		1: GND vs. INN
		2: $V_{DD}/4$
		3: $3\times V_{DD}/4$
		4: INP vs. GND
		5: $V_{DD}/2$
		6: $V_{DD}/4$
		7: $3\times V_{DD}/4$
ADC_AENC_UX	u4(23:20)	0: INP vs. INN
		1: GND vs. INN
		2: $V_{DD}/4$
		3: $3\times V_{DD}/4$
		4: INP vs. GND

			5: $V_{DD}/2$ 6: $V_{DD}/4$ 7: $3\times V_{DD}/4$
	ADC_AENC_VN	u4(27:24)	0: INP vs. INN 1: GND vs. INN 2: $V_{DD}/4$ 3: $3\times V_{DD}/4$ 4: INP vs. GND 5: $V_{DD}/2$ 6: $V_{DD}/4$ 7: $3\times V_{DD}/4$
	ADC_AENC_WY	u4(31:28)	0: INP vs. INN 1: GND vs. INN 2: $V_{DD}/4$ 3: $3\times V_{DD}/4$ 4: INP vs. GND 5: $V_{DD}/2$ 6: $V_{DD}/4$ 7: $3\times V_{DD}/4$
$0x0D_h$	AENC_0_SCALE_OFFSET		This register is used to set calibration data for ADC channel AENC 0 (Offset and amplitude correction).
	AENC_0_OFFSET	u16(15:0)	Offset for analog encoder ADC channel 0.
	AENC_0_SCALE	s16(31:16)	Scaling factor for analog encoder ADC channel 0.
$0x0E_h$	AENC_1_SCALE_OFFSET		This register is used to set calibration data for ADC channel AENC 1 (Offset and amplitude correction).
	AENC_1_OFFSET	u16(15:0)	Offset for analog encoder ADC channel 1.
	AENC_1_SCALE	s16(31:16)	Scaling factor for analog encoder ADC channel 1.
$0x0F_h$	AENC_2_SCALE_OFFSET		This register is used to set calibration data for ADC channel AENC 2 (Offset and amplitude correction).
	AENC_2_OFFSET	u16(15:0)	Offset for analog encoder ADC channel 2.
	AENC_2_SCALE	s16(31:16)	Scaling factor for analog encoder ADC channel 2.

$0x11_h$	AENC_SELECT		This register is used to select correct ADC to compensate wiring twists.
	AENC_0_SELECT	u8(7:0)	Select analog encoder ADC channel for raw analog encoder signal AENC_0_RAW. 0: <AENC_UX_RAW> 1: AENC_VN_RAW 2: AENC_WY_RAW
	AENC_1_SELECT	u8(15:8)	Select analog encoder ADC channel for raw analog encoder signal AENC_1_RAW. 0: AENC_UX_RAW 1: <AENC_VN_RAW> 2: AENC_WY_RAW
	AENC_2_SELECT	u8(23:16)	Select analog encoder ADC channel for raw analog encoder signal AENC_2_RAW. 0: AENC_UX_RAW 1: AENC_VN_RAW 2: <AENC_WY_RAW>
$0x12_h$	ADC_IWY_IUX		This register can be used to monitor phase current values (offset-compensated, scaled and correctly assigned).
	ADC_IUX	s16(15:0)	Register of scaled current ADC value including signed added offset as input for the FOC.
	ADC_IWY	s16(31:16)	Register of scaled current ADC value including signed added offset as input for the FOC.
$0x13_h$	ADC_IV		This register can be used to monitor phase current ADC_IV (offset-compensated, scaled and correctly assigned).
	ADC_IV	s16(15:0)	Register of scaled current ADC value including signed added offset as input for the FOC.
$0x15_h$	AENC_WY_UX		This register displays AENC input signals (offset-compensated, scaled and correctly assigned).

	AENC_UX	s16(15:0)	Register of scaled analog encoder value including signed added offset as input for the interpolator.
	AENC_WY	s16(31:16)	Register of scaled analog encoder value including signed added offset as input for the interpolator.
0x16$_h$	AENC_VN		This register displays AENC input signal AENC_VN (offset-compensated, scaled and correctly assigned).
	AENC_VN	s16(15:0)	Register of scaled analog encoder value including signed added offset as input for the interpolator.
0x17$_h$	PWM_POLARITIES		This register sets the polarity of PWM output signal to match gate driver.
	PWM_POLARITIES[0]	bit(0)	Low-side gate control 0: off 1: on
	PWM_POLARITIES[1]	bit(1)	High-side gate control 0: off 1: on
0x18$_h$	PWM_MAXCNT		This register is used to configure PWM output frequency.
	PWM_MAXCNT	u12(11:0)	PWM maximum (count-1), PWM frequency is fPWM[Hz] = 100MHz/(PWM_MAXCNT+1)
0x19$_h$	PWM_BBM_H_BBM_L		This register sets the BBM times for PWM output signals. BBM time must be matched and power stage needs to avoid cross conduction in half bridge.
	PWM_BBM_L	u8(7:0)	Break before make time tBBM_L[10ns] for low-side MOS-FET gate control
	PWM_BBM_H	u8(15:8)	Break before make time tBBM_H[10ns] for high-side MOS-FET gate control
0x1A$_h$	PWM_SV_CHOP		This register is used to enable PWM, set different PWM test modes, and switch on the SVPWM feature for higher voltage utilization (BLDC/PMSM only).
	PWM_CHOP	u8(7:0)	PWM chopper mode, defining how to chopper

			0: off, free running 1: off, low side permanent = ON 2: off, high side permanent = ON 3: off, free running 4: off, free running 5: low side chopper, high side off 6: high side chopper, low side off 7: centered PWM for FOC
	PWM_SV	bit(8)	Use space vector PWM 0: Space vector PWM disabled 1: Space vector PWM enabled
$0x1B_h$	MOTOR_TYPE_N_POLE_PAIRS		This register is used to set motor type and number of pole pairs.
	N_POLE_PAIRS	u16(15:0)	Number n of pole pairs of the motor for calcualtion phi_e = phi_m × N_POLE_PAIRS.
	MOTOR_TYPE	u8(23:16)	0: No motor 1: Single-phase DC 2: Two-phase stepper 3: Three-phase BLDC
$0x1C_h$	PHI_E_EXT		This register is used to set an electrical angle for SW mode when encoder is connected to MCU and not to TMC4671.
	PHI_E_EXT	s16(15:0)	Electrical angle phi_e_ext for external writing into this register.
$0x1F_h$	OPENLOOP_MODE		This register is used to change direction of openloop angle.
	OPENLOOP_PHI_DIRECTION	bit(12)	Open loop phi direction. 0: positive 1: negative
$0x20_h$	OPENLOOP_ACCELERATION		This register is used to change acceleration when openloop angle velocity should change.
	OPENLOOP_ACCELERATION	u32(31:0)	Acceleration of open loop phi.
$0x21_h$	OPENLOOP_VELOCITY_TARGET		This register is used to set a target velocity for openloop angle generator. The velocity is ramped up and down according to OPENLOOP_ACCELERATION and PID_VELOCITY_LIMIT.

	OPENLOOP_VELOCITY_TARGET	s32(31:0)	Target velocity of open loop phi.
0x22$_h$	OPENLOOP_VELOCITY_ACTUAL		This register displays actual open-loop angle velocity in RPM.
	OPENLOOP_VELOCITY_ACTUAL	s32(31:0)	Actual velocity of open loop generator.
0x23$_h$	OPENLOOP_PHI		This register displays actual output of openloop angle generator
	OPENLOOP_PHI	s16(15:0)	Angle phi open loop (either mapped to electrical angel phi_e or mechanical angle phi_m).
0x24$_h$	UQ_UD_EXT		This register is used to set voltage values for openllop current control mode (UQ_UD_EXT_MODE).
	UD_EXT	s16(15:0)	External writable parameter for open loop voltage control mode, useful during system setup, U_D component.
	UQ_EXT	s16(31:16)	External writable parameter for open loop voltage control mode, useful during system setup, U_Q component.
0x25$_h$	ABN_DECODER_MODE		This register is used to configure decoder input signals and N pulse action as well as count direction.
	apol	bit(0)	Polarity of A pulse. 0: off 1: on
	bpol	bit(1)	Polarity of B pulse. 0: off 1: on
	npol	bit(2)	Polarity of N pulse. 0: off 1: on
	use_abn_as_n	bit(3)	N and A and B 0: Ignore A and B polarity with Npulse = N 1: Npulse = N and A and B
	cln	bit(8)	Write direction at Npulse event between ABN_DECODER_COUNT_N and ABN_DECODER_COUNT. 0: COUNT => COUNT_N 1: COUNT_N => COUNT

	direction	bit(12)	Decoder count direction. 0: positive 1: negative
0x26$_h$	ABN_DECODER_PPR		This register is used to set PPR number of encoder.
	ABN_DECODER_PPR	u24(23:0)	Decoder pulses per mechanical revolution.
0x27$_h$	ABN_DECODER_COUNT		This register displays the actual count of encoder steps. It can be overwritten for initialization.
	ABN_DECODER_COUNT	u24(23:0)	Raw decoder count; the digital decoder engine counts modulo (decoder_ppr).
0x28$_h$	ABN_DECODER_COUNT_N		This register displays the count value at last N pulse event. It can also be used to overwrite decoder count at N pulse event according to decoder mode register setting.
	ABN_DECODER_COUNT_N	u24(23:0)	Decoder count latched on N pulse, when N pulse clears decoder_count also decoder_count_n is 0.
0x29$_h$	ABN_DECODER_PHI_E_PHI_M_OFFSET		This register can be used to set offsets for electrical and mechanical angle calculated from decoder.
	ABN_DECODER_PHI_M_OFFSET	s16(15:0)	ABN_DECODER_PHI_M_OFFSET to shift (rotate) angle DECODER_PHI_M.
	ABN_DECODER_PHI_E_OFFSET	s16(31:16)	ABN_DECODER_PHI_E_OFFSET to shift (rotate) angle DECODER_PHI_E.
0x2A$_h$	ABN_DECODER_PHI_E_PHI_M		This register displays actual angle values for ABN encoder.
	ABN_DECODER_PHI_M	s16(15:0)	ABN_DECODER_PHI_M = ABN_DECODER_COUNT × 2^16 / ABN_DECODER_PPR + ABN_DECODER_PHI_M_OFFSET;
	ABN_DECODER_PHI_E	s16(31:16)	ABN_DECODER_PHI_E = (ABN_DECODER_PHI_M × N_POLE_PAIRS_) + ABN_DECODER_PHI_E_OFFSET
0x2C$_h$	ABN_2_DECODER_MODE		This register is used to configure decoder input signals and N pulse action as well as count direction.
	apol	bit(0)	Polarity of A pulse. 0: off

			1: on
	bpol	bit(1)	Polarity of B pulse. 0: off 1: on
	npol	bit(2)	Polarity of N pulse. 0: off 1: on
	use_abn_as_n	bit(3)	0: Ignore A and B polarity with Npulse = N, 1 : Npulse = N and A and B 0: Ignore A and B polarity with Npulse = N 1: Npulse = N and A and B
	cln	bit(8)	Write direction at Npulse event between ABN_2_DECODER_COUNT_N and ABN_2_DECODER_COUNT. 0: COUNT => COUNT_N 1: COUNT_N => COUNT
	direction	bit(12)	Decoder count direction. 0: positive 1: negative
$0x2D_h$	ABN_2_DECODER_PPR		This register is used to set PPR number of encoder.
	ABN_2_DECODER_PPR	u24(23:0)	Decoder_2 pules per mechanical revolution. This second ABN encoder interface is for positioning or velocity control but NOT for motor commutation.
$0x2E_h$	ABN_2_DECODER_COUNT		This register displays the actual count of encoder steps. It can be overwritten for initialization.
	ABN_2_DECODER_COUNT	u24(23:0)	Raw decoder_2 count; the digital decoder engine counts modulo (decoder_2_ppr).
$0x2F_h$	ABN_2_DECODER_COUNT_N		This register displays the count value at last N pulse event. It can also be used to overwrite decoder count at N pulse event according to decoder mode register setting.
	ABN_2_DECODER_COUNT_N	u24(23:0)	Decoder_2 count latched on N pulse, when N pulse clears decoder_2_count also decoder_2_count_n is 0.

$0x30_h$	ABN_2_DECODER_PHI_M_OFFSET		This register can be used to set offsets for electrical and mechanical angle calculated from decoder.
	ABN_2_DECODER_PHI_M_OFFSET	s16(15:0)	ABN_2_DECODER_PHI_M_OFFSET to shift (rotate) angle DECODER_2_PHI_M.
$0x31_h$	ABN_2_DECODER_PHI_M		This register displays actual angle values for ABN encoder.
	ABN_2_DECODER_PHI_M	s16(15:0)	ABN_2_DECODER_PHI_M = ABN_2_DECODER_COUNT × 2^16 / ABN_2_DECODER_PPR + ABN_2_DECODER_PHI_M_OFFSET;
$0x33_h$	HALL_MODE		This register is used to set basic settings for the digital Hall interface.
	polarity	bit(0)	polarity
			0: off
			1: on
	synchronous PWM sampling	bit(4)	enable sampling synchronous to PWM
			0: off
			1: on
	interpolation	bit(8)	interpolation
			0: off
			1: on
	direction	bit(12)	direction
			0: off
			1: on
	HALL_BLANK	u12(27:16)	tBLANK = 10ns × HALL_BLANK
$0x34_h$	HALL_POSITION_060_000		This register is used to calibrate hall sensor offset.
	HALL_POSITION_000	s16(15:0)	s16 hall sensor position at 0°
	HALL_POSITION_060	s16(31:16)	s16 hall sensor position at 60°.
$0x35_h$	HALL_POSITION_180_120		This register is used to calibrate hall sensor offset.
	HALL_POSITION_120	s16(15:0)	s16 hall sensor position at 120°.
	HALL_POSITION_180	s16(31:16)	s16 hall sensor position at 180°.
$0x36_h$	HALL_POSITION_300_240		This register is used to calibrate hall sensor offset.
	HALL_POSITION_240	s16(15:0)	s16 hall sensor position at 240°.

	HALL_POSITION_300	s16(31:16)	s16 hall sensor position at 300°.
0x37$_h$	HALL_PHI_E_PHI_M_OFFSET		This register is used to set offsets for calculated angles from hall interface.
	HALL_PHI_M_OFFSET	s16(15:0)	Offset of mechanical angle hall_phi_m of hall decoder.
	HALL_PHI_E_OFFSET	s16(31:16)	Offset for electrical angle hall_phi_e of hall decoder.
0x38$_h$	HALL_DPHI_MAX		This register is used to set a maximum difference of two hall sensor transitions for hall position extrapolation.
	HALL_DPHI_MAX	u16(15:0)	Maximum dx for interpolation (default for digital hall: u16/6).
0x39$_h$	HALL_PHI_E_INTERPOLATED_PHI_E		This register displays interpolated and raw angle of hall interface.
	HALL_PHI_E	s16(15:0)	Raw electrical angle hall_phi_e of hall decoder, selection programmed through HALL_MODE control bit.
	HALL_PHI_E_INTERPOLATED	s16(31:16)	Interpolated electrical angle hall_phi_e_interpolated, selection programmed through HALL_MODE control bit.
0x3A$_h$	HALL_PHI_M		This register displays the mechanical angle calculated in hall sensor interface.
	HALL_PHI_M	s16(15:0)	Mechanical angle hall_phi_m of hall decoder.
0x3B$_h$	AENC_DECODER_MODE		This register sets basic information for the analog encoder interface.
	AENC_DECODER_MODE[0]	bit(0)	120deg_n90deg 0: 90 degree 1: 120 degree
	AENC_DECODER_MODE[12]	bit(12)	decoder count direction 0: positive 1: negative
0x3C$_h$	AENC_DECODER_N_THRESHOLD		This register sets analog encoder N pulse processing function.

