Reference Manual

TMC9660 Register Mode

TMC9660 Register Mode Reference Manual

GENERAL DESCRIPTION

The TMC9660 is a highly integrated monolithic gate driver and motor controller IC with buck converter. It includes a smart gate driver, a high-performance motion controller with hardware-based field-oriented control (FOC) and servo controller (velocity, position, ramp generator), motor position feedback interfaces (A/B/N encoder, HALL), an analog signal processing block for bottom shunt current measurement (programmable current-sense amplifiers [CSAs] and analog-to-digital converters [ADCs]).

The TMC9660 supports two modes of operation: one mode to directly access the hardware registers, and the enhanced and simplified parameter mode. This document is the reference manual for the TMC9660 Register Mode and provides the necessary information needed to configure and operate the device. For general information on the IC, refer to the main TMC9660 Data Sheet.

APPLICATIONS

- Robotics
- Power Tools
- Gardening
- Automated Guided Vehicles (AGV)/Warehouse Automation
- Pump (e.g., Peristaltic)
- Industrial 3D Printing
- Factory Automation
- Desktop Manufacturing
- E-Bike/Light Electric Vehicles or LEV

TMC9660 FEATURES

- Three-Phase Permanent Magnet Synchronous Motors (PMSM)/Brushless DC (BLDC), Two-Phase Stepper Motor, and Brushed DC Motor Support
- 7.7V to 70V Single-Supply Operating Voltage Range
- Smart Gate Driver with Adjustable Strength up to 1A/2A Source/Sink
- Field-Oriented Controller/FOC in Hardware for Wide Bandwidth Current Control Loop
- Position, Velocity, and Torque Controller in Hardware for Fast and Precise Control
- 8-Point Ramp Generator with Ramp Calculation in Real Time in Hardware
- Fast Space Vector Pulse Width Modulation (SVPWM) Engine (2kHz …100kHz) with 120MHz Clock
- Feedback Position Sensor Support (Hall, A/B/N)
- Bottom Shunt Current Measurement (Programmable CSA and ADCs)
- Charge Pump with Voltage Doubler
- Trickle Charge Pump for 100% PWM Duty Cycle
- SPI, UART Interfaces for Communication with Main/Application Controller
- Internal Oscillator with Phase Locked Loop (PLL) and Optional External Crystal or Clock Support
- Compact Monolithic Solution, 64-Pin, 9mm x 9mm TQFN Package

One Analog Way, Wilmington, MA 01887-2356, U.S.A. | Tel: 781.329.4700

SIMPLIFIED BLOCK DIAGRAM

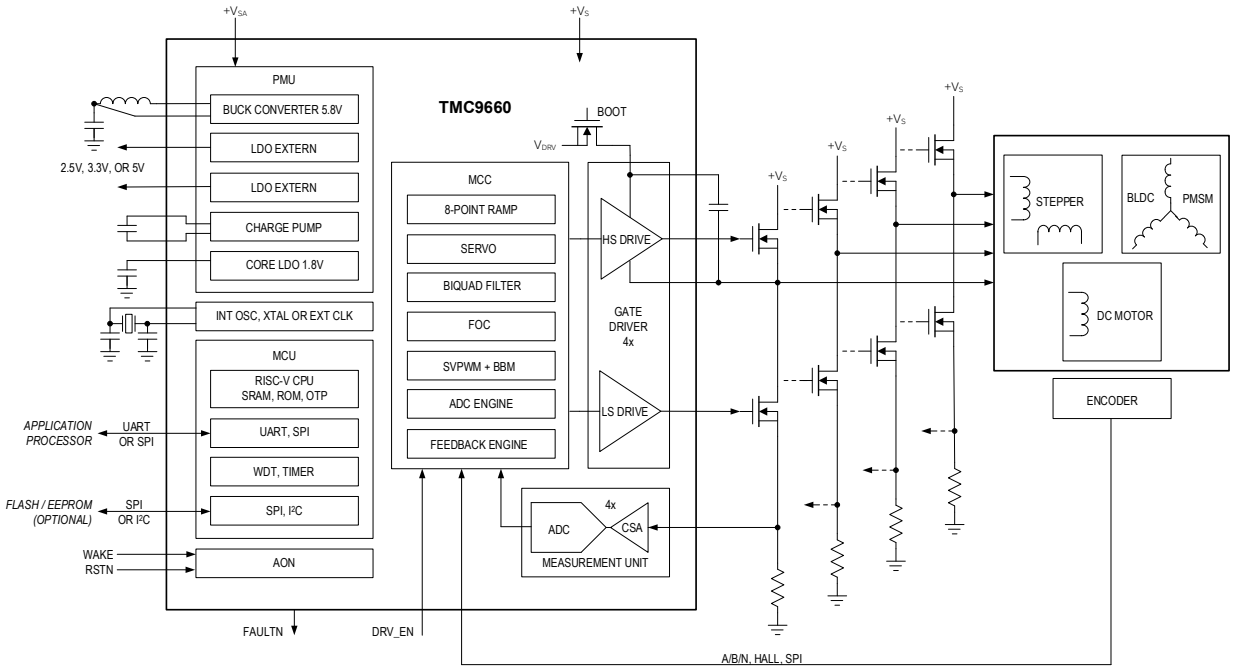

Figure 1. TMC9660 Block diagram

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

HOW TO START

The TMC9660 is an advanced device and should preferably be used on one of the available evaluation platforms first. This provides a tool to easily set up, run, and evaluate the TMC9660 for the targeted application. Also, consider choosing the parameter version of the TMC9660 over the register version, as the parameter version offers more features and an easier setup.