	AENC_DECODER_N_THRESHOLD	u16(15:0)	Threshold for generating of N pulse from analog AENC_N signal (only needed for analog SinCos encoders with analog N signal).
0x3D$_h$	AENC_DECODER_PHI_A_RAW		Displays raw angle after ATAN2 calculation.
	AENC_DECODER_PHI_A_RAW	s16(15:0)	Raw analog angle phi calculated from analog AENC inputs (analog hall, analog SinCos, ...).
0x3E$_h$	AENC_DECODER_PHI_A_OFFSET		This register sets the offset of PHI_A for phase alignment.
	AENC_DECODER_PHI_A_OFFSET	s16(15:0)	Offset for angle phi from analog decoder (analog hall, analog SinCos, ...).
0x3F$_h$	AENC_DECODER_PHI_A		This register displays offset compensated PHI_A angle.
	AENC_DECODER_PHI_A	s16(15:0)	Resulting phi available for the FOC (phi_e might need to be calculated from this angle through aenc_decoder_ppr, for analog hall sensors phi_a might be used directly as phi_e depends on analog hall signal type).
0x40$_h$	AENC_DECODER_PPR		This register sets the number of periods per revolution for analog encoder.
	AENC_DECODER_PPR	s16(15:0)	Number of periods per revolution also called lines per revolution (different nomenclatur compared to digital ABN encoders). This value should be chosen 1 or a power of 2.
0x41$_h$	AENC_DECODER_COUNT		Displays the count value of analog encoder periods.
	AENC_DECODER_COUNT	s32(31:0)	Decoder position, raw unscaled. May be overwritten.
0x42$_h$	AENC_DECODER_COUNT_N		Displays the count value at last N pulse event. Can also be used to auto-overwrite decoder count at N pulse event.
	AENC_DECODER_COUNT_N	s32(31:0)	Latched decoder position on analog N pulse event.
0x45$_h$	AENC_DECODER_PHI_E_PHI_M_OFFSET		This register sets offsets for electrical and mechanical angle calculated from AENC interface.

	AENC_DECODER_PHI_M_OFFSET	s16(15:0)	Offset for mechanical angle phi_m.
	AENC_DECODER_PHI_E_OFFSET	s16(31:16)	Offset for electrical angle phi_e.
$0x46_h$	AENC_DECODER_PHI_E_PHI_M		Displays actual angle values of analog encoder interface.
	AENC_DECODER_PHI_M	s16(15:0)	Resulting angle phi_m.
	AENC_DECODER_PHI_E	s16(31:16)	Resulting angle phi_e.
$0x4D_h$	CONFIG_DATA		This multi-purpose register is used to set configuration parameters of controller cascade and input signal conditioning.
	biquad_x_a_1	s32(31:0)	
	biquad_x_a_2	s32(31:0)	
	biquad_x_b_0	s32(31:0)	
	biquad_x_b_1	s32(31:0)	
	biquad_x_b_2	s32(31:0)	
	biquad_x_enable	bit(0)	0: off
			1: on
	biquad_v_a_1	s32(31:0)	
	biquad_v_a_2	s32(31:0)	
	biquad_v_b_0	s32(31:0)	
	biquad_v_b_1	s32(31:0)	
	biquad_v_b_2	s32(31:0)	
	biquad_v_enable	bit(0)	0: off
			1: on
	biquad_t_a_1	s32(31:0)	
	biquad_t_a_2	s32(31:0)	
	biquad_t_b_0	s32(31:0)	
	biquad_t_b_1	s32(31:0)	
	biquad_t_b_2	s32(31:0)	
	biquad_t_enable	bit(0)	0: off
			1: on
	biquad_f_a_1	s32(31:0)	
	biquad_f_a_2	s32(31:0)	
	biquad_f_b_0	s32(31:0)	
	biquad_f_b_1	s32(31:0)	
	biquad_f_b_2	s32(31:0)	
	biquad_f_enable	bit(0)	0: off

		1: on
ref_switch_config	u16(15:0)	
		1: on
SINGLE_PIN_IF_CFG	u8(7:0)	
SINGLE_PIN_IF_STATUS	u16(31:16)	
SINGLE_PIN_IF_OFFSET	u16(15:0)	Offset for scaling of single pin interface input
SINGLE_PIN_IF_SCALE	s16(31:16)	Gain factor of single pin interface input
CURRENT_I_nQ8.8_Q4.12	bit(0)	If this bit is set, Q4.12 representation of I parameter for torque/flux control is used. If bit is not set, Q8.8 representation is used
		0: Q8.8 representation is used
		1: Q4.12 representation is used
CURRENT_P_nQ8.8_Q4.12	bit(1)	If this bit is set, Q4.12 representation of P for parameter for torque/flux control is used. If bit is not set, Q8.8 representation is used
		0: Q8.8 representation is used
		1: Q4.12 representation is used
VELOCITY_I_nQ8.8_Q4.12	bit(2)	If this bit is set, Q4.12 representation of I parameter for velocity control is used. If bit is not set, Q8.8 representation is used
		0: Q8.8 representation is used
		1: Q4.12 representation is used
VELOCITY_P_nQ8.8_Q4.12	bit(3)	If this bit is set, Q4.12 representation of P for parameter for velocity control is used. If bit is not set, Q8.8 representation is used
		0: Q8.8 representation is used
		1: Q4.12 representation is used
POSITION_I_nQ8.8_Q4.12	bit(4)	If this bit is set, Q4.12 representation of I parameter for position control is used. If bit is not set, Q8.8 representation is used
		0: Q8.8 representation is used
		1: Q4.12 representation is used

	POSITION_P_nQ8.8_Q4.12	bit(5)	If this bit is set, Q4.12 representation of P for parameter for position control is used. If bit is not set, Q8.8 representation is used 0: Q8.8 representation is used 1: Q4.12 representation is used
$0x4E_h$	CONFIG_ADDR CONFIG_ADDR	 u32(31:0)	This register is used to select function of CONFIG_DATA register. 1: biquad_x_a_1 2: biquad_x_a_2 4: biquad_x_b_0 5: biquad_x_b_1 6: biquad_x_b_2 7: biquad_x_enable 9: biquad_v_a_1 10: biquad_v_a_2 12: biquad_v_b_0 13: biquad_v_b_1 14: biquad_v_b_2 15: biquad_v_enable 17: biquad_t_a_1 18: biquad_t_a_2 20: biquad_t_b_0 21: biquad_t_b_1 22: biquad_t_b_2 23: biquad_t_enable 25: biquad_f_a_1 26: biquad_f_a_2 28: biquad_f_b_0 29: biquad_f_b_1 30: biquad_f_b_2 31: biquad_f_enable 51: ref_switch_config 60: SINGLE_PIN_IF_STATUS_CFG 61: SINGLE_PIN_IF_SCALE_OFFSET 62: ADVANCED_PI_REPRESENT.

$0x50_h$	VELOCITY_SELECTION		This register is used to select an angle signal for the velocity control loop and velocity calculation.
	VELOCITY_SELECTION	u8(7:0)	Selects the velocity source for velocity measurement. 0: PHI_E_SELECTION 1: phi_e_ext 2: phi_e_openloop 3: phi_e_abn 4: reserved 5: phi_e_hal 6: phi_e_aenc 7: phi_a_aenc 8: reserved 9: phi_m_abn 10: phi_m_abn_2 11: phi_m_aenc 12: phi_m_hal
	VELOCITY_METER_SELECTION	u8(15:8)	0: default 1: advanced
$0x51_h$	POSITION_SELECTION		This register is used to select an angle signal for the position calculation and control loop.
	POSITION_SELECTION	u8(7:0)	0: phi_e selected through PHI_E_SELECTION 1: phi_e_ext 2: phi_e_openloop 3: phi_e_abn 4: reserved 5: phi_e_hal 6: phi_e_aenc 7: phi_a_aenc 8: reserved 9: phi_m_abn 10: phi_m_abn_2 11: phi_m_aenc 12: phi_m_hal

$0x52_h$	PHI_E_SELECTION		This register is used to select an angle signal for FOC transformation as electrical angle of the motor.
	PHI_E_SELECTION	u8(7:0)	0: reserved 1: phi_e_ext 2: phi_e_openloop 3: phi_e_abn 4: reserved 5: phi_e_hal 6: phi_e_aenc 7: phi_a_aenc
$0x53_h$	PHI_E		This register displays the actual chosen electrical angle value.
	PHI_E	s16(15:0)	Angle used for the inner FOC loop.
$0x54_h$	PID_FLUX_P_FLUX_I		This register sets control parameters for flux controller.
	PID_FLUX_I	s16(15:0)	
	PID_FLUX_P	s16(31:16)	
$0x56_h$	PID_TORQUE_P_TORQUE_I		This register sets control parameters for torque controller.
	PID_TORQUE_I	s16(15:0)	
	PID_TORQUE_P	s16(31:16)	
$0x58_h$	PID_VELOCITY_P_VELOCITY_I		This register sets control parameters for velocity controller.
	PID_VELOCITY_I	s16(15:0)	
	PID_VELOCITY_P	s16(31:16)	
$0x5A_h$	PID_POSITION_P_POSITION_I		This register sets control parameters for position controller.
	PID_POSITION_I	s16(15:0)	
	PID_POSITION_P	s16(31:16)	
$0x5D_h$	PIDOUT_UQ_UD_LIMITS		This register sets the output voltage/duty cycle limit for the current controllers. iPARK CIRLIM block limits voltage output vector length to this value.
	PIDOUT_UQ_UD_LIMITS	s16(15:0)	Two dimensional circular limiter for inputs of iPark. HINT: the absolute value of the register is used (possible values: 0 ... 32767).

$0x5E_h$	PID_TORQUE_FLUX_LIMITS		This register is used to set target current limit for both controllers.
	PID_TORQUE_FLUX_LIMITS	u16(15:0)	PID torque limt and PID flux limit, limits the target values coming from the target registers.
$0x60_h$	PID_VELOCITY_LIMIT		This register is used to set an absolute velocity limit for velocity controller input.
	PID_VELOCITY_LIMIT	u32(31:0)	Velocity limit.
$0x61_h$	PID_POSITION_LIMIT_LOW		This register is used to set a lower limit for position controller input.
	PID_POSITION_LIMIT_LOW	s32(31:0)	Position limit low, programmable position barrier.
$0x62_h$	PID_POSITION_LIMIT_HIGH		This register is used to set a higher limit for position controller input.
	PID_POSITION_LIMIT_HIGH	s32(31:0)	Position limit high, programmable position barrier.
$0x63_h$	MODE_RAMP_MODE_MOTION		This register is used to set a motion mode, a downsampling factor for velocity and position control loop, and the PI controller structure type.
	MODE_MOTION	u8(7:0)	0: stopped_mode 1: torque_mode 2: velocity_mode 3: position_mode 4: reserved 5: reserved 6: reserved 7: reserved 8: uq_ud_ext 9: reserved 10: AGPI_A torque_mode 11: AGPI_A velocity_mode 12: AGPI_A position_mode 13: PWM_I torque_mode 14: PWM_I velocity_mode 15: PWM_I position_mode

	MODE_PID_SMPL	u7(30:24)	Update rate of advanced architecture position controller: $\frac{f_{PWM}}{MODE\_PID\_SMPL + 1}$ (48)
	MODE_PID_TYPE	bit(31)	0: parallel/classic PI 1: sequential/advanced PI
$0x64_h$	PID_TORQUE_FLUX_TARGET		Target values for torque and flux controllers in torque mode.
	PID_FLUX_TARGET	s16(15:0)	
	PID_TORQUE_TARGET	s16(31:16)	
$0x65_h$	PID_TORQUE_FLUX_OFFSET		Offsets for software torque and flux control loop inputs for feed-forward control.
	PID_FLUX_OFFSET	s16(15:0)	Flux offset for feed forward control.
	PID_TORQUE_OFFSET	s16(31:16)	Torque offset for feed forward control.
$0x66_h$	PID_VELOCITY_TARGET		Target velocity value for velocity controller in velocity mode.
	PID_VELOCITY_TARGET	s32(31:0)	Target velocity register (for velocity mode).
$0x67_h$	PID_VELOCITY_OFFSET		Offset velocity value for velocity controller in velocity and position mode.
	PID_VELOCITY_OFFSET	s32(31:0)	Velocity offset for feed forward control.
$0x68_h$	PID_POSITION_TARGET		Target position value for position controller in position mode.
	PID_POSITION_TARGET	s32(31:0)	Target position register (for position mode).
$0x69_h$	PID_TORQUE_FLUX_ACTUAL		Actual torque and flux value measured by ADC and depending on the motor type after park and clarke transformation.
	PID_FLUX_ACTUAL	s16(15:0)	
	PID_TORQUE_ACTUAL	s16(31:16)	
$0x6A_h$	PID_VELOCITY_ACTUAL		Filtered actual velocity derived from chosen angle signal.
	PID_VELOCITY_ACTUAL	s32(31:0)	Actual velocity.

0x6B$_h$	PID_POSITION_ACTUAL		Actual position derived from chosen position signal.
	PID_POSITION_ACTUAL	s32(31:0)	Actual multi turn position for positioning. Input position differences are accumulated. Lower 16 bits display one revolution of input angle. Upper 16 bits display revolutions. WRITE on PID_POSITION_ACTUAL writes same value into PID_POSITION_TARGET to avoid unwanted move.
0x6C$_h$	PID_ERROR_DATA		Register displays control errors of controllers for testing according to selection PID_ERROR_ADDR .
	PID_TORQUE_ERROR	s32(31:0)	PID torque error.
	PID_FLUX_ERROR	s32(31:0)	PID flux error.
	PID_VELOCITY_ERROR	s32(31:0)	PID velocity error.
	PID_POSITION_ERROR	s32(31:0)	PID position error.
	PID_TORQUE_ERROR_SUM	s32(31:0)	PID torque error.
	PID_FLUX_ERROR_SUM	s32(31:0)	PID flux error sum.
	PID_VELOCITY_ERROR_SUM	s32(31:0)	PID velocity error sum.
	PID_POSITION_ERROR_SUM	s32(31:0)	PID position error sum.
0x6D$_h$	PID_ERROR_ADDR		Register is used to set function of PID_ERROR_DATA register.
	PID_ERROR_ADDR	u8(7:0)	0: PID_TORQUE_ERROR
			1: PID_FLUX_ERROR
			2: PID_VELOCITY_ERROR
			3: PID_POSITION_ERROR
			4: PID_TORQUE_ERROR_SUM
			5: PID_FLUX_ERROR_SUM
			6: PID_VELOCITY_ERROR_SUM
			7: PID_POSITION_ERROR_SUM
0x6E$_h$	INTERIM_DATA		This register is used to display internal signals from controller cascade for monitoring.
	PIDIN_TARGET_TORQUE	s32(31:0)	PIDIN target torque.
	PIDIN_TARGET_FLUX	s32(31:0)	PIDIN target flux.
	PIDIN_TARGET_VELOCITY	s32(31:0)	PIDIN target velocity.
	PIDIN_TARGET_POSITION	s32(31:0)	PIDIN target position.
	PIDOUT_TARGET_TORQUE	s32(31:0)	PIDOUT target torque.