The following sections provide a description of the different features. They are meant to be used in conjunction with the *Register Map* provided at the end of this document. They also contain small step-by-step guides that reference all the required steps to get a feature running.

Consider the following approach:

1. Follow *Boot Configuration* to establish a connection to the Bootloader of the TMC9660. Setup the general configuration parameters mentioned in the electrical data sheet like the voltage regulators and the communication interface. Also, select which pins are used for the Hall and ABN encoder.
2. Boot into the application in register mode and establish a connection as described in .
3. Follow the *Gate driver setup*. If the user is not yet sure about the values to put in, leave the motor unconnected, apply a fixed PWM on the output and conform proper gate switching behavior with an oscilloscope.
4. Follow the *PWM Engine .*
5. Connect a motor and let it run in *Open-Loop Voltage Mode*. This provides the user with encoder and current signals that can be used to confirm the encoder and ADC configuration.
6. Setup the Encoders as described in *Encoder setup.*
7. Setup the ADC as described in *Current sensing and Analog Measurement Setup.*
8. Complete the *FIELD ORIENTED CONTROL* engine setup. The user should now be able to run the motor in torque, velocity and position mode, depending on the user application needs.
9. If required, follow the *Ramp Generator .*

COMMUNICATION INTERFACES

For register read/write access either UART or SPI interface may be used.

UART Interface

The UART interface uses two signals/pins – UART_TX (transmit data out) and UART_RX (receive data in). For bus communication – e.g., RS485 – an additional signal/pin UART_TXEN is available for switching an external transceiver between transmit and receive mode in hardware.

The communication protocol itself follows a strict request/reply order. That is, new commands should not be sent out from the attached microcontroller before the reply for the previous command has been received.

The TMC9660 does not send out any reply before receiving a command first to avoid any collision on the bus interface.

All bytes are sent LSB first.

Every command consists of nine bytes. It starts with one-byte address and sync bit, one-byte command field, a one-byte type field, a one-byte motor/bank field and a four-byte value field.

Table 1. Request format for register read/write access through UART

BYTE	0		1	2	3	4	5	6	7	8
Bit	0	1-7	8-15	16-26	27-31	32-63				64-71
Desc.	Sync bit	Module Address	Command	Register Offset	Register Block	Data (highest byte first)				Checksum

Table 2. Request format for RAMDebug access through UART

BYTE	0		1	2	3	4	5	6	7	8
Bit	0	1-7	8-15	16-26	27-31	32-63				64-71
Desc.	Sync bit	Module Address	Command	Subcommand	Index	Data (highest byte first)				Checksum

Command is 146 for writing a register, 148 for reading a register and 142 for accessing RAMDebug.

The sync bit is always 1 – this enables automatic baud rate detection.

The module address reuses the upper 7 bits of the bootloader device address.

The checksum is calculated by adding up the first 8 bytes using 8-bit addition. Here is an example for checksum calculation using C code:

```
unsigned char nCommand [9];

// nCommand[0..7] contain the first 8 request bytes

// calculate checksum from command bytes
uint8_t nChecksum = 0;
for (uint8_t i =0; i < 8; i++)
   nChecksum += nCommand[i];
nCommand[8] = nChecksum; // insert checksum as last byte of the command
```

Figure 2. Checksum calculation for UART communication

For each command, a reply is sent back from the TMC9660. The reply also comprises 9 bytes.

Table 3. Reply format for register read/write access through UART

BYTE	0	1		2	3	4	5	6	7	8
Bit	0-7	8	9-15	16-23	24-31	32-63				64-71
Desc.	Host Address	Sync bit	Module Address	Reply Status	Command	Data (highest byte first)				Checksum

The sync bit is still present in the reply datagram and is still 1.

The checksum calculation is the same as for the request format.

SPI Interface

The SPI interface for communication with an external microcontroller uses the SPI peripheral device of the TMC9660 and requires 4 pins for communication. Its default SPI-Mode is Mode 3.

The external microcontroller operates as SPI controller. Every SPI command from the external microcontroller to the TMC9660 is expected to have a length of 48-bit. A reply for this command has the same length and is sent from the TMC9660 back to the external microcontroller with the next SPI command. All data is sent with most significant bit (MSB) first. The data is sent in big-endian.

Table 4. Request format for register access through SPI

BYTE	0		1	2	3	4	5
Bit	47..42	41	40..32	31..0			
Desc.	Register block	Write (1) or Read (0)	Register offset in block	Writes: Data to be written (big-endian: highest byte first) Reads: Don't care			

All available registers are grouped into blocks. Within each block, individual registers are selected with the register offset.