PIDOUT_TARGET_FLUX	s32(31:0)	PIDOUT target flux.
PIDOUT_TARGET_VELOCITY	s32(31:0)	PIDOUT target velocity.
PIDOUT_TARGET_POSITION	s32(31:0)	PIDOUT target position.
FOC_IUX	s16(15:0)	
FOC_IWY	s16(31:16)	
FOC_IV	s16(15:0)	
FOC_IA	s16(15:0)	
FOC_IB	s16(31:16)	
FOC_ID	s16(15:0)	
FOC_IQ	s16(31:16)	
FOC_UD	s16(15:0)	
FOC_UQ	s16(31:16)	
FOC_UD_LIMITED	s16(15:0)	
FOC_UQ_LIMITED	s16(31:16)	
FOC_UA	s16(15:0)	
FOC_UB	s16(31:16)	
FOC_UUX	s16(15:0)	
FOC_UWY	s16(31:16)	
FOC_UV	s16(15:0)	
PWM_UX	s16(15:0)	
PWM_WY	s16(31:16)	
PWM_V	s16(15:0)	
ADC_I_0	s16(15:0)	
ADC_I_1	s16(31:16)	
PID_FLUX_ACTUAL_DIV256	s8(7:0)	
PID_TORQUE_ACTUAL_DIV256	s8(15:8)	
PID_FLUX_TARGET_DIV256	s8(23:16)	
PID_TORQUE_TARGET_DIV256	s8(31:24)	
PID_TORQUE_ACTUAL	s16(15:0)	
PID_TORQUE_TARGET	s16(31:16)	
PID_FLUX_ACTUAL	s16(15:0)	
PID_FLUX_TARGET	s16(31:16)	
PID_VELOCITY_ACTUAL_DIV256	s16(15:0)	
PID_VELOCITY_TARGET_DIV256	s16(31:16)	
PID_VELOCITY_ACTUAL_LSB	s16(15:0)	
PID_VELOCITY_TARGET_LSB	s16(31:16)	

	PID_POSITION_ACTUAL_DIV256	s16(15:0)	
	PID_POSITION_TARGET_DIV256	s16(31:16)	
	PID_POSITION_ACTUAL_LSB	s16(15:0)	
	PID_POSITION_TARGET_LSB	s16(31:16)	
	FF_VELOCITY	s32(31:0)	
	FF_TORQUE	s16(15:0)	
	ACTUAL_VELOCITY_PPTM	s32(31:0)	
	REF_SWITCH_STATUS	u16(15:0)	
	HOME_POSITION	s32(31:0)	
	LEFT_POSITION	s32(31:0)	
	RIGHT_POSITION	s32(31:0)	
	SINGLE_PIN_IF_TARGET_TORQUE	s16(15:0)	
	SINGLE_PIN_IF_PWM_DUTY_CYCLE	s16(31:16)	
	SINGLE_PIN_IF_TARGET_VELOCITY	s32(31:0)	
	SINGLE_PIN_IF_TARGET_POSITION	s32(31:0)	
$0x6F_h$	INTERIM_ADDR		Sets function of register INTERIM_DATA.
	INTERIM_ADDR	u8(7:0)	0: PIDIN_TARGET_TORQUE
			1: PIDIN_TARGET_FLUX
			2: PIDIN_TARGET_VELOCITY
			3: PIDIN_TARGET_POSITION
			4: PIDOUT_TARGET_TORQUE
			5: PIDOUT_TARGET_FLUX
			6: PIDOUT_TARGET_VELOCITY
			7: PIDOUT_TARGET_POSITION
			8: FOC_IWY_IUX
			9: FOC_IV
			10: FOC_IB_IA
			11: FOC_IQ_ID
			12: FOC_UQ_UD
			13: FOC_UQ_UD_LIMITED
			14: FOC_UB_UA
			15: FOC_UWY_UUX
			16: FOC_UV
			17: PWM_WY_UX
			18: PWM_UV

			19: ADC_I1_I0 20: PID_TORQUE_TARGET_FLUX_TARGET_TORQUE_ACTUAL_FLUX_ACTUAL_DIV256 21: PID_TORQUE_TARGET_TORQUE_ACTUAL 22: PID_FLUX_TARGET_FLUX_ACTUAL 23: PID_VELOCITY_TARGET_VELOCITY_ACTUAL_DIV256 24: PID_VELOCITY_TARGET_VELOCITY_ACTUAL 25: PID_POSITION_TARGET_POSITION_ACTUAL_DIV256 26: PID_POSITION_TARGET_POSITION_ACTUAL 27: FF_VELOCITY 28: FF_TORQUE 29: ACTUAL_VELOCITY_PPTM 30: REF_SWITCH_STATUS 31: HOME_POSITION 32: LEFT_POSITION 33: RIGHT_POSITION 42: SINGLE_PIN_IF_PWM_DUTY_CYCLE_TORQUE_TARGET 43: SINGLE_PIN_IF_VELOCITY_TARGET 44: SINGLE_PIN_IF_POSITION_TARGET
$0x75_h$	ADC_VM_LIMITS		Sets supply voltage limits for brake chopper output action.
	ADC_VM_LIMIT_LOW	u16(15:0)	Low limit for brake chopper output BRAKE_OUT.
	ADC_VM_LIMIT_HIGH	u16(31:16)	High limit for brake chopper output BRAKE_OUT.
$0x76_h$	TMC4671_INPUTS_RAW		Displays actual input signals of IC for monitoring and connection testing.
	A of ABN_RAW	bit(0)	A of ABN_RAW 0: off 1: on

B of ABN_RAW	bit(1)	B of ABN_RAW 0: off 1: on
N of ABN_RAW	bit(2)	N of ABN_RAW 0: off 1: on
-	bit(3)	— 0: off 1: on
A of ABN_2_RAW	bit(4)	A of ABN_2_RAW 0: off 1: on
B of ABN_2_RAW	bit(5)	B of ABN_2_RAW 0: off 1: on
N of ABN_2_RAW	bit(6)	N of ABN_2_RAW 0: off 1: on
-	bit(7)	— 0: off 1: on
HALL_UX of HALL_RAW	bit(8)	HALL_UX of HALL_RAW 0: off 1: on
HALL_V of HALL_RAW	bit(9)	HALL_V of HALL_RAW 0: off 1: on
HALL_WY of HALL_RAW	bit(10)	HALL_WY of HALL_RAW 0: off 1: on
-	bit(11)	— 0: off 1: on
REF_SW_R_RAW	bit(12)	REF_SW_R_RAW 0: off 1: on

REF_SW_H_RAW	bit(13)	REF_SW_H_RAW 0: off 1: on
REF_SW_L_RAW	bit(14)	REF_SW_L_RAW 0: off 1: on
ENABLE_IN_RAW	bit(15)	ENABLE_IN_RAW 0: off 1: on
STP of DIRSTP_RAW	bit(16)	STP of DIRSTP_RAW 0: off 1: on
DIR of DIRSTP_RAW	bit(17)	DIR of DIRSTP_RAW 0: off 1: on
PWM_IN_RAW	bit(18)	PWM_IN_RAW 0: off 1: on
-	bit(19)	— 0: off 1: on
HALL_UX_FILT	bit(20)	ESI_0 of ESI_RAW 0: off 1: on
HALL_V_FILT	bit(21)	ESI_1 of ESI_RAW 0: off 1: on
HALL_WY_FILT	bit(22)	ESI_2 of ESI_RAW 0: off 1: on
-	bit(23)	— 0: off 1: on
-	bit(24)	CFG_0 of CFG 0: off 1: on

	-	bit(25)	CFG_1 of CFG 0: off 1: on
	-	bit(26)	CFG_2 of CFG 0: off 1: on
	-	bit(27)	CFG_3 of CFG 0: off 1: on
	PWM_IDLE_L_RAW	bit(28)	PWM_IDLE_L_RAW 0: off 1: on
	PWM_IDLE_H_RAW	bit(29)	PWM_IDLE_H_RAW 0: off 1: on
	-	bit(30)	DRV_ERR_IN_RAW 0: off 1: on
	-	bit(31)	— 0: off 1: on
$0x77_h$	TMC4671_OUTPUTS_RAW		Displays actual output signals of IC for monitoring and connection testing.
	TMC4671_OUTPUTS_RAW[0]	bit(0)	PWM_UX1_L 0: off 1: on
	TMC4671_OUTPUTS_RAW[1]	bit(1)	PWM_UX1_H 0: off 1: on
	TMC4671_OUTPUTS_RAW[2]	bit(2)	PWM_VX2_L 0: off 1: on
	TMC4671_OUTPUTS_RAW[3]	bit(3)	PWM_VX2_H 0: off 1: on
	TMC4671_OUTPUTS_RAW[4]	bit(4)	PWM_WY1_L

			0: off 1: on
	TMC4671_OUTPUTS_RAW[5]	bit(5)	PWM_WY1_H 0: off 1: on
	TMC4671_OUTPUTS_RAW[6]	bit(6)	PWM_Y2_L 0: off 1: on
	TMC4671_OUTPUTS_RAW[7]	bit(7)	PWM_Y2_H 0: off 1: on
$0x78_h$	STEP_WIDTH		Sets a step width of an acutal input step signal on STEP/DIR interface. Target position is decreased/increased by this value according to Dir signal.
	STEP_WIDTH	s32(31:0)	STEP WIDTH = 0 => STP pulses ignored, resulting direction = DIR XOR sign (STEP_WIDTH), affects PID_POSITION_TARGET
$0x79_h$	UART_BPS		Sets the desired UART baudrate. Must be entered as hexadecimal number (example, baudrate 9600 is set by entering $0x00009600_h$)
	UART_BPS	u24(23:0)	$0x00009600_h$, $0x00115200_h$, $0x00921600_h$, $0x03000000_h$ (default = 0x00009600)
$0x7B_h$	GPIO_dsADCI_CONFIG		Sets the function and controls the GPIOs if RTMI is not used. Check functional description for detailed explanation of options.
	GPIO_dsADCI_CONFIG[0]	bit(0)	SEL_nDBGSPIM_GPIO 0: off 1: on
	GPIO_dsADCI_CONFIG[1]	bit(1)	SEL_nGPIO_dsADCS_A 0: off 1: on
	GPIO_dsADCI_CONFIG[2]	bit(2)	SEL_nGPIO_dsADCS_B 0: off 1: on
	GPIO_dsADCI_CONFIG[3]	bit(3)	SEL_GPIO_GROUP_A_nIN_OUT

			0: off
			1: on
	GPIO_dsADCI_CONFIG[4]	bit(4)	SEL_GPIO_GROUP_B_nIN_OUT
			0: off
			1: on
	GPIO_dsADCI_CONFIG[5]	bit(5)	SEL_GROUP_A_DSADCS_nCLKIN_CLKOUT
			0: off
			1: on
	GPIO_dsADCI_CONFIG[6]	bit(6)	SEL_GROUP_B_DSADCS_nCLKIN_CLKOUT
			0: off
			1: on
	GPO	u8(23:16)	
	GPI	u8(31:24)	
0x7C$_h$	STATUS_FLAGS		Displays actual status flags to set status output. The register is also used to reset status flags.
	STATUS_FLAGS[0]	bit(0)	pid_x_target_limit
			0: off
			1: on
	STATUS_FLAGS[1]	bit(1)	reserved
			0: off
			1: on
	STATUS_FLAGS[2]	bit(2)	pid_x_errsum_limit
			0: off
			1: on
	STATUS_FLAGS[3]	bit(3)	pid_x_output_limit
			0: off
			1: on
	STATUS_FLAGS[4]	bit(4)	pid_v_target_limit
			0: off
			1: on
	STATUS_FLAGS[5]	bit(5)	reserved
			0: off
			1: on
	STATUS_FLAGS[6]	bit(6)	pid_v_errsum_limit

		0: off 1: on
STATUS_FLAGS[7]	bit(7)	pid_v_output_limit 0: off 1: on
STATUS_FLAGS[8]	bit(8)	pid_id_target_limit 0: off 1: on
STATUS_FLAGS[9]	bit(9)	reserved 0: off 1: on
STATUS_FLAGS[10]	bit(10)	pid_id_errsum_limit 0: off 1: on
STATUS_FLAGS[11]	bit(11)	pid_id_output_limit 0: off 1: on
STATUS_FLAGS[12]	bit(12)	pid_iq_target_limit 0: off 1: on
STATUS_FLAGS[13]	bit(13)	reserved 0: off 1: on
STATUS_FLAGS[14]	bit(14)	pid_iq_errsum_limit 0: off 1: on
STATUS_FLAGS[15]	bit(15)	pid_iq_output_limit 0: off 1: on
STATUS_FLAGS[16]	bit(16)	ipark_cirlim_limit_u_d 0: off 1: on
STATUS_FLAGS[17]	bit(17)	ipark_cirlim_limit_u_q 0: off 1: on
STATUS_FLAGS[18]	bit(18)	ipark_cirlim_limit_u_r

		0: off 1: on
STATUS_FLAGS[20]	bit(20)	ref_sw_r 0: off 1: on
STATUS_FLAGS[21]	bit(21)	ref_sw_h 0: off 1: on
STATUS_FLAGS[22]	bit(22)	ref_sw_l 0: off 1: on
STATUS_FLAGS[23]	bit(23)	— 0: off 1: on
STATUS_FLAGS[24]	bit(24)	pwm_min 0: off 1: on
STATUS_FLAGS[25]	bit(25)	pwm_max 0: off 1: on
STATUS_FLAGS[26]	bit(26)	adc_i_clipped 0: off 1: on
STATUS_FLAGS[27]	bit(27)	aenc_clipped 0: off 1: on
STATUS_FLAGS[28]	bit(28)	enc_n 0: off 1: on
STATUS_FLAGS[29]	bit(29)	enc_2_n 0: off 1: on
STATUS_FLAGS[30]	bit(30)	aenc_n 0: off 1: on
STATUS_FLAGS[31]	bit(31)	reserved

			0: off 1: on
$0x7D_h$	STATUS_MASK STATUS_MASK	 u32(31:0)	Register is used to set a mask for STATUS_FLAGS register to set STATUS output pin.

7.3 Register Map - Defaults, Min., Max.

RD/WR	ADDR	NAME	DEFAULT	MIN	MAX
R	$0x00_h$	CHIPINFO_DATA			
		SI_TYPE	$0x0_h$	$0x0_h$	$0xFFFFFFFF_h$
		SI_VERSION	$0x0_h$	$0x0_h$	$0xFFFFFFFF_h$
		SI_DATE	$0x0_h$	$0x0_h$	$0xFFFFFFFF_h$
		SI_TIME	$0x0_h$	$0x0_h$	$0xFFFFFF_h$
		SI_VARIANT	$0x0_h$	$0x0_h$	$0xFFFFFFFF_h$
		SI_BUILD	$0x0_h$	$0x0_h$	$0xFFFFFFFF_h$
RW	$0x01_h$	CHIPINFO_ADDR			
		CHIP_INFO_ADDRESS	$0x0_h$	$0x0_h$	$0x5_h$
R	$0x02_h$	ADC_RAW_DATA			
		ADC_I0_RAW	$0x0_h$	$0x0_h$	$0xFFFF_h$
		ADC_I1_RAW	$0x0_h$	$0x0_h$	$0xFFFF_h$
		ADC_VM_RAW	$0x0_h$	$0x0_h$	$0xFFFF_h$
		ADC_AGPI_A_RAW	$0x0_h$	$0x0_h$	$0xFFFF_h$
		ADC_AGPI_B_RAW	$0x0_h$	$0x0_h$	$0xFFFF_h$
		ADC_AENC_UX_RAW	$0x0_h$	$0x0_h$	$0xFFFF_h$
		ADC_AENC_VN_RAW	$0x0_h$	$0x0_h$	$0xFFFF_h$
		ADC_AENC_WY_RAW	$0x0_h$	$0x0_h$	$0xFFFF_h$
RW	$0x03_h$	ADC_RAW_ADDR			
		ADC_RAW_ADDR	$0x0_h$	$0x0_h$	$0x3_h$
RW	$0x04_h$	dsADC_MCFG_B_ MCFG_A			
		cfg_dsmodulator_a	$0x0_h$	$0x0_h$	$0x3_h$
		mclk_polarity_a	$0x0_h$	$0x0_h$	$0x1_h$
		mdat_polarity_a	$0x0_h$	$0x0_h$	$0x1_h$
		sel_nclk_mclk_i_a	$0x0_h$	$0x0_h$	$0x1_h$
		cfg_dsmodulator_b	$0x0_h$	$0x0_h$	$0x3_h$
		mclk_polarity_b	$0x0_h$	$0x0_h$	$0x1_h$
		mdat_polarity_b	$0x0_h$	$0x0_h$	$0x1_h$
		sel_nclk_mclk_i_b	$0x0_h$	$0x0_h$	$0x1_h$
RW	$0x05_h$	dsADC_MCLK_A			
		dsADC_MCLK_A	$0x20000000_h$	$0x0_h$	$0xFFFFFFFF_h$
RW	$0x06_h$	dsADC_MCLK_B			