Table 5. Register Block Number

REGISTER BLOCK	PERIPHERAL
0	MCC – Motion Control Core
1	ADC – ADC Configuration
2	SYS_CTRL – System Control
31	RAMDebug

For all blocks except RAMDebug, the register offset within a register block is the register number listed in the register map.

For RAMDebug, the register offset is composed of the RAMDebug subcommand in the upper 4 bits, and the RAMDebug index in the lower 6 bits. For example, a RAMDebug Get info request (subcommand 10, index 0) would have a register offset value of 640 (binary 1010000000) with the subcommand occupying SPI command bits 40-37, and the index occupying SPI command bits 36-32. For more details on RAMDebug, see the *RAMDebug* section.

Table 6. Reply format for Register read/write access through SPI

BYTE	0			1	2	3	4	5
Bit	47	46..44	43..33	32	31..0			
Desc.	Boot bit	Datagram counter	Reserved	Request status. 1: Successful 0: Failed	Register read data. (big-endian: highest byte first)			

The boot bit is set to 1 on the very first datagram, and 0 on all subsequent datagrams.

The datagram counter is incremented by 1 for each reply datagram, wrapping around back to 0 from a value of 7.

Example Communication

This example communication shows two send and receive data packages from a controller to the TMC9660 in MSB-first.

First transmission

Send a "READ MCC->INFO_CHIP" request, receive a "BOOT status".

Sent by the controller:
0x00, 0x00, 0x00, 0x00, 0x00, 0x00

The first reply from the TMC9660 is:
0x80, 0x00, 0x00, 0x00, 0x00, 0x00

Second transmission

Send another "READ MCC->INFO_CHIP" request, receive the reply to the first datagram with a datagram counter of 1.

Sent by the controller:
0x00, 0x00, 0x00, 0x00, 0x00, 0x00

The next reply from the TMC9660 after the initial boot reply (the register data might defer between product versions):
0x10, 0x01, 0x54, 0x4D, 0x00, 0x01

BOOT CONFIGURATION

This section lists configuration settings for setting up the TMC9660 for register mode operation. For details on how the configuration mechanism is used, refer to the *Bootloader Configuration* section in the TMC9660 Data Sheet and AN-2601. The tables in the following sections list the options that can be configured for the register mode.

ABN Encoder 1

Table 7. Boot configuration options for ABN 1 encoder

NAME	OFFSET	BITS	DESCRIPTION
ABN1_ENABLE	32	1	Enables the usage of ABN1. When enabled, the following other ABN1 settings take effect, otherwise they are ignored. Default: 0
ABN1_A	32	10-11	Selects which pin to use for A input: **0: GPIO5** 1: GPIO8 2: GPIO17 3: RESERVED
ABN1_B	32	12-13	Selects which pin to use for B input: **0: GPIO1** 1: GPIO13 2: GPIO18 3: RESERVED
ABN1_N	32	14-15	Selects which pin to use for N input: **0: N channel disabled** 1: GPIO14 2: GPIO16 3: RESERVED

Hall

Table 8. Boot configuration options for Hall encoder

NAME	OFFSET	BITS	DESCRIPTION
HALL_ENABLE	32	0	Enables the usage of the Hall encoder. When enabled, the following other Hall settings take effect, otherwise they are ignored. Default: 0
HALL_U	32	4-5	Selects which pin to use for U input: **0: GPIO2** 1: GPIO7 2: GPIO9 3: RESERVED
HALL_V	32	6-7	Selects which pin to use for V input: **0: GPIO3** 1: GPIO15 2: RESERVED 3: RESERVED

NAME	OFFSET	BITS	DESCRIPTION
HALL_W	32	8-9	Selects which pin to use for W input: **0: GPIO4** 1: GPIO8 2: GPIO10 3: RESERVED

ENCODER SETUP

The TMC9660 supports two different motor feedback systems, Hall and ABN. Before using them, make sure the bootloader is configured correctly to use the right pins.

Hall Feedback

The Hall decoder within the product is designed to interpret the signals from digital Hall effect sensors, which are magnetic field sensors that produce digital outputs corresponding to the rotor's position.

The hall decoder configuration is done through the MCC_HALL_MODE register. This register allows for setting the polarity of the hall signals, enabling or disabling extrapolation for the electrical angle, and defining the ordering of the hall signals. The maximum change in the electrical angle for extrapolation can be set using the MCC_HALL_DPHI_MAX register. An offset for the electrical angle can be set using the MCC_HALL_PHI_E_OFFSET register. The count of passed hall states is stored in the MCC_HALL_COUNT register. The electrical angle, which can be either raw or extrapolated, is stored in the MCC_HALL_PHI_E_EXTRAPOLATED_PHI_E register.

The exact position of the hall sensor at different angles can be stored in the MCC_HALL_POSITION_060_POSITION_000, MCC_HALL_POSITION_180_POSITION_120, and MCC_HALL_POSITION_300_POSITION_240 registers.

Step-by-Step Setup: Hall Feedback

The following provides the basic register configuration to use Hall sensor feedback for rotor position.

- MCC_HALL_MODE: Set the order of the Hall signals, their polarity and extrapolation.
- MCC_HALL_PHI_E_OFFSET: Apply an offset between the hall state and PHI_E if necessary. This can usually stay at 0. If the user needs to apply large offsets, check that the HALL signals are assigned correctly.
- MCC_HALL_PHI_E_EXTRAPOLATED_PHI_E: Check that the PHI_E is correct. For example, using the guide that follows.
- MCC_PHI_SELECTION: Select the hall as the feedback system.

Verify the Hall Angle:

- Follow *Step-by-Step Setup: Open-Loop Voltage Mode,* setting only UD and leaving UQ at 0 and leading to a slowly turning motor.
- Compare MCC_HALL_PHI_E_EXTRAPOLATED_PHI_E -> PHI_E and MCC_PHI_E -> PHI_E
  The absolute difference between the two should be lower than one sixth of the maximum PHI_E Value (10922)
- If the user encounters a larger difference, check the HALL pin assignments within the Bootloader and also the MCC_HALL_MODE settings.

ABN Encoder

The ABN interface on the TMC9660 is designed to decode the signals from an incremental encoder. This data is then used by the motor control algorithms to accurately determine the rotor's position and facilitate precise control. Several registers within the MCC are dedicated to configuring and managing the ABN interface.

The register MCC_ABN_MODE controls various aspects of the ABN decoder's behavior. This includes the polarity of the A, B, and N (index) signals, the type of N signal detection (single or combined), and whether to clear the count on N signal detection. It also allows for enabling or disabling a digital filter on the N signal and selecting the count.

The encoder's resolution in counts per revolution (CPR) is split into two values, the CPR and its inverse. This allows for less resource intensive calculation of the resulting motor angle but requires both values to be updated together. The CPR value is stored in MCC_ABN_CPR, while its inverse (2^32 / CPR) is stored in MCC_ABN_CPR_INV.

The raw and latched encoder counts are stored in the registers MCC_ABN_COUNT and MCC_ABN_COUNT_N, respectively. The raw count represents the current count value from the encoder, while the latched count is the value captured at the N pulse. An offset can be applied to the calculated electrical angle (PHI_E) using the register MCC_ABN_PHI_E_OFFSET. This provides flexibility in aligning the encoder's position data with the motor's electrical cycle.

Step-by-Step Setup: ABN Encoder

The following is the basic register configuration to use an ABN encoder for rotor position detection.

MCC_ABN_MODE: Set the polarity of the signals and the direction.

MCC_ABN_CPR: Set the CPR according to the used encoder.

MCC_ABN_CPR_INV: Calculate and set CPR_INV.

Follow *Step-by-Step Setup: Open-Loop Voltage Mode,* setting only UD and leaving UQ at 0.

MCC_PHI_EXT: Set MCC_PHI_E_EXT to 0.

MCC_PHI_E_SELECTION: Select phi_e_ext, this stops the motor and holds it in one position.

MCC_ABN_COUNT: Set ABN_COUNT to 0.

MCC_MOTION_CONFIG: Set MCC_MOTION_MODE to 0 to stop any motor movement.

MCC_PHI_SELECTION: Select the ABN as the feedback system.

Verify the ABN Angle:

Follow *Step-by-Step Setup: Open-Loop Voltage Mode,* setting only UD and leaving UQ at 0. MCC_RAMPER_V_TARGET should be adjusted to let the motor turn slowly.

Compare MCC_ABN_PHI_E_ PHI_M -> PHI_E and MCC_PHI_E -> PHI_E

They should follow each other closely. If the user encounters a larger difference, check the ABN pin assignments within the Bootloader and also the polarity and direction. If one PHI_E is incrementing faster or slower than the other, check the CPR settings.

CURRENT SENSING AND ANALOG MEASUREMENT SETUP

The TMC9660 provides four different internal ADCs. Each ADC is assigned to one of the four phases, ADCs are counted from 0 to 3.

Each ADC input is connected to three different sources of the particular phase. The values of these input sources are measured within each PWM cycle and are signed 16-bit values:

1. Current-sense amplifier outputs
    a. Measured raw values are available in the register fields:
        i. I0 and I1 in MCC_ADC_I1_I0_RAW register
        ii. I2 and I3 in MCC_ADC_I3_I2_RAW register
2. External analog input signals AIN0…AIN3 at the GPIO2…5 pins
    a. Measured raw values are available in the register fields:
        i. AIN0 and AIN1 in MCC_ADC_AIN1_AIN0_RAW register
        ii. AIN2 and AIN3 in MCC_ADC_AIN3_AIN2_RAW register
3. Divided phase voltage outputs
    a. Measured raw values are available in the register fields:
        i. U0 and U1 in MCC_ADC_U1_U0_RAW register
        ii. U2 and U3 in MCC_ADC_U3_U2_RAW register

Further on, an additional internal value is processed for ADC0 (temperature TEMP) and ADC2 (supply voltage VM). TEMP and VM are displayed in the MCC_ADC_TEMP_VM register. These fourth sources are alternately measured with the autozero (AZ) values of the particular current-sense amplifiers (CSA). The AZ values are utilized for automatic offset correction of the CS amplifiers.