		dsADC_MCLK_B	0x20000000$_h$	0x0$_h$	0xFFFFFFFF$_h$
RW	0x07$_h$	dsADC_MDEC_B_MDEC_A			
		dsADC_MDEC_A	0x100$_h$	0x0$_h$	0xFFFF$_h$
		dsADC_MDEC_B	0x100$_h$	0x0$_h$	0xFFFF$_h$
RW	0x08$_h$	ADC_I1_SCALE_OFFSET			
		ADC_I1_OFFSET	0x0$_h$	0x0$_h$	0xFFFF$_h$
		ADC_I1_SCALE	0x100$_h$	-0x8000$_h$	0x7FFF$_h$
RW	0x09$_h$	ADC_I0_SCALE_OFFSET			
		ADC_I0_OFFSET	0x0$_h$	0x0$_h$	0xFFFF$_h$
		ADC_I0_SCALE	0x100$_h$	-0x8000$_h$	0x7FFF$_h$
RW	0x0A$_h$	ADC_I_SELECT			
		ADC_I0_SELECT	0x0$_h$	0x0$_h$	0x3$_h$
		ADC_I1_SELECT	0x1$_h$	0x0$_h$	0x3$_h$
		ADC_I_UX_SELECT	0x0$_h$	0x0$_h$	0x2$_h$
		ADC_I_V_SELECT	0x1$_h$	0x0$_h$	0x2$_h$
		ADC_I_WY_SELECT	0x2$_h$	0x0$_h$	0x2$_h$
RW	0x0B$_h$	ADC_I1_I0_EXT			
		ADC_I0_EXT	0x0$_h$	0x0$_h$	0xFFFF$_h$
		ADC_I1_EXT	0x0$_h$	0x0$_h$	0xFFFF$_h$
RW	0x0C$_h$	DS_ANALOG_INPUT_STAGE_CFG			
		ADC_I0	0x0$_h$	0x0$_h$	0x7$_h$
		ADC_I1	0x0$_h$	0x0$_h$	0x7$_h$
		ADC_VM	0x0$_h$	0x0$_h$	0x7$_h$
		ADC_AGPI_A	0x0$_h$	0x0$_h$	0x7$_h$
		ADC_AGPI_B	0x0$_h$	0x0$_h$	0x7$_h$
		ADC_AENC_UX	0x0$_h$	0x0$_h$	0x7$_h$
		ADC_AENC_VN	0x0$_h$	0x0$_h$	0x7$_h$
		ADC_AENC_WY	0x0$_h$	0x0$_h$	0x7$_h$
RW	0x0D$_h$	AENC_0_SCALE_OFFSET			
		AENC_0_OFFSET	0x0$_h$	0x0$_h$	0xFFFF$_h$
		AENC_0_SCALE	0x100$_h$	-0x8000$_h$	0x7FFF$_h$
RW	0x0E$_h$	AENC_1_SCALE_OFFSET			

		AENC_1_OFFSET	$0x0_h$	$0x0_h$	$0xFFFF_h$
		AENC_1_SCALE	$0x100_h$	$-0x8000_h$	$0x7FFF_h$
RW	$0x0F_h$	AENC_2_SCALE_ OFFSET			
		AENC_2_OFFSET	$0x0_h$	$0x0_h$	$0xFFFF_h$
		AENC_2_SCALE	$0x100_h$	$-0x8000_h$	$0x7FFF_h$
RW	$0x11_h$	AENC_SELECT			
		AENC_0_SELECT	$0x0_h$	$0x0_h$	$0x2_h$
		AENC_1_SELECT	$0x1_h$	$0x0_h$	$0x2_h$
		AENC_2_SELECT	$0x2_h$	$0x0_h$	$0x2_h$
R	$0x12_h$	ADC_IWY_IUX			
		ADC_IUX	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
		ADC_IWY	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
R	$0x13_h$	ADC_IV			
		ADC_IV	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
R	$0x15_h$	AENC_WY_UX			
		AENC_UX	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
		AENC_WY	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
R	$0x16_h$	AENC_VN			
		AENC_VN	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
RW	$0x17_h$	PWM_POLARITIES			
		PWM_POLARITIES[0]	$0x0_h$	$0x0_h$	$0x1_h$
		PWM_POLARITIES[1]	$0x0_h$	$0x0_h$	$0x1_h$
RW	$0x18_h$	PWM_MAXCNT			
		PWM_MAXCNT	$0xF9F_h$	$0x0_h$	$0xFFFF_h$
RW	$0x19_h$	PWM_BBM_H_BBM_L			
		PWM_BBM_L	$0x14_h$	$0x0_h$	$0xFF_h$
		PWM_BBM_H	$0x14_h$	$0x0_h$	$0xFF_h$
RW	$0x1A_h$	PWM_SV_CHOP			
		PWM_CHOP	$0x0_h$	$0x0_h$	$0x7_h$
		PWM_SV	$0x0_h$	$0x0_h$	$0x1_h$
RW	$0x1B_h$	MOTOR_TYPE_N_ POLE_PAIRS			
		N_POLE_PAIRS	$0x1_h$	$0x1_h$	$0xFFFF_h$
		MOTOR_TYPE	$0x0_h$	$0x0_h$	$0x3_h$
RW	$0x1C_h$	PHI_E_EXT			

		PHI_E_EXT	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
RW	$0x1F_h$	OPENLOOP_MODE			
		OPENLOOP_PHI_DIRECTION	$0x0_h$	$0x0_h$	$0x1_h$
RW	$0x20_h$	OPENLOOP_ACCELERATION			
		OPENLOOP_ACCELERATION	$0x0_h$	$0x0_h$	$0xFFFFF_h$
RW	$0x21_h$	OPENLOOP_VELOCITY_TARGET			
		OPENLOOP_VELOCITY_TARGET	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
RW	$0x22_h$	OPENLOOP_VELOCITY_ACTUAL			
		OPENLOOP_VELOCITY_ACTUAL	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
RWI	$0x23_h$	OPENLOOP_PHI			
		OPENLOOP_PHI	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
RW	$0x24_h$	UQ_UD_EXT			
		UD_EXT	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
		UQ_EXT	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
RW	$0x25_h$	ABN_DECODER_MODE			
		apol	$0x0_h$	$0x0_h$	$0x1_h$
		bpol	$0x0_h$	$0x0_h$	$0x1_h$
		npol	$0x0_h$	$0x0_h$	$0x1_h$
		use_abn_as_n	$0x0_h$	$0x0_h$	$0x1_h$
		cln	$0x0_h$	$0x0_h$	$0x1_h$
		direction	$0x0_h$	$0x0_h$	$0x1_h$
RW	$0x26_h$	ABN_DECODER_PPR			
		ABN_DECODER_PPR	$0x10000_h$	$0x0_h$	$0xFFFFF_h$
RW	$0x27_h$	ABN_DECODER_COUNT			
		ABN_DECODER_COUNT	$0x0_h$	$0x0_h$	$0xFFFFF_h$
RW	$0x28_h$	ABN_DECODER_COUNT_N			

		ABN_DECODER_COUNT_N	0x0$_h$	0x0$_h$	0xFFFFFF$_h$
RW	0x29$_h$	ABN_DECODER_PHI_E_PHI_M_OFFSET			
		ABN_DECODER_PHI_M_OFFSET	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		ABN_DECODER_PHI_E_OFFSET	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
R	0x2A$_h$	ABN_DECODER_PHI_E_PHI_M			
		ABN_DECODER_PHI_M	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		ABN_DECODER_PHI_E	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
RW	0x2C$_h$	ABN_2_DECODER_MODE			
		apol	0x0$_h$	0x0$_h$	0x1$_h$
		bpol	0x0$_h$	0x0$_h$	0x1$_h$
		npol	0x0$_h$	0x0$_h$	0x1$_h$
		use_abn_as_n	0x0$_h$	0x0$_h$	0x1$_h$
		cln	0x0$_h$	0x0$_h$	0x1$_h$
		direction	0x0$_h$	0x0$_h$	0x1$_h$
RW	0x2D$_h$	ABN_2_DECODER_PPR			
		ABN_2_DECODER_PPR	0x10000$_h$	0x1$_h$	0xFFFFFF$_h$
RW	0x2E$_h$	ABN_2_DECODER_COUNT			
		ABN_2_DECODER_COUNT	0x0$_h$	0x0$_h$	0xFFFFFF$_h$
RW	0x2F$_h$	ABN_2_DECODER_COUNT_N			
		ABN_2_DECODER_COUNT_N	0x0$_h$	0x0$_h$	0xFFFFFF$_h$
RW	0x30$_h$	ABN_2_DECODER_PHI_M_OFFSET			
		ABN_2_DECODER_PHI_M_OFFSET	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
R	0x31$_h$	ABN_2_DECODER_PHI_M			

		ABN_2_DECODER_PHI_M	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
RW	0x33$_h$	HALL_MODE			
		polarity	0x0$_h$	0x0$_h$	0x1$_h$
		synchronous PWM sampling	0x0$_h$	0x0$_h$	0x1$_h$
		interpolation	0x0$_h$	0x0$_h$	0x1$_h$
		direction	0x0$_h$	0x0$_h$	0x1$_h$
		HALL_BLANK	0x0$_h$	0x0$_h$	0xFFF$_h$
RW	0x34$_h$	HALL_POSITION_060_000			
		HALL_POSITION_000	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		HALL_POSITION_060	0x2AAA$_h$	-0x8000$_h$	0x7FFF$_h$
RW	0x35$_h$	HALL_POSITION_180_120			
		HALL_POSITION_120	0x5555$_h$	-0x8000$_h$	0x7FFF$_h$
		HALL_POSITION_180	-0x8000$_h$	-0x8000$_h$	0x7FFF$_h$
RW	0x36$_h$	HALL_POSITION_300_240			
		HALL_POSITION_240	-0x5556$_h$	-0x8000$_h$	0x7FFF$_h$
		HALL_POSITION_300	-0x2AAB$_h$	-0x8000$_h$	0x7FFF$_h$
RW	0x37$_h$	HALL_PHI_E_PHI_M_OFFSET			
		HALL_PHI_M_OFFSET	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		HALL_PHI_E_OFFSET	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
RW	0x38$_h$	HALL_DPHI_MAX			
		HALL_DPHI_MAX	0x2AAA$_h$	0x0$_h$	0xFFFF$_h$
R	0x39$_h$	HALL_PHI_E_INTERPOLATED_PHI_E			
		HALL_PHI_E	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		HALL_PHI_E_INTERPOLATED	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
R	0x3A$_h$	HALL_PHI_M			
		HALL_PHI_M	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
RW	0x3B$_h$	AENC_DECODER_MODE			
		AENC_DECODER_MODE[0]	0x0$_h$	0x0$_h$	0x1$_h$

		AENC_DECODER_MODE[12]	$0x0_h$	$0x0_h$	$0x1_h$
RW	$0x3C_h$	AENC_DECODER_N_THRESHOLD			
		AENC_DECODER_N_THRESHOLD	$0x0_h$	$0x0_h$	$0xFFFF_h$
R	$0x3D_h$	AENC_DECODER_PHI_A_RAW			
		AENC_DECODER_PHI_A_RAW	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
RW	$0x3E_h$	AENC_DECODER_PHI_A_OFFSET			
		AENC_DECODER_PHI_A_OFFSET	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
R	$0x3F_h$	AENC_DECODER_PHI_A			
		AENC_DECODER_PHI_A	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
RW	$0x40_h$	AENC_DECODER_PPR			
		AENC_DECODER_PPR	$0x1_h$	$-0x8000_h$	$0x7FFF_h$
R	$0x41_h$	AENC_DECODER_COUNT			
		AENC_DECODER_COUNT	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
RW	$0x42_h$	AENC_DECODER_COUNT_N			
		AENC_DECODER_COUNT_N	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
RW	$0x45_h$	AENC_DECODER_PHI_E_PHI_M_OFFSET			
		AENC_DECODER_PHI_M_OFFSET	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
		AENC_DECODER_PHI_E_OFFSET	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
R	$0x46_h$	AENC_DECODER_PHI_E_PHI_M			
		AENC_DECODER_PHI_M	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
		AENC_DECODER_PHI_E	$0x0_h$	$-0x8000_h$	$0x7FFF_h$

RW	$0x4D_h$	CONFIG_DATA			
		biquad_x_a_1	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_x_a_2	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_x_b_0	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_x_b_1	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_x_b_2	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_x_enable	$0x0_h$	$0x0_h$	$0x1_h$
		biquad_v_a_1	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_v_a_2	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_v_b_0	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_v_b_1	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_v_b_2	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_v_enable	$0x0_h$	$0x0_h$	$0x1_h$
		biquad_t_a_1	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_t_a_2	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_t_b_0	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_t_b_1	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_t_b_2	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_t_enable	$0x0_h$	$0x0_h$	$0x1_h$
		biquad_f_a_1	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_f_a_2	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_f_b_0	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_f_b_1	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_f_b_2	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		biquad_f_enable	$0x0_h$	$0x0_h$	$0x1_h$
		feed_forward_ velocity_gain	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		feed_forward_ velocity_filter_ constant	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		feed_forward_ torque_gain	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		feed_forward_ torgue_filter_ constant	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
		VELOCITY_METER_ PPTM_MIN_POS_DEV	$0x0_h$	$0x0_h$	$0xFFFF_h$

		ref_switch_config	$0x0_h$	$0x0_h$	$0xFFFF_h$
		SINGLE_PIN_IF_CFG	$0x0_h$	$0x0_h$	$0xFF_h$
		SINGLE_PIN_IF_ STATUS	$0x0_h$	$0x0_h$	$0xFFFF_h$
		SINGLE_PIN_IF_ OFFSET	$0x0_h$	$0x0_h$	$0xFFFF_h$
		SINGLE_PIN_IF_ SCALE	$0x0_h$	$-0x7FFF_h$	$0x7FFF_h$
		CURRENT_P_nQ8.8_ Q4.12	$0x0_h$	$0x0_h$	$0x1_h$
		CURRENT_I_nQ8.8_ Q4.12	$0x0_h$	$0x0_h$	$0x1_h$
		VELOCITY_P_nQ8.8_ Q4.12	$0x0_h$	$0x0_h$	$0x1_h$
		VELOCITY_I_nQ8.8_ Q4.12	$0x0_h$	$0x0_h$	$0x1_h$
		POSITION_P_nQ8.8_ Q4.12	$0x0_h$	$0x0_h$	$0x1_h$
		POSITION_I_nQ8.8_ Q4.12	$0x0_h$	$0x0_h$	$0x1_h$
RW	$0x4E_h$	CONFIG_ADDR			
		CONFIG_ADDR	$0x0_h$	$0x1_h$	$0x3E_h$
RW	$0x50_h$	VELOCITY_ SELECTION			
		VELOCITY_ SELECTION	$0x0_h$	$0x0_h$	$0xC_h$
		VELOCITY_METER_ SELECTION	$0x0_h$	$0x0_h$	$0x1_h$
RW	$0x51_h$	POSITION_ SELECTION			
		POSITION_ SELECTION	$0x0_h$	$0x0_h$	$0xC_h$
RW	$0x52_h$	PHI_E_SELECTION			
		PHI_E_SELECTION	$0x0_h$	$0x0_h$	$0x7_h$
R	$0x53_h$	PHI_E			
		PHI_E	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
RW	$0x54_h$	PID_FLUX_P_FLUX_I			
		PID_FLUX_I	$0x0_h$	$0x0_h$	$0x7FFF_h$
		PID_FLUX_P	$0x0_h$	$0x0_h$	$0x7FFF_h$
RW	$0x56_h$	PID_TORQUE_P_ TORQUE_I			