Different settings can be selected, but only a few are relevant for the application setup itself. These are described in the Current Assignment section that follows basic CSA configuration and current value calculations sections.

Basic Current-Sense Amplifier Configuration

Basically, only the following configuration register must be adapted according to the application setup. Four different CS amplifiers are available – one for each phase. These internal CS amplifiers are counted from 0 to 3.

Following explained register fields are part of the register ADC_CSA_SETUP:

1. Enabling all required CS amplifiers
    a. Register fields CSA0_EN, CSA1_EN, CSA2_EN, and CSA3_EN activate the particular CS amplifier.
2. Current sense gain settings
    a. CSA012_GAIN sets the gain for the CS amplifiers 0, 1, and 2 and CSA3_GAIN for the fourth amplifier.
    b. Following gain values are available:
        i. _GAIN = 0 → CSA gain = 5x
        ii. _GAIN = 1 → CSA gain = 10x
        iii. _GAIN = 2 → CSA gain = 20x
        iv. _GAIN = 3 → CSA gain = 40x
    c. In case the value at the input of the internal amplifier must be directly forwarded to the internal ADC, the bypass register field must be activated – CSA012_BYPASS, resp. CSA3_BYPASS
3. Bandwidth filter settings
    a. The filter at for the CS amplifier can be adjusted as well in CSA012_FILT and CSA3_FILT.
    b. Following settings are available:

i. _FILT = 0 → CSA bandwidth = 1.82MHz
ii. _FILT = 1 → CSA bandwidth = 1.33MHz
iii. _FILT = 2 → CSA bandwidth = 1.00MHz
iv. _FILT = 3 → CSA bandwidth = 740.74kHz

c. To obtain best possible results for the particular filter setting, the following ADC sample time configuration must be applied. This register field ADC_SHIFT_SAMPLE is part of the ADC_SETUP register.
i. _FILT = 0 → ADC_SHIFT_SAMPLE = 0 (= 500ns)
ii. _FILT = 1 → ADC_SHIFT_SAMPLE = 1 (= 600ns)
iii. _FILT = 2 → ADC_SHIFT_SAMPLE = 2 or 3 (= 700ns / 800ns)
iv. _FILT = 3 → ADC_SHIFT_SAMPLE = 4 (= 900ns)

Measured Current Values

ADC values of the raw currents in [mV] can be calculated back for I0…I3 (registers MCC_ADC_I1_I0_RAW resp. MCC_ADC_I3_I2_RAW) and AIN0…AIN3 (registers MCC_ADC_AIN1_AIN0_RAW resp. MCC_ADC_AIN3_AIN2_RAW) by using the following formula:

$$U_{CSAx}[mV] = Ix \times \frac{625}{16382}, with\ x = 0 \dots 3$$

Example: I1 = 12000 → U_{CSA1} = 457.82mV

$$U_{AINx}[mV] = AINx \times \frac{625}{16382}, with\ x = 0 \dots 3$$

Example: AIN2 = 3000 → U_{AIN2} = 114.45mV

Taking shunt resistor and CSA gain into account the shunt current can be obtained by following formula:

$$I_{CSx}[mA] = Ix \times \frac{625}{16382} \times \frac{1}{R_{shunt}[\Omega]} \times \frac{1}{gain}, with\ x = 0 \dots 3$$

Example: CSA gain = 10x, R_{shunt} = 8mΩ, I1 = 12000 → I_{CS1} = 5.72A

During a current acquisition sequence autozero values of the CS amplifiers are also measured. Per default, these autozero values are not filtered over consecutive PWM cycles. To filter these autozero values, CSA_AZ_FLTLNGTH_EXP in the CSA_SETUP registers has to be set to a value above 0. These filter values are then calculated by the following equation:

$$newFilterValue = oldFilterValue + \frac{newValue - oldFilterValue}{2^{CSA\_AZ\_FLTLNGTH\_EXP}}$$

Measured Phase Voltage Values

Raw voltage values of each phase in [mV] can be calculated back for U0…U3 (registers MCC_ADC_U1_U0_RAW resp. MCC_ADC_U3_U2_RAW) by using the following formula:

$$U_x[mV] = Ux \times PHASE\_DIV\_GAIN \times \frac{625}{16382}, with\ x = 0 \dots 3$$

PHASE_DIV_GAIN register field is part of MCC_GDRV_HW register and is adjustable for four different divider values. This value must be adapted according to the maximum expected supply voltage:

- PHASE_DIV_GAIN = 0 → divider value = 80

- PHASE_DIV_GAIN = 1 → divider value = 40
- PHASE_DIV_GAIN = 2 → divider value = 20
- PHASE_DIV_GAIN = 3 → divider value = 10

Phase dividers must be activated by using register fields PHASE_DIV_EN_UVW for phase U, V, and W and PHASE_DIV_EN_Y2 for the fourth phase separately (MCC_GDRV_HW register).