		PID_TORQUE_I	0x0$_h$	0x0$_h$	0x7FFF$_h$
		PID_TORQUE_P	0x0$_h$	0x0$_h$	0x7FFF$_h$
RW	0x58$_h$	PID_VELOCITY_P_VELOCITY_I			
		PID_VELOCITY_I	0x0$_h$	0x0$_h$	0x7FFF$_h$
		PID_VELOCITY_P	0x0$_h$	0x0$_h$	0x7FFF$_h$
RW	0x5A$_h$	PID_POSITION_P_POSITION_I			
		PID_POSITION_I	0x0$_h$	0x0$_h$	0x7FFF$_h$
		PID_POSITION_P	0x0$_h$	0x0$_h$	0x7FFF$_h$
RW	0x5D$_h$	PIDOUT_UQ_UD_LIMITS			
		PIDOUT_UQ_UD_LIMITS	0x5A81$_h$	0x0$_h$	0x7FFF$_h$
RW	0x5E$_h$	PID_TORQUE_FLUX_LIMITS			
		PID_TORQUE_FLUX_LIMITS	0x7FFF$_h$	0x0$_h$	0x7FFF$_h$
RW	0x60$_h$	PID_VELOCITY_LIMIT			
		PID_VELOCITY_LIMIT	0x7FFFFFFF$_h$	0x0$_h$	0xFFFFFFFF$_h$
RW	0x61$_h$	PID_POSITION_LIMIT_LOW			
		PID_POSITION_LIMIT_LOW	-0x7FFFFFFF$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
RW	0x62$_h$	PID_POSITION_LIMIT_HIGH			
		PID_POSITION_LIMIT_HIGH	0x7FFFFFFF$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
RW	0x63$_h$	MODE_RAMP_MODE_MOTION			
		MODE_MOTION	0x0$_h$	0x0$_h$	0xF$_h$
		MODE_PID_SMPL	0x0$_h$	0x0$_h$	0x7F$_h$
		MODE_PID_TYPE	0x0$_h$	0x0$_h$	0x1$_h$
RW	0x64$_h$	PID_TORQUE_FLUX_TARGET			
		PID_FLUX_TARGET	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		PID_TORQUE_TARGET	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
RW	0x65$_h$	PID_TORQUE_FLUX_OFFSET			

		PID_FLUX_OFFSET	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		PID_TORQUE_OFFSET	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
RW	0x66$_h$	PID_VELOCITY_TARGET			
		PID_VELOCITY_TARGET	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
RW	0x67$_h$	PID_VELOCITY_OFFSET			
		PID_VELOCITY_OFFSET	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
RW	0x68$_h$	PID_POSITION_TARGET			
		PID_POSITION_TARGET	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
R	0x69$_h$	PID_TORQUE_FLUX_ACTUAL			
		PID_FLUX_ACTUAL	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		PID_TORQUE_ACTUAL	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
R	0x6A$_h$	PID_VELOCITY_ACTUAL			
		PID_VELOCITY_ACTUAL	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
RW	0x6B$_h$	PID_POSITION_ACTUAL			
		PID_POSITION_ACTUAL	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
R	0x6C$_h$	PID_ERROR_DATA			
		PID_TORQUE_ERROR	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		PID_FLUX_ERROR	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		PID_VELOCITY_ERROR	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		PID_POSITION_ERROR	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		PID_TORQUE_ERROR_SUM	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		PID_FLUX_ERROR_SUM	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		PID_VELOCITY_ERROR_SUM	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$

		PID_POSITION_ERROR_SUM	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
RW	0x6D$_h$	PID_ERROR_ADDR			
		PID_ERROR_ADDR	0x0$_h$	0x0$_h$	0x7$_h$
RW	0x6E$_h$	INTERIM_DATA			
		PIDIN_TARGET_TORQUE	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		PIDIN_TARGET_FLUX	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		PIDIN_TARGET_VELOCITY	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		PIDIN_TARGET_POSITION	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		PIDOUT_TARGET_TORQUE	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		PIDOUT_TARGET_FLUX	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		PIDOUT_TARGET_VELOCITY	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		PIDOUT_TARGET_POSITION	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		FOC_IUX	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		FOC_IWY	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		FOC_IV	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		FOC_IA	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		FOC_IB	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		FOC_ID	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		FOC_IQ	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		FOC_UD	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		FOC_UQ	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		FOC_UD_LIMITED	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		FOC_UQ_LIMITED	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		FOC_UA	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		FOC_UB	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		FOC_UUX	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		FOC_UWY	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		FOC_UV	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		PWM_UX	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		PWM_WY	0x0$_h$	-0x8000$_h$	0x7FFF$_h$
		PWM_V	0x0$_h$	-0x8000$_h$	0x7FFF$_h$

ADC_I_0	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
ADC_I_1	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
PID_FLUX_ACTUAL_DIV256	$0x0_h$	$-0x80_h$	$0x7F_h$
PID_TORQUE_ACTUAL_DIV256	$0x0_h$	$-0x80_h$	$0x7F_h$
PID_FLUX_TARGET_DIV256	$0x0_h$	$-0x80_h$	$0x7F_h$
PID_TORQUE_TARGET_DIV256	$0x0_h$	$-0x80_h$	$0x7F_h$
PID_TORQUE_ACTUAL	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
PID_TORQUE_TARGET	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
PID_FLUX_ACTUAL	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
PID_FLUX_TARGET	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
PID_VELOCITY_ACTUAL_DIV256	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
PID_VELOCITY_TARGET_DIV256	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
PID_VELOCITY_ACTUAL_LSB	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
PID_VELOCITY_TARGET_LSB	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
PID_POSITION_ACTUAL_DIV256	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
PID_POSITION_TARGET_DIV256	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
PID_POSITION_ACTUAL_LSB	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
PID_POSITION_TARGET_LSB	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
FF_VELOCITY	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
FF_TORQUE	$0x0_h$	$-0x8000_h$	$0x7FFF_h$
ACTUAL_VELOCITY_PPTM	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$
REF_SWITCH_STATUS	$0x0_h$	$0x0_h$	$0xFFFF_h$
HOME_POSITION	$0x0_h$	$-0x80000000_h$	$0x7FFFFFFF_h$

		LEFT_POSITION	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		RIGHT_POSITION	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		SINGLE_PIN_IF_ TARGET_TORQUE	0x0$_h$	-0x8000$_h$	0x8000$_h$
		SINGLE_PIN_IF_ PWM_DUTY_CYCLE	0x0$_h$	-0x8000$_h$	0x8000$_h$
		SINGLE_PIN_IF_ TARGET_VELOCITY	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
		SINGLE_PIN_IF_ TARGET_POSITION	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
RW	0x6F$_h$	INTERIM_ADDR			
		INTERIM_ADDR	0x0$_h$	0x0$_h$	0xD7$_h$
RW	0x75$_h$	ADC_VM_LIMITS			
		ADC_VM_LIMIT_LOW	0xFFFF$_h$	0x0$_h$	0xFFFF$_h$
		ADC_VM_LIMIT_HIGH	0xFFFF$_h$	0x0$_h$	0xFFFF$_h$
R	0x76$_h$	TMC4671_INPUTS_ RAW			
		A of ABN_RAW	0x0$_h$	0x0$_h$	0x1$_h$
		B of ABN_RAW	0x0$_h$	0x0$_h$	0x1$_h$
		N of ABN_RAW	0x0$_h$	0x0$_h$	0x1$_h$
		-	0x0$_h$	0x0$_h$	0x1$_h$
		A of ABN_2_RAW	0x0$_h$	0x0$_h$	0x1$_h$
		B of ABN_2_RAW	0x0$_h$	0x0$_h$	0x1$_h$
		N of ABN_2_RAW	0x0$_h$	0x0$_h$	0x1$_h$
		-	0x0$_h$	0x0$_h$	0x1$_h$
		HALL_UX of HALL_ RAW	0x0$_h$	0x0$_h$	0x1$_h$
		HALL_V of HALL_RAW	0x0$_h$	0x0$_h$	0x1$_h$
		HALL_WY of HALL_ RAW	0x0$_h$	0x0$_h$	0x1$_h$
		-	0x0$_h$	0x0$_h$	0x1$_h$
		REF_SW_R_RAW	0x0$_h$	0x0$_h$	0x1$_h$
		REF_SW_H_RAW	0x0$_h$	0x0$_h$	0x1$_h$
		REF_SW_L_RAW	0x0$_h$	0x0$_h$	0x1$_h$
		ENABLE_IN_RAW	0x0$_h$	0x0$_h$	0x1$_h$
		STP of DIRSTP_RAW	0x0$_h$	0x0$_h$	0x1$_h$
		DIR of DIRSTP_RAW	0x0$_h$	0x0$_h$	0x1$_h$

		PWM_IN_RAW	0x0$_h$	0x0$_h$	0x1$_h$
		-	0x0$_h$	0x0$_h$	0x1$_h$
		HALL_UX_FILT	0x0$_h$	0x0$_h$	0x1$_h$
		HALL_V_FILT	0x0$_h$	0x0$_h$	0x1$_h$
		HALL_WY_FILT	0x0$_h$	0x0$_h$	0x1$_h$
		-	0x0$_h$	0x0$_h$	0x1$_h$
		-	0x0$_h$	0x0$_h$	0x1$_h$
		-	0x0$_h$	0x0$_h$	0x1$_h$
		-	0x0$_h$	0x0$_h$	0x1$_h$
		-	0x0$_h$	0x0$_h$	0x1$_h$
		PWM_IDLE_L_RAW	0x0$_h$	0x0$_h$	0x1$_h$
		PWM_IDLE_H_RAW	0x0$_h$	0x0$_h$	0x1$_h$
		-	0x0$_h$	0x0$_h$	0x1$_h$
		-	0x0$_h$	0x0$_h$	0x1$_h$
R	0x77$_h$	TMC4671_OUTPUTS_RAW			
		TMC4671_OUTPUTS_RAW[0]	0x0$_h$	0x0$_h$	0x1$_h$
		TMC4671_OUTPUTS_RAW[1]	0x0$_h$	0x0$_h$	0x1$_h$
		TMC4671_OUTPUTS_RAW[2]	0x0$_h$	0x0$_h$	0x1$_h$
		TMC4671_OUTPUTS_RAW[3]	0x0$_h$	0x0$_h$	0x1$_h$
		TMC4671_OUTPUTS_RAW[4]	0x0$_h$	0x0$_h$	0x1$_h$
		TMC4671_OUTPUTS_RAW[5]	0x0$_h$	0x0$_h$	0x1$_h$
		TMC4671_OUTPUTS_RAW[6]	0x0$_h$	0x0$_h$	0x1$_h$
		TMC4671_OUTPUTS_RAW[7]	0x0$_h$	0x0$_h$	0x1$_h$
RW	0x78$_h$	STEP_WIDTH			
		STEP_WIDTH	0x0$_h$	-0x80000000$_h$	0x7FFFFFFF$_h$
RW	0x79$_h$	UART_BPS			
		UART_BPS	0x00009600$_h$	0x0$_h$	0xFFFFFF$_h$
RW	0x7B$_h$	GPIO_dsADCI_CONFIG			

		GPIO_dsADCI_CONFIG[0]	0x0$_h$	0x0$_h$	0x1$_h$
		GPIO_dsADCI_CONFIG[1]	0x0$_h$	0x0$_h$	0x1$_h$
		GPIO_dsADCI_CONFIG[2]	0x0$_h$	0x0$_h$	0x1$_h$
		GPIO_dsADCI_CONFIG[3]	0x0$_h$	0x0$_h$	0x1$_h$
		GPIO_dsADCI_CONFIG[4]	0x0$_h$	0x0$_h$	0x1$_h$
		GPIO_dsADCI_CONFIG[5]	0x0$_h$	0x0$_h$	0x1$_h$
		GPIO_dsADCI_CONFIG[6]	0x0$_h$	0x0$_h$	0x1$_h$
		GPO	0x0$_h$	0x0$_h$	0xFF$_h$
		GPI	0x0$_h$	0x0$_h$	0xFF$_h$
RW	0x7C$_h$	STATUS_FLAGS			
		STATUS_FLAGS[0]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[1]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[2]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[3]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[4]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[5]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[6]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[7]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[8]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[9]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[10]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[11]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[12]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[13]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[14]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[15]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[16]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[17]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[18]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[19]	0x0$_h$	0x0$_h$	0x1$_h$
		STATUS_FLAGS[20]	0x0$_h$	0x0$_h$	0x1$_h$

		STATUS_FLAGS[21]	$0x0_h$	$0x0_h$	$0x1_h$
		STATUS_FLAGS[22]	$0x0_h$	$0x0_h$	$0x1_h$
		STATUS_FLAGS[23]	$0x0_h$	$0x0_h$	$0x1_h$
		STATUS_FLAGS[24]	$0x0_h$	$0x0_h$	$0x1_h$
		STATUS_FLAGS[25]	$0x0_h$	$0x0_h$	$0x1_h$
		STATUS_FLAGS[26]	$0x0_h$	$0x0_h$	$0x1_h$
		STATUS_FLAGS[27]	$0x0_h$	$0x0_h$	$0x1_h$
		STATUS_FLAGS[28]	$0x0_h$	$0x0_h$	$0x1_h$
		STATUS_FLAGS[29]	$0x0_h$	$0x0_h$	$0x1_h$
		STATUS_FLAGS[30]	$0x0_h$	$0x0_h$	$0x1_h$
		STATUS_FLAGS[31]	$0x0_h$	$0x0_h$	$0x1_h$
RW	$0x7D_h$	STATUS_MASK			
		WARNING_MASK	$0x0_h$	$0x0_h$	$0xFFFFFFFF_h$

8 Pinout

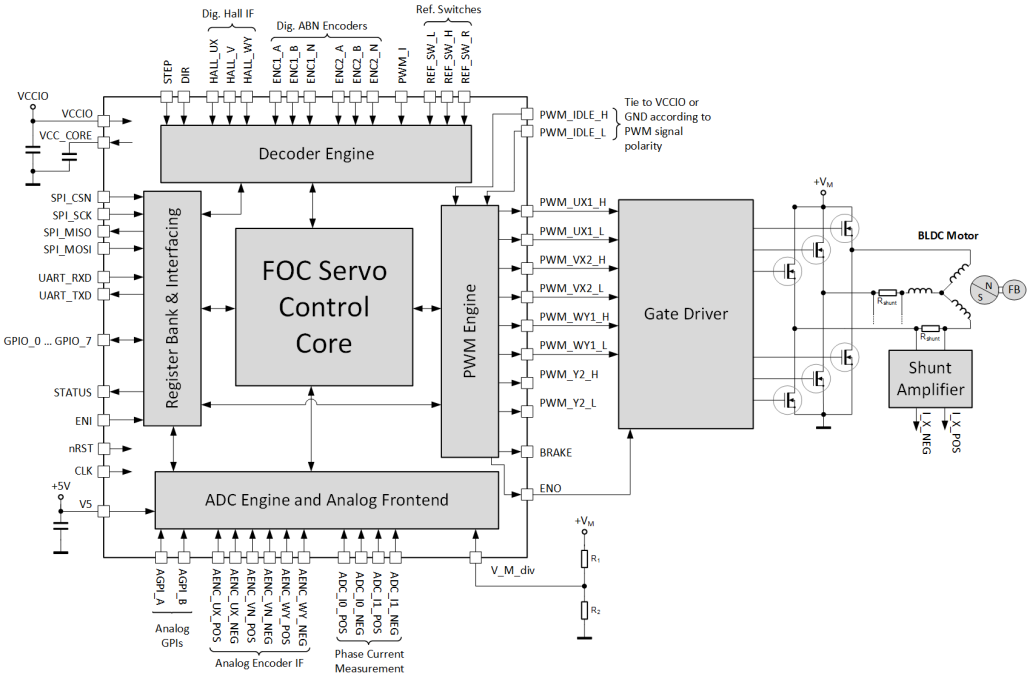


Figure 40: TMC4671 Pinout with Three-Phase Power Stage and BLDC Motor

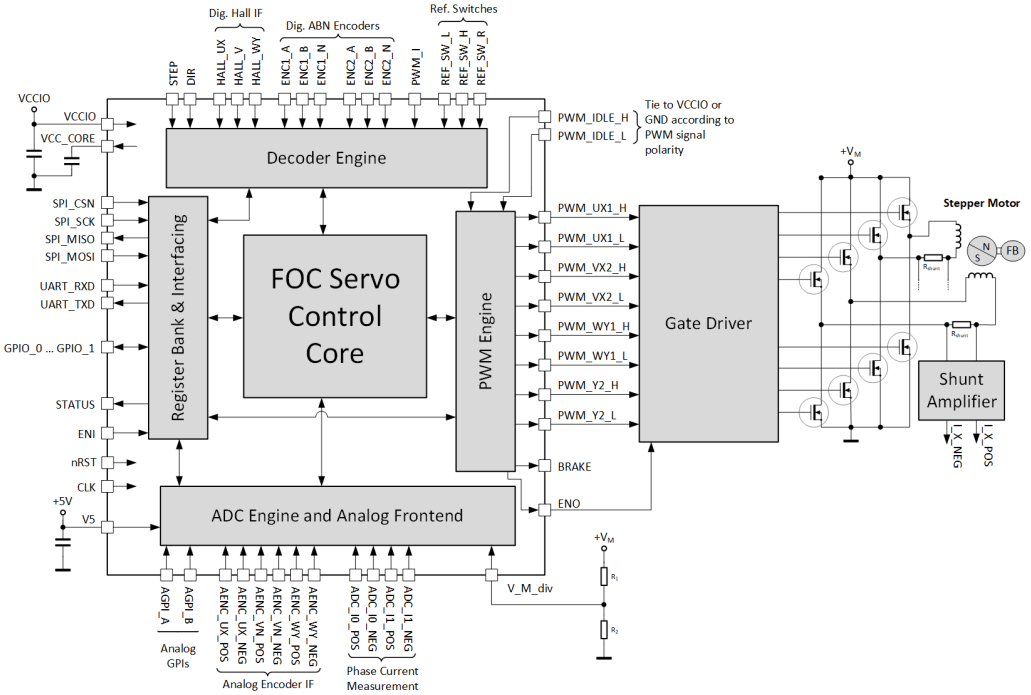


Figure 41: TMC4671 Pinout with Stepper Motor

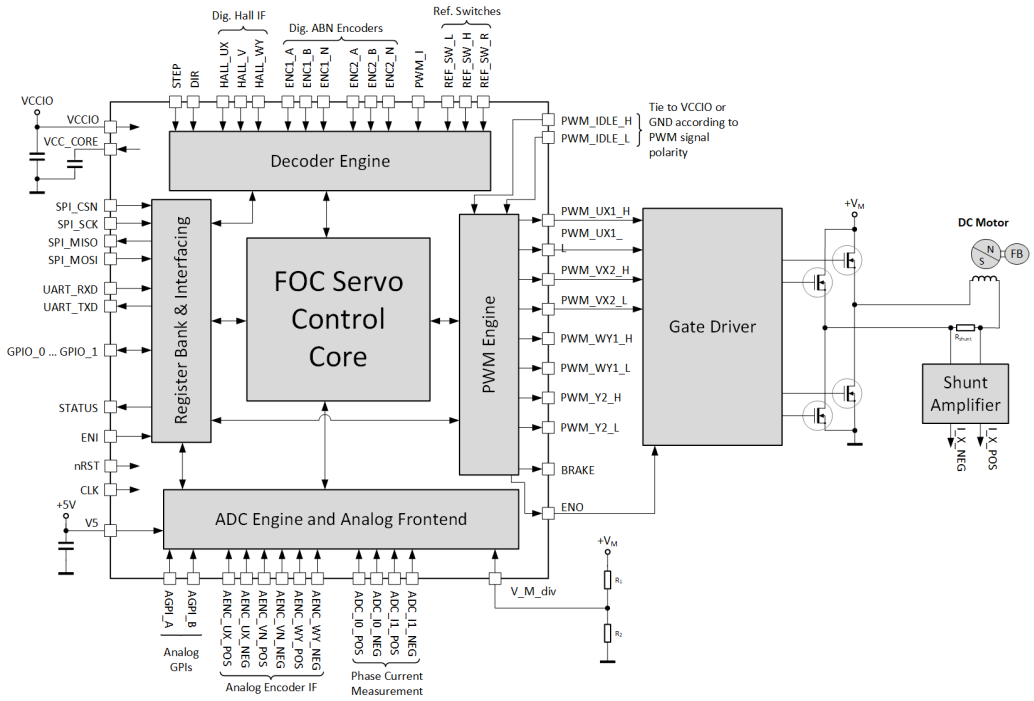


Figure 42: TMC4671 Pinout with DC Motor or Voice Coil

NOTE

All power supply pins (V_{CCIO}, V_{CCCORE}) must be connected.

All ground pins (GND, GNDA, …) must be connected.

Analog inputs (AI) are 5V single-ended or differential inputs (Input range: GNDA to V5). Use voltage dividers or operational amplifiers to scale down higher input voltages.

Digital inputs (I) resp. (IO) are 3.3V single-ended inputs.

IO	Description
AI	Analog input, 5V
I	Digital input, 3.3V
IO	Digital input or digital output, direction programmable, 3.3V
O	Digital output, 3.3V

Table 34: Pin Type Definition

9 TMC4671 Pin Table

Name	Pin	IO	Description
nRST	50	I	Active low reset input
CLK	51	I	Clock input; must be 25MHz for correct timing
TEST	54	I	TEST input, must be connected to GND
ENI	55	I	Enable input; if high, controllers and PWM are enabled
ENO	32	O	Enable output; feeds through ENI when CLK is applied and IC is not in reset condition
STATUS	12	O	Output for interrupt of CPU (warning and status change)
SPI_nSCS	6	I	SPI active low chip select input
SPI_SCK	7	I	SPI clock input
SPI_MOSI	8	I	SPI main node out subnode input
SPI_MISO	9	O	SPI main node in subnode output, high impedance, when SPI_nSCS = '1'
UART_RXD	10	I	UART receive data RxD for in-system-user communication channel
UART_TXD	11	O	UART transmit data TXD for in-system-user communication channel
PWM_I	58	I	PWM input for target value generation
DIR	56	I	Direction input of step-direction interface
STP	57	I	Step pulse input for step-direction interface
HALL_UX	38	I	Digital hall input H1 for 3-phase (U) or 2-phase (X)
HALL_V	37	I	Digital hall input H2 for 3-phase (V)
HALL_WY	36	I	Digital hall input H3 for 3-phase (W) or 2-phase (Y)
ENC_A	35	I	A input of incremental encoder
ENC_B	34	I	B input of incremental encoder
ENC_N	33	I	N input of incremental encoder
ENC2_A	64	I	A input of incremental encoder
ENC2_B	65	I	B input of incremental encoder
ENC2_N	66	I	N input of incremental encoder
REF_L	67	I	Left (L) reference switch
REF_H	68	I	Home (H) reference switch
REF_R	69	I	Right (R) reference switch
ADC_I0_POS	16	AI	Positive input for phase current signal measurement I0 (I_U, I_X)

Name	Pin	IO	Description
ADC_I0_NEG	17	AI	Negative input for phase current signal measurement I0 (I_U, I_X)
ADC_I1_POS	18	AI	Positive input for phase current signal measurement I1 (I_V, I_W, I_Y)
ADC_I1_NEG	19	AI	Negative input for phase current signal measurement I1 (I_V, I_W, I_Y)
ADC_VM	20	AI	Analog input for motor supply voltage divider (VM) measurement
AGPI_A	21	AI	Analog general purpose input A (analog GPI)
AGPI_B	22	AI	Analog general purpose input B (analog GPI)
AENC_UX_POS	25	AI	Positive analog input for hall or analog encoder signal, 3-phase (U) or 2-phase (X (cos))
AENC_UX_NEG	26	AI	Negative analog input for hall or analog encoder signal, 3-phase (U) or 2-phase (X (cos))
AENC_VN_POS	27	AI	Positive analog input for hall or analog encoder signal, 3-phase (V) or 2-phase (N)
AENC_VN_NEG	28	AI	Negative analog input for hall or analog encoder signal, 3-phase (V) or 2-phase (N)
AENC_WY_POS	29	AI	Positive analog input for hall or analog encoder signal, 3-phase (W) or 2-phase (Y (sin))
AENC_WY_NEG	30	AI	Negative analog input for hall or analog encoder signal, 3-phase (W) or 2-phase (Y (sin))
GPIO0 / ADC_I0_MCD	70	IO	GPIO or ΔΣ-Demodulator clock input MCLKI, clock output MCLKO, or single bit DAC output MDAC for ADC_I_0
GPIO1 / ADC_I1_MCD	71	IO	GPIO or ΔΣ-Demodulator clock input MCLKI, clock output MCLKO, or single bit DAC output MDAC for ADC_I_1
GPIO2 / ADC_VM_MCD	74	IO	GPIO or ΔΣ-Demodulator clock input MCLKI, clock output MCLKO, or single bit DAC output MDAC for ADC_VM_MCD
GPIO3 / AGPI_A_MCD / DBGSPI_nSCS	75	IO	GPIO or ΔΣ-Demodulator clock input MCLKI, clock output MCLKO, or single bit DAC output MDAC for AENC_UX_MCD, SPI debug port pin DBGSPI_nSCS
GPIO4 / AGPI_B_MCD / DBGSPI_SCK	76	IO	GPIO or ΔΣ-Demodulator clock input MCLKI, clock output MCLKO, or single bit DAC output MDAC for AENC_VN_MCD, SPI debug port pin DBGSPI_SCK
GPIO5 / AENC_UX_MCD / DBGSPI_MOSI	1	IO	GPIO or ΔΣ-Demodulator clock input MCLKI, clock output MCLKO, or single bit DAC output MDAC for AENC_WY_MCD, SPI debug port pin DBGSPI_MOSI

Name	Pin	IO	Description
GPIO6 / AENC_VN_MCD / DBGSPI_MISO	4	IO	GPIO or ΔΣ-Demodulator clock input MCLKI, clock output MCLKO, or single bit DAC output MDAC for AGPI_A_MCD, SPI debug port pin DBGSPI_MISO
GPIO7 / AENC_WY_MCD / DBGSPI_TRG	5	IO	GPIO or ΔΣ-Demodulator clock input MCLKI, clock output MCLKO, or single bit DAC output MDAC for AGPI_B_MCD, SPI debug port pin DBGSPI_TRG
PWM_IDLE_H	59	I	Idle level of high-side gate control signals (not used)
PWM_IDLE_L	60	I	Idle level of low-side gate control signals (not used)
PWM_UX1_H	39	O	High-side gate control output U (3-phase) resp. X1 (2-phase)
PWM_UX1_L	40	O	Low-side gate control output U (3-phase) resp. X1 (2-phase)
PWM_VX2_H	41	O	High-side gate control output V (3-phase) resp. X2 (2-phase)
PWM_VX2_L	42	O	Low-side gate control output V (3-phase) resp. X2 (2-phase)
PWM_WY1_H	46	O	High-side gate control output W (3-phase) resp. Y1 (2-phase)
PWM_WY1_L	47	O	Low-side gate control output W (3-phase) resp. Y1 (2-phase)
PWM_Y2_H	48	O	High-side gate control output Y2 (2-phase only)
PWM_Y2_L	49	O	Low-side gate control output Y2 (2-phase only)
BRAKE	31	O	Brake chopper control output signal

Table 35: Functional Pin Description

Feedback input pins not needed in target application can be left open or tied to GND.

Name	Pin	IO	Description
VCCIO1	2	3.3V	Digital IO supply voltage; use 100 nF decoupling capacitor
VCCIO2	13	3.3V	Digital IO supply voltage; use 100 nF decoupling capacitor
VCCIO3	43	3.3V	Digital IO supply voltage; use 100 nF decoupling capacitor
VCCIO4	52	3.3V	Digital IO supply voltage; use 100 nF decoupling capacitor
VCCIO5	61	3.3V	Digital IO supply voltage; use 100 nF decoupling capacitor
VCCIO6	72	3.3V	Digital IO supply voltage; use 100 nF decoupling capacitor
GNDIO1	3	0V	Digital IO ground
GNDIO2	14	0V	Digital IO ground
GNDIO3	44	0V	Digital IO ground
GNDIO4	53	0V	Digital IO ground
GNDIO5	62	0V	Digital IO ground
GNDIO6	73	0V	Digital IO ground
VCCCORE1	15	1.8V	Digital core supply voltage output; use 100 nF decoupling capacitor
VCCCORE2	45	1.8V	Digital core supply voltage output; use 100 nF decoupling capacitor
VCCCORE3	63	1.8V	Digital core supply voltage output; use 100 nF decoupling capacitor
V5	23	5V	Analog reference voltage
GNDA	24	0V	Analog reference ground
GNDPAD	–	0V	Bottom ground pad

Table 36: Supply Voltage Pins and Ground Pins

10 Electrical Characteristics

10.1 Absolute Maximum Ratings

The maximum ratings may not be exceeded under any circumstances. Operating the circuit at or near more than one maximum rating at a time for extended periods shall be avoided by application design.

Parameter	Symbol	Min	Max	Unit
Digital I/O supply voltage	V_{CCIO}		3.6	V
Logic input voltage	V_I		3.6	V
Maximum current drawn on V_{CCIO} with no load on pins	I_IO		70	mA
Maximum current drawn on V_{CCIO} with no load on pins and clock off	I_IO_0Hz		3	mA
Maximum current drawn on V5 at fCLK = 25MHz	I_V5		25	mA
Maximum current to/from digital pins and analog low voltage I/Os	IIO		10	mA
Junction temperature	TJ	-40	125	°C
Storage temperature	TSTG	-55	150	°C
ESD-protection for interface pins (human body model, HBM)	VESDAP		2	kV
ESD-protection for handling (human body model, HBM)	VESD1		2	kV
ADC input voltage	VAI	0	5	V
AENC input voltage	VAI	0	5.25	V

Table 37: Absolute Maximum Ratings

V_{CCCORE} is generated internally from V_{CCIO} and shall not be overpowered by external supply.

10.2 Electrical Characteristics

10.2.1 Operational Range

Parameter	Symbol	Min	Max	Unit
Junction temperature	TJ	-40	125	°C
Digital I/O 3.3V supply voltage	VIO3V	3.15	3.45	V
Core supply voltage	V_{CCCORE}	1.65	1.95	V

Table 38: Operational Range

The ΔΣ ADCs can operate in differential or single-ended mode. In differential mode, the differential input voltage range must be in between -2.5V and +2.5V. However, it is recommended to use the input voltage range from -1.25V to 1.25V, due to nonlinearity of ΔΣ ADCs. In single-ended mode, the operational

input range of the positive input channel should be between 0V and 2.5V. Recommended maximum input voltage is 1.25V.

10.2.2 Operational Conditions

Parameter	Symbol	Typ	Unit
Power dissipation with maximum current drawn on V_{CCIO} with no load on pins	P_V_{CCIO}	252	mW
Power dissipation with maximum current drawn on V5 at fCLK = 25MHz	P_V5	125	mW
Total power dissipation	P_MAX	377	mW

Table 39: Operational Range

10.2.3 DC Characteristics

DC characteristics contain the spread of values guaranteed within the specified supply voltage range, unless otherwise specified. Typical values represent the average value of all parts measured at +25 °C. Temperature variation also causes stray to some values. A device with typical values does not leave min/max range within the full temperature range.