Example: U2 = 11000, PHASE_DIV_GAIN = 2 → U_W = 8.39V

Measured Temperature and Supply Voltage Values

Raw voltage value of the supply voltage in [V] can be calculated back for VM (register MCC_ADC_TEMP_VM_RAW) by using the following formula:

$$U_{VS}[V] = VM \times \frac{40}{16382}$$

Example: VM = 9900 → U_{VS} = 24.17V

Raw voltage value of the temperature [°C] can be calculated back for TEMP (register MCC_ADC_TEMP_VM_RAW) by using the following formula:

$$T[°C] = \frac{TEMP \times \frac{625}{16382} - 620.325}{2.363}$$

Example: TEMP = 22000 → T = 92.68 °C

Measurement Trigger Point Adaptations

The ADC measurements start with every PWM cycle. As a center aligned PWM scheme is used, all PWM signals are 0 at this point.

Typically, the default ADC sample time of 500ns is enough for precise measurements of all values (current sense, analog inputs, phase voltage, temperature, supply voltage). If a longer sample time must be used, register field ADC_SHIFT_SAMPLE of ADC_SETUP register can be adjusted from 0…15 for sample times between 500….2000ns (additional 100ns per each count). For example, for longer filter CSA settings sample time should be prolonged, see the *Basic Current-Sense Amplifier Configuration* section. Be aware that higher sample periods than 1us could lead to an incorrect controller sequence for the highest PWM frequency of 100kHz and all enabled multiplexer measurements due to too long ADC measurement cycles.

Another feature to manipulate the ADC timing is the shift of the first measurement by adapting register field TRIGGER_POS (register MCC_ADC_I_GEN_CONFIG): Related to the PWM cycle the trigger position is delayed by

$$triggerDelay = \frac{(TRIGGER\_POS)}{65536} \times Period_{PWM}$$

Be aware that with this timing shift it is possible that the closed-loop control scheme is disrupted. Too high values here should be avoided.

As stated, one measurement cycle comprises per default following measurements in the sequence as follows:

1. Current sense
2. External analog input voltage
3. Phase voltage
4. Temperature/Supply voltage (alternately to autozero measurements)

Register ADC_SRC_CONFIG provides register fields for each ADC0…3 to alter sequence and de-/activate measurements (x represents in the following the ADC count no. 0…3):

- ADCx_MUX0_CFG: Sequence no. of the current sense acquisition (default =1 for 1st measurement)
- ADCx_MUX1_CFG: Sequence no. of the external analog value acquisition (default =2 for 2nd measurement)
- ADCx_MUX2_CFG: Sequence no. of the phase voltage value acquisition (default =3 for 3rd measurement)

If any of these values are set to 0, the particular measurement is skipped.

For example, setting ADC2_MUX0_CFG=2, ADC2_MUX1_CFG=0 and ADC2_MUX2_CFG=1 result in the following measurement sequence at ADC2:

1. Phase voltage
2. Current sense
   (External analog input voltage acquisition is skipped)

Supply voltage resp. temperature voltage measurements can be omitted by setting ADC0_MUX3_DIS = 1 resp. ADC2_MUX3_DIS = 1. It is still recommended to keep these measurements for temperature and supply voltage active.

Finally, setting register field ADCx_MUX2_DETOUR to 1 skips phase voltage measurements. Instead, external analog value acquisition is executed twice per PWM cycle (ADCx_MUX2_CFG must be still active and contain a valid sequence number!).

ADC_STATUS Bits

ADC_STATUS contains several status flags related to the ADC operations. If the current acquisition sequence is not valid, ADCx_MUXSEQ_FAIL is activated (see the section which mentions x = 0…3).

Further on, RDY_ADCx bits indicated if all four ADCs are operational (RDY_ADCx = 1) or if any is still in configuration or off (RDY_ADCx = 0).

Finally, activated ADCx_WTCHDG_FAIL bits (=1) indicate that the particular ADC is not responding correctly. Power cycle the chip if this happens during regular operation.

Current Assignment

To feature a setup with the TMC9660 that provides many degrees of freedom for the application, several current value sources and different measurement methods can be assigned for the relevant currents of the Motion Control Core (MCC) unit. The final current values that are required for motion control are as follows:

- BLDC motors (UVW):
  - Register field IUX of register MCC_ADC_IWY_IUX
  - Register field IV of register MCC_ADC_IV
  - Register field IWY of register MCC_ADC_IWY_IUX
- Stepper motors (XY):
  - Register field IUX of register MCC_ADC_IWY_IUX
  - Register field IWY of register MCC_ADC_IWY_IUX

Which current values are assigned is selected by the _SELECT register fields of register MCC_ADC_I_GEN_CONFIG. Four different current value register fields define the basic current values: I0, I1, I2, and I3 of registers MCC_ADC_I1_I0_SCALED and MCC_ADC_I3_I2_SCALED. How these values are obtained is explained in the following sections.