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Input voltage low level	VINL	V_{CCIO} = 3.3V	-0.3		0.8	V
Input voltage high level	VINH	V_{CCIO} = 3.3V	2.3		3.6	V
Input with pull-down		V_{IN} = 3.3V	5	30	110	µA
Input with pull-up		V_{IN} = 0V	-110	-30	-5	µA
Input low current		V_{IN} = 0V	-10		10	µA
Input high current		V_{IN} = V_{CCIO}	-10		10	µA
Output voltage low level	VOUTL	V_{CCIO} = 3.3V	0.4			V
Output voltage high level	VOUTH	V_{CCIO} = 3.3V	2.64			V
Output driver strength standard	IOUT_DRV		4			mA
Input impedance of analog Input	R_ADC	TJ = 25°C	85	100	115	kΩ

Table 40: DC Characteristics

All I/O lines include Schmitt-Trigger inputs to enhance noise margin.

10.2.4 ABN Encoder Interface

Parameter	Symbol	Condition	Min	Typ	Max	Unit
A/B/N counting frequency (each channel)	f_{CNT}			2		MHz
A/B/N input low time	t_{ABNL}		160			ns
A/B/N input high time	t_{ABNH}		160			ns
A/B/N filtering time	$t_{FILTABN}$			120		ns

Table 41: ABN Characteristics

11 Sample Circuits

Consider electrical characteristics while designing electrical circuitry. Most sample circuits in this chapter are taken from the evalutation board for the TMC4671 (TMC4671-EVAL).

11.1 Supply Pins

Provide V_{CCIO} and V5 to the TMC4671. V_{CCCORE} is internally generated and needs just an external decoupling capacitor. Place one 100nF decoupling capacitor at every supply pin. Table 42 lists additionally needed decoupling capacitors.

Pin Name	Supply Voltage	Additional Cap.
V5	5V	4.7µF
V_{CCIO}	3.3V	4.7µF & 470nF
V_{CCCORE}	1.8V	None

Table 42: Additional Decoupling Capacitors for Supply Voltages

11.2 Clock and Reset Circuitry

The TMC4671 needs an external oscillator for correct operation at 25MHz.

11.3 Digital Encoder, Hall Sensor Interface, and Reference Switches

Digital encoders, hall sensors, and reference switches usually operate on a supply voltage of 5V. As the TMC4671 is usually operated at a V_{CCIO} Voltage of 3.3V, a protection circuit for the TMC4671 input pin is needed. In Figure 43, a sample circuit for the ENC_A signal is shown, which can be reused for all encoder and hall signals as well as for reference switch signals. Table 43 gives the parametrization of the components for different operations.

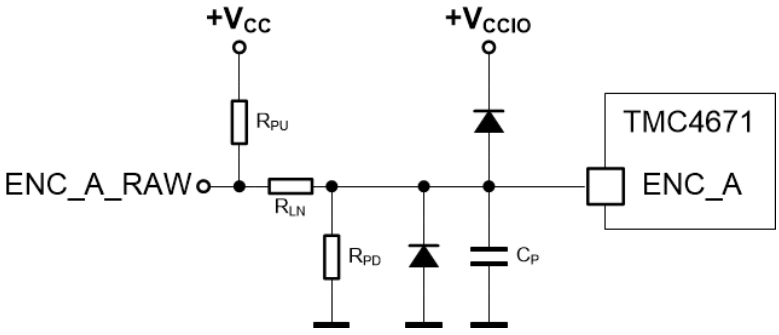


Figure 43: Sample Circuit for Interfacing of an Encoder Signal

Application	R_{PU}	R_{PD}	R_{LN}	C_P
5V encoder signal	4K7	n.c.	100R	100pF

Table 43: Reference Values for Circuitry Components

The raw signal (ENC_A_RAW) is divided by a voltage divider and filtered by a low-pass filter. A pull-up resistor is applied for open collector encoder output signals. Diodes protect the input pin (ENC_A) against overvoltage and undervoltage. The cutoff-frequency of the low-pass is:

$$f_c = \frac{1}{2\,\pi\,R_{PD}C_P} \tag{49}$$

11.4 Analog Frontend

Analog encoders encode the motor position into sinusoidal signals. These signals must be digitalized by the TMC4671 to determine the rotor position. The input voltage range depends on V5 input, which is usually 5V and GNDA (usually 0V). Due to nonlinearity issues of the ADC near input limits, an ADC input value from 1V to 4V is recommended. For a single-ended application, the sample circuit from Figure 44 can be used. All single-ended analog input pins (AGPI_A, AGPI_B, and ADC_VM) have their negative input value tied to GNDA internally. So, this sample circuit can also be used for them.

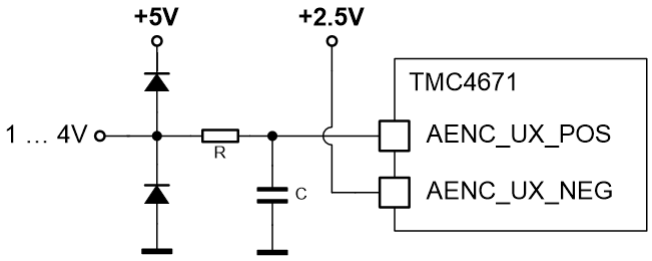


Figure 44: Sample Circuit for Interfacing of a Single-Ended Analog Signal

If the power stage and the TMC4671 share a common ground, the ADC_VM input signal can be generated by a voltage divider to scale the voltage down to the needed range.
If the analog encoder has differential output signals, these can be used without signal conditioning (no OP AMPs), when voltage range matches. Differential analog inputs can be used to digitize differential analog input signals with high common-mode voltage error suppression.

11.5 Phase Current Measurement

The TMC4671 requires two phase currents of a two-phase or three-phase motor to be measured. For a DC motor, only one current in the phase needs to be measured (see Figure 46). In the ADC engine, mapping of current signals to motor phases can be changed. Default setting is I0 for the current running into the motor in phase U for a three-phase motor and the current running into the motor from half-bridge X1 of a two-phase motor. Figure 45 and Figure 46 illustrate the currents to be measured and their positive direction.

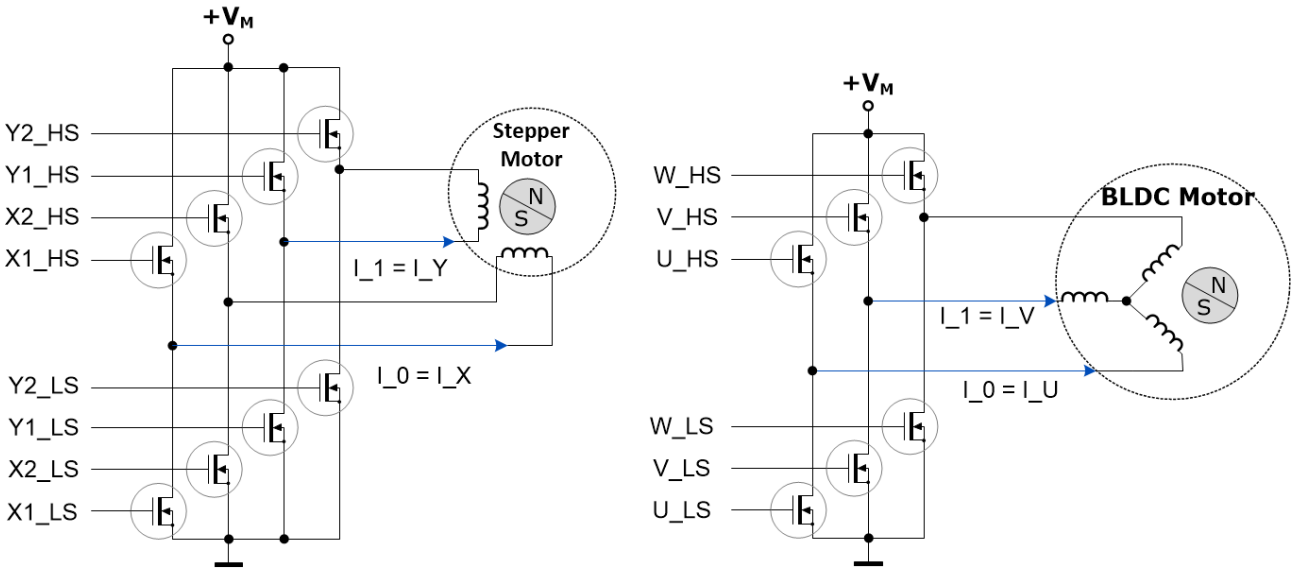


Figure 45: Phase Current Measurement: Current Directions for Two-Phase and Three-Phase Motors

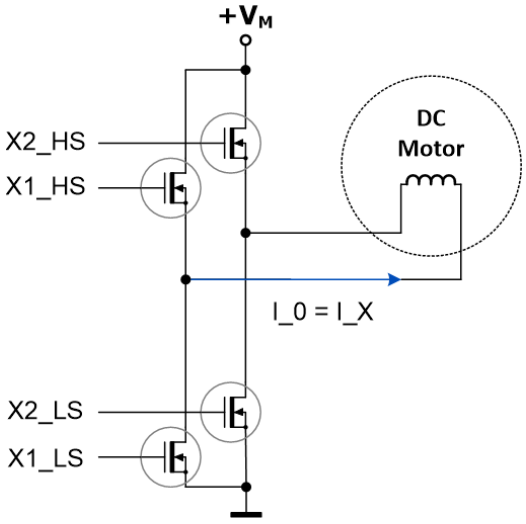


Figure 46: Phase Current Measurement: Current Direction for DC or Voice Coil Motor

There are two main options to measure the phase currents as described above. First option is to use a shunt resistor and a shunt amplifier like the LT1999 or the AD8418A. The other option is to use a real current sensor, which uses the hall effect or other magnetic effects to implement an isolated current measurement. Shunt measurement might be the more cost-effective solution for low voltage applications up to 100V, while current sensors are more useful at higher voltage levels.
In general, the sample circuit in Figure 47 can be used for shunt measurement circuitry. Consider design guidelines of shunt amplifier supplier additionally. TRINAMIC also supplies power stage boards with current shunt measurement circuitry (TMC-UPS10A/70V-EVAL). For current measurement, current sensors with voltage output can also be used. These can use the hall effect or other magnetic effects. Main concerns are bandwidth, accuracy, and measurement range.

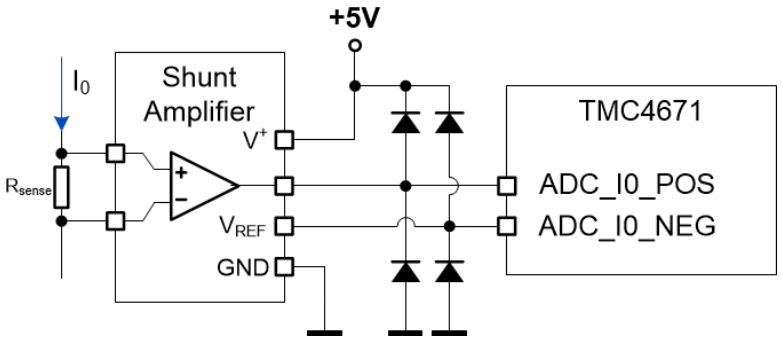


Figure 47: Current Shunt Amplifier Sample Circuit

11.6 Power Stage Interface

The TMC4671 is equipped with a configurable PWM engine to control various gate drivers. Gate driver switch signals can be matched to power stage needs. This includes signal polarities, frequency, BBM-times for low-side and high-side switches, and an enable signal. Consider gate driver circuitry, when connecting to the TMC4671.

12 Setup Guidelines

For easy setup of the TMC4671 on a given hardware platform like the TMC4671 evaluation-kit, follow these general guidelines to safely set up the system for various modes of operation.

NOTE	These guidelines fit to hardware platforms comparable to the TMC4671-evaluation kit. If system structure differs, adjust the configuration. Also make use of the RTMI adapter and the TMCL IDE to set up the system as it reduces commissioning time significantly.

Step 0: Setup of SPI communication
As a first step of the configuration of the TMC4671, test the SPI communication by reading and writing as example to the first registers to identify the silicon. If communication fails, inspect the CLK and nRST signals. For easy software setup, use the TMC API provided on the TRINAMIC website.
Step 1: Check connections
Access register TMC4671_INPUTS_RAW to see if all connected digital inputs are working correctly. Example, check the sensor signals by turning the motor manually.
Step 2: Setup of PWM and Gatedriver configuration
Set register MOTOR_TYPE_N_POLE_PAIRS according to the motor used in the application, example, a BLDC-motor with four pole pairs. For a DC motor, set the number of pole pairs to one. The PWM can be configured with the corresponding registers PWM_POLARITIES (gate driver polarities), PWM_MAXCNT (PWM frequency), PWM_BBM_H_BBM_L (BBM times), and PWM_SV_CHOP (PWM mode). After setting the register PWM_SV_CHOP to 7, the PWM is on and ready to use. Inspect the PWM outputs after turning on the PWM, when using a new hardware design.
Step 3: Open Loop Mode
In the open loop mode, turn the motor by applying voltage to the motor. This mode is useful for test and setup of ADCs and position sensors. It is activated by setting the corresponding registers for PHI_E_SELECTION, and MODE_MOTION. With UD_EXT, the applied voltage can be regulated upwards until the motor starts to turn. Acceleration and target velocity can be changed by their respective registers.
Step 4: Setup of ADC for current measurement
Configure the ADCs according to the motortype and application. The decimation rate of the Delta Sigma ADCs must match the chosen PWM frequency. When the motor turns in open loop mode, the current measurement can be easily calibrated. Configure the offset and gain of phase current signals by setting the corresponding registers. Configure the current measurements and PWM channels by matching phase voltages and phase currents. Register ADC_I_SELECT can be used to switch relations.
Step 5: Setup of Feedback Systems
In open loop mode also, the feedback systems can be checked for correct operation. Configure and compare the position sensor(s) used in the application against open loop angles. Encoder initialization routines can be used to set angle offsets for relative position encoders according to application needs.
Step 6: Setup of FOC Controllers
The feedback system, position, and velocity signal switches have to be configured accordingly inside the FOC, as well as controller output limits.
Determine the PI controller parameters for the FOC controllers. Depending on the application only, configure the controllers for the used cascade (example, only the current controller in torque mode), starting with the current controller, followed by the velocity controller, followed by the position controller. It is recommended to determine the PI controller parameters with the RTMI, as it supports real-time access to registers and the TMCL IDE offers tools for automated controller tuning. Controller tuning without real-time access might lead to poor performance.
Afterwards, choose the desired motion mode and feed in reference values.
Step 7: Advanced Functions
For performance improvements, apply biquad filters and feed forward control.

13 Package Dimensions

Package: QFN76, 0.4 mm pitch, size 10.5 mm x 6.5 mm.

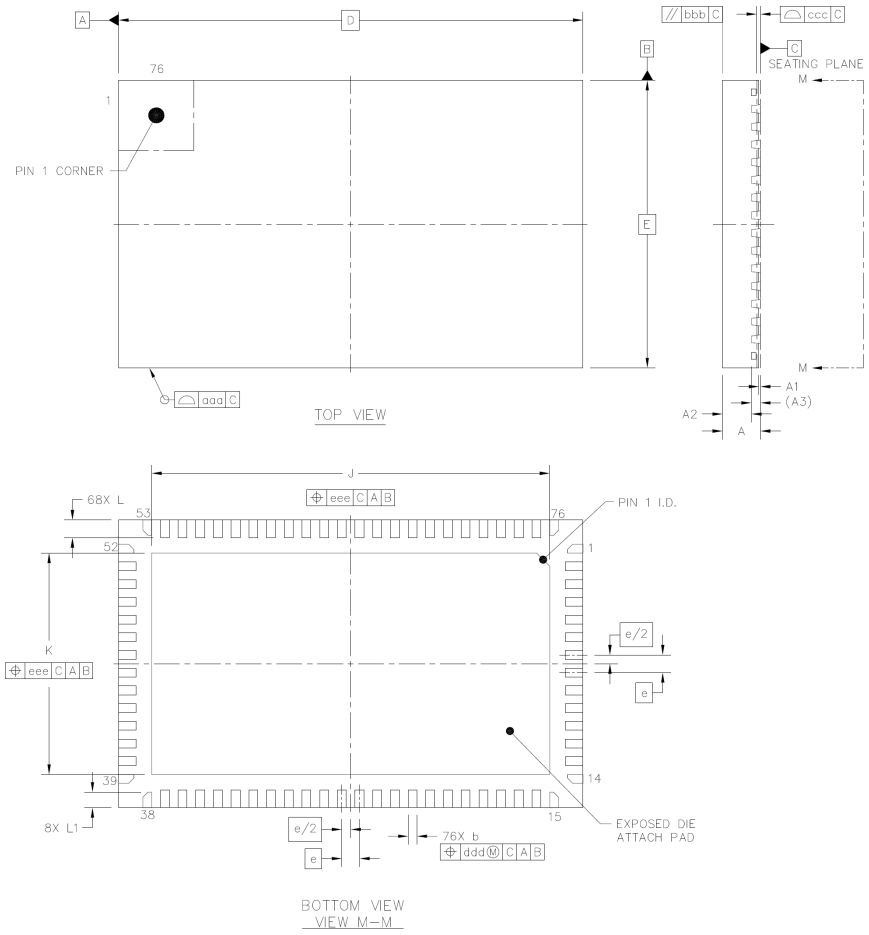


Figure 48: QFN76 Package Outline

QFN76 Package Dimensions in mm				
Description	**Dimension[mm]**	**min.**	**typ.**	**max.**
Total Thickness	A	0.80	0.85	0.90
Stand Off	A1	0.00	0.035	0.05
Mold Thickness	A2	—	0.65	—
L/F Thickness	A3		0.203 REF	
Lead Width	b	0.15	0.2	0.25
Body Width	D		10.5 BSC	
Body Length	E		6.5 BSC	
Lead Pitch	e		0.4 BSC	

QFN76 Package Dimensions in mm				
EP Size	J	8.9	9	9.1
EP Size	K	4.9	5	5.1
Lead Length	L	0.35	0.40	0.45
Lead Length	L1	0.30	0.35	0.40
Package Edge Tolerance	aaa		0.1	
Mold Flatness	bbb		0.1	
Coplanarity	ccc		0.08	
Lead Offset	ddd		0.1	
Exposed Pad Offset	eee		0.1	

Table 44: Package Outline Dimensions

Figure 49 shows the package from top view. Decals for some CAD programs are available on the product's website.

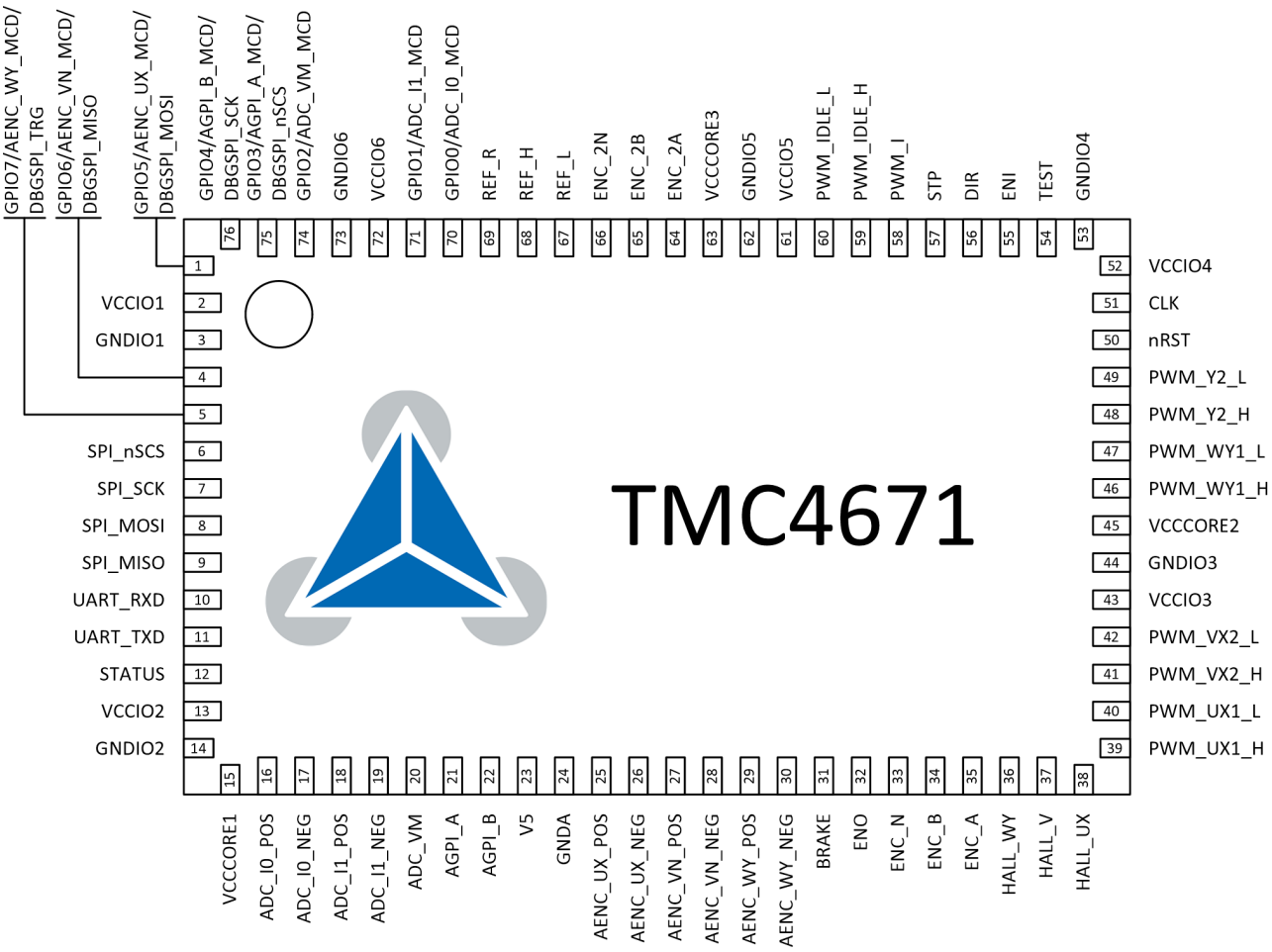


Figure 49: Pinout of TMC4671 (Top View)

14 Supplemental Directives

14.1 Producer Information

14.2 Copyright

14.3 Trademark Designations and Symbols

Trademark designations and symbols used in this documentation indicate that a product or feature is owned and registered as trademark and/or patent either by ADI Trinamic or by other manufacturers, whose products are used or referred to in combination with ADI Trinamic's products and ADI Trinamic's product documentation.

This Data Sheet is a non-commercial publication that seeks to provide concise scientific and technical user information to the target user. Thus, trademark designations and symbols are only entered in the Short Spec of this document that introduces the product at a quick glance. The trademark designation /symbol is also entered when the product or feature name occurs for the first time in the document. All trademarks and brand names used are property of their respective owners.

14.4 Target User

The documentation provided here, is for programmers and engineers only, who are equipped with the necessary skills and have been trained to work with this type of product.

The Target User knows how to responsibly make use of this product without causing harm to himself or others, and without causing damage to systems or devices, in which the user incorporates the product.

14.5 Disclaimer: Life Support Systems

ADI Trinamic/Trinamic Motion Control GmbH & Co. KG does not authorize or warrant any of its products for use in life support systems, without the specific written consent of ADI Trinamic/Trinamic Motion Control GmbH & Co. KG.

Life support systems are equipment intended to support or sustain life, and whose failure to perform, when properly used in accordance with instructions provided, can be reasonably expected to result in personal injury or death.

Information given in this document is believed to be accurate and reliable. However, no responsibility is assumed for the consequences of its use nor for any infringement of patents or other rights of third parties which may result from its use. Specifications are subject to change without notice.

14.6 Disclaimer: Intended Use

The data specified in this user manual is intended solely for the purpose of product description. No representations or warranties, either express or implied, of merchantability, fitness for a particular purpose

or of any other nature are made hereunder with respect to information/specification or the products to which information refers and no guarantee with respect to compliance to the intended use is given.

In particular, this also applies to the stated possible applications or areas of applications of the product. TRINAMIC products are not designed for and must not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death (safety-Critical Applications) without ADI Trinamic's/Trinamic Motion Control GmbH & Co. KG specific written consent.

ADI Trinamic/Trinamic Motion Control GmbH & Co. KG products are not designed nor intended for use in military or aerospace applications or environments or in automotive applications unless specifically designated for such use by ADI Trinamic/Trinamic Motion Control GmbH & Co. KG.
ADI Trinamic/Trinamic Motion Control GmbH & Co. KG conveys no patent, copyright, mask work right or other trade mark right to this product. ADI Trinamic/Trinamic Motion Control GmbH & Co. KG assumes no liability for any patent and/or other trade mark rights of a third party resulting from processing or handling of the product and/or any other use of the product.

14.7 Collateral Documents & Tools

This product documentation is related and/or associated with additional tool kits, firmware and other items, as provided on the product page at: www.analog.com.

15 Errata of TMC4671-LA/-ES2/-ES

15.1 Errata of TMC4671-LA

15.1.1 PID_POSITION_ACTUAL Glitches when Calculated from Hall Sensor Angle and Hall Interpolation is Activated

Hall interpolation is not intended for positioning applications, especially not with changes of direction. In the following register, configuration glitches on PID_POSITION_ACTUAL 0x6B can occur:

Register	Value (Descr.)	Note
0x51 POSITION_SELECTION	5 (PHI_E_HALL selected) 6 (PHI_M_HALL selected) 0 (use selection in register 0x52)	Here PID_POSITION_ACTUAL is calculated from PHI_E_HALL or PHI_M_HALL.
0x52 PHI_E_SELECTION	5 (PHI_E_HALL selected)	
0x33 HALL_MODE	bit 8 = 1 (hall interpolation enabled)	

Table 45: Registersettings Susceptible to Glitches

In this configuration, PID_POSITION_ACTUAL might not be able to count full revolutions correctly as the interpolated PHI_E_HALL might glitch when motor stops. As a result, a full revolution is counted where there not one is completed. If the user does not rely on correct position information in register PID_POSITION_ACTUAL, the silicon error does not affect the application.

Workaround
The hall sensor interpolation must be switched off.

15.1.2 Analog Encoder PPR

Using an analog encoder with a PPR $\neq 2^n$ (n = 0,1,2...31) may lead to a constant offset in AENC_DECODER_PHI_M and AENC_DECODER_PHI_E when the decoder count in AENC_DECODER_COUNT crosses 0.
Workaround
When using PPR $\neq 2^n$ (n = 0,1,2...31), 0x41 AENC_DECODER_COUNT must be between [-2^{31}; -2^{16}] and

[2^{16}; 2^{31}]. Example, 0x41 AENC_DECODER_COUNT is set to 2^{31} at encoder initialization. This results in a positioning range of $2^{((16/PPR)-2)}$.

15.2 Fixes of TMC4671-LA/-ES2 vs. Errata of TMC4671-ES

#	TMC4671-ES Erratum	TMC4671-LA Fix	Description
1	SPI subnode MSB read error	SPI subnode correction	Read using SPI subnode now works correct
2	RTMI critical timing	RTMI enhanced	RTMI works with isolated RTMI-USB IF
3	PI advanced controller	PI scaling updated	Scaling selectable available

4	ADC group clock cross talk	ADC clocks corrected	Crosstalk eliminated
5	PWM_IDLE_L/_H un-used	PWM outputs are at high impedance until ENI is high	Possible to configure PWM signal polarity and afterwards enable PWM signals. Idle state can be set with PD or PU resistor on PWM outputs.
6	Space Vector PWM	SVPWM scaling corrected	SVPWM gives +12% effective voltage. With space vector PWM enabled, voltage scaling is modified.
7	Step direction target position	Processing corrected	Step direction as target position
8	ABN encoder register access	Access corrected	ABN counter over-writeable
9	ENI and ENO	Function updated	ENI and ENO act as enable signals
10	-	Hall sync PWM sample	Optional hall sampling at PWM center
11	-	PWM_POLARITIES register initialized to 0x0	Active high PWM signal polarity is preferred.
12	-	Registers PHI_M_EXT and POSITION_EXT removed	Registers not used.
13	Watchdog not properly working	Watchdog removed	Watchdog intended to monitor CLK. Watchdog flag cannot be reset.

Table 46: TMC4671-ES Errata vs. TMC4671-ES2/-LA Fixes

15.3 Errata of TMC4671-ES Engineering Samples as Reference

1. SPI subnode Interface
   The SPI subnode Interface in the TMC4671-ES might show corrupted MSB of read data.
2. Realtime Monitoring Interface
   The RTMI of TMC4671-ES cannot be used with galvanic isolation due to timing issue.
3. PI Controllers
   The P Factor of the TMC4671-ES in the advanced position controller was not properly scaled and the integrator in the advanced PI controller is not reset when P or I parameters are set to zero.
4. Integrated ADCs
   The Delta Sigma ADCs of TMC4671-ES show signal crosstalk caused by ADC clock crosstalk.
5. Pins PWM_IDLE_H and PWM_IDLE_L without function
   Pins PWM_IDLE_H and PWM_IDLE_L of TMC4671-ES are proposed to set gate driver control polarity.
6. Space vector PWM does not allow higher voltage utilization
   The space vector PWM of the TMC4671-ES does not allow higher voltage utilization.
7. Step direction counter not used as target position
   The step direction counter of the TMC4671-ES correctly counts but is not available as target position.

8. Register write access to ABN encoder count register and N pulse status bits
   Write access to count registers of TMC4671-ES cleared these to zero and encoder N pulse input signals were not available within the status register.

9. ENO and ENI
   Enable input (ENI) and enable output (ENO) of the TMC4671-ES partially worked as intended. (incomplete reset assignment, missing error sum clear on disable).

15.4 Actions to Avoid Trouble

Factors to consider when moving from TMC4671-ES to TMC4671-LA:

- Update P and I parameter for the advanced PI controller in case of switching numerical representation from Q8.8 to Q4.12 (classical PI controller is unchanged).
- Mount pull-up resistors, if required, for gate driver control signals during power-on reset.
- Check setting of SVPWM control bit to avoid unwanted speed-up by SVPWM in torque mode (power-on default is disabled without speed-up).
- Check setting of additional hall_sync_pwm_enable bit for high speed application with usage of hall signals (power-on default is disabled).

15.5 Recommendations

- TMC4671-LA (TMC4671-ES2) is drop-in compatible to the TMC4671-ES. Nevertheless, the TMC4671-LA must be functional qualified as replacement to avoid unwanted behavior caused by corrections of errata of TMC4671-ES.

  For example: The space vector PWM /SVPWM) control bit does not have an effect for the TMC4671-ES in torque mode. The space vector PWM is corrected for the TMC4671-LA. So, if the SVPWM control bit is unwanted enabled for the TMC4671-ES, the TMC4671-LA runs approximately +12% faster in torque mode with the same settings.

16 Figures Index

17 Tables Index

18 Revision History

18.1 IC Revision

Version	Date	Description
V1.0	07/17	Engineering samples TMC4671-ES (1v0 2017-07-03-19:43)
V1.3	04/19	Release version TMC4671-LA (1v3 2019-04-30-12:55)

Table 47: IC Revision

18.2 Document Revision

Version	Date	Description
Rev 0	04/19	Initial release.
Rev 1	02/24	Changes on all pages.

Table 48: Document Revision

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Analog Devices Inc.:

TMC-UPS-10A70V-A-EVAL TMC-UPS-2A24V-A-EVAL TMC4671-LA TMC4671+TMC6100-BOB TMC4671-LEV-REF USB-2-RTMI TMC4671-BOB TMC4671-EVAL TMC4671-2A24V-EV-KIT TMC4671-10A70V-EV-KIT TMC4671-TOSV-REF