The following register fields define correct corresponding current source: UX1_SELECT, VX2_SELECT, WY1_SELECT, Y2_SELECT. By defining a number between 0 and 3 for these select register fields, the corresponding current I0 to I3 is assigned.

Finally, the MEASUREMENT_MODE register field (register MCC_ADC_I_GEN_CONFIG) defines how and from which source these current values are obtained:

- MEASUREMENT_MODE = 0 (not recommended)
  - Values are directly transferred from IUX, IV, IWY (BLDC) resp. IUX, IWY (Stepper)
- MEASUREMENT_MODE = 1 (BLDC only, not recommended)
  - Values are directly transferred from IV and IWY, IU is calculated accordingly
- MEASUREMENT_MODE = 2 (BLDC only, not recommended)
  - Values are directly transferred from IUX and IWY, IV is calculated accordingly
- MEASUREMENT_MODE = 3 (BLDC only, not recommended)
  - Values are directly transferred from IUX and IV, IW is calculated accordingly
- MEASUREMENT_MODE = 4 (preferred mode)
  - Values are switched, calculated, and assigned automatically according to the actual motor phase and PWM_SWITCH_LIMIT
  - For stepper motors, the correct values for X and Y are automatically taken from UX1/VX2 resp. WY1/Y2

I0, I1, I2, and I3 of registers MCC_ADC_I1_I0_SCALED and MCC_ADC_I3_I2_SCALED are calculated individually based on the raw current values from the internal current-sense amplifiers:

- I0 and I1 from MCC_ADC_I1_I0_RAW register
- I2 and I3 from MCC_ADC_I3_I2_RAW register

These current values can be scaled, and an offset can be compensated. These scale and offset values can be defined individually for each current I0…I3 in several register fields:

$$Ix\_SCALED = (Ix + OFFSET) \times \frac{SCALE}{1024}$$

(x represents the particular phase =0…3).

SCALE and OFFSET register fields are defined for each phase in the corresponding MCC_ADC_I0_CONFIG, MCC_ADC_I1_CONFIG, MCC_ADC_I2_CONFIG, and MCC_ADC_I3_CONFIG registers.

These I0_SCALED, I1_SCALED, I2_SCALED, and I3_SCALED values are available in the MCC_ADC_I1_I0_SCALED and MCC_ADC_I3_I2_SCALED registers that are mentioned and represent the base values for the selection.

Gate driver setup

The gate driver is responsible for providing the necessary signals to the gates of the power transistors, ensuring efficient and reliable motor control. It supports various configuration options and protection mechanisms to adapt to different applications.

Currents and Timings

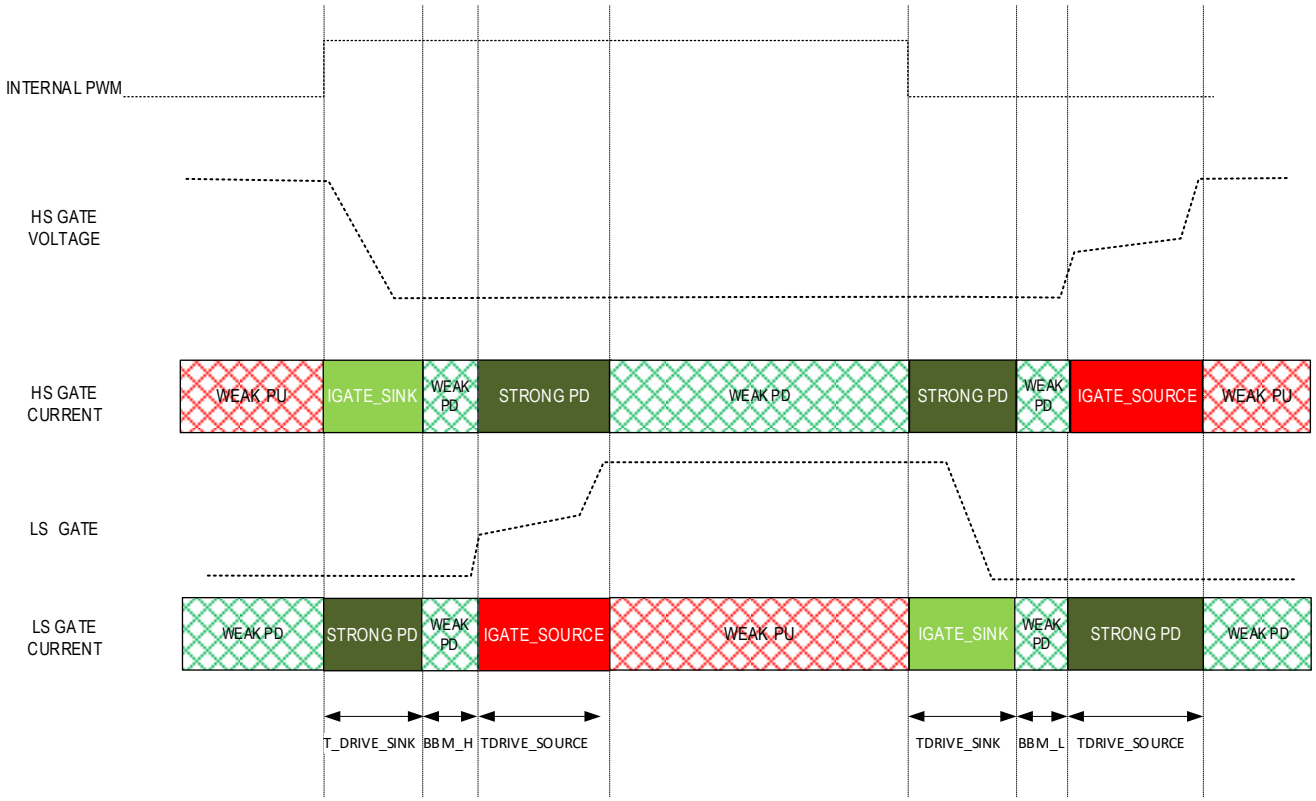

Figure 3. Gate driver timings and currents

The T_DRIVE parameters, configured within the MCC_GDRV_TIMING register, determine the charge and discharge duration of the MOSFETs. During this time the user selectable currents within the MCC_GDRV_CFG register are applied, while the I_STRONG current drives the complementary MOSFET. The T_DRIVE must be set sufficiently long to allow a proper switching of the MOSFET under all conditions. However, setting it higher than necessary may introduce a long deadtime and degrade control performance.

The adaptive mode found in MCC_GDRV_CFG is a feature designed to optimize MOSFET gate driver performance by dynamically adjusting the MOSFET's discharge time. The gate driver continuously monitors the gate voltage during the discharge cycle. If the gate is fully discharged prematurely, the discharge cycle is shortened by terminating the gate drive current before the full T_DRIVE_SINK time elapses. In this operation mode, the T_DRIVE_SINK parameters act as an upper limit for the discharge time. The adaptive mode has no effect on the T_DRIVE_SOURCE.

The Break Before Make (BBM) time, also referred to as dead time defines the interval between the deactivation of one MOSFET and the activation of the complementary MOSFET. This delay is essential to prevent a short circuit, often termed "shoot-through," which can lead to significant power dissipation and potentially damage the device. The BBM time is configurable through the MCC_GDRV_BBM register. During the BBM, the HOLD current is applied. It is generally recommended to set the BBM values to zero and rely on the T_DRIVE parameter in the MCC_GDRV_TIMING register as it is usually sufficient to prevent any cross conduction.

Enabling the Gate Driver

Before utilizing any gate driver functionality, it's crucial to enable the internal analog bias voltage by setting the BIAS_EN bit in the MCC_GDRV_HW register to 1. This step powers up the necessary circuitry for subsequent gate driver operations.

Next enable the internal charge pump by setting CHARGEPUMP_EN. It allows for independent charging of the bootstrap capacitors, enabling 100% duty cycle operation.

The gate driver uses bootstrap capacitors to provide the necessary voltage for driving the high-side transistors. The BST_ILIM_MAX field in the MCC_GDRV_HW register allows the user to define the maximum charge current for these capacitors. Additionally, the BST_SW_CP_EN bit enables an internal charge pump for efficient charging of the bootstrap capacitors and is recommended to be turned on.

The gate driver controls four half-bridges, each responsible for driving a phase of the motor. To enable the PWM signals for each half-bridge, the user needs to set the corresponding BRIDGE_ENABLE bits in the MCC_GDRV_HW register:

- BRIDGE_ENABLE_U: Enables the PWM bridge for the UX1 phase.
- BRIDGE_ENABLE_V: Enables the PWM bridge for the VX2 phase.
- BRIDGE_ENABLE_W: Enables the PWM bridge for the WY1 phase.
- BRIDGE_ENABLE_Y2: Enables the PWM bridge for the Y2 phase.

Protection Features

The gate driver incorporates several protection features to ensure the safe and reliable operation of the motor drive system. These features are designed to detect and respond to fault conditions, preventing damage to the power stage.

Overcurrent and Short-Circuit Protection

Overcurrent protection (OCP) is a critical safety feature that monitors the current flowing through the power transistors. If the current exceeds a predefined threshold, the OCP mechanism triggers, typically by disabling the gate driver. This helps to prevent excessive current from damaging the transistors.

The high-side OCP is always measured using the voltage drop over the MOSFET, while the low-side OCP can also use the shunt. This and the OCP thresholds for the low-side and high-side transistors can be configured through the MCC_GDRV_OCP_UVW and MCC_GDRV_OCP_Y2 registers. These registers also allow the user to set the deglitch time, blanking time, and threshold level for each channel. The deglitch time determines the duration for which the overcurrent condition must persist before it is recognized as a fault. The configurable blanking prevents the OCP detection after charge and discharge cycle to filter out any transient spikes or noise. The current threshold level is programmable and represents the current at which the OCP mechanism is triggered. The OCP mechanism includes an automatic retry feature, specified in the MCC_GDRV_PROT register. After a specified number of retries, if the overcurrent condition persists, the gate driver remain disabled until the fault is cleared.

Undervoltage Events

Undervoltage lockout (UVLO) protects the gate driver and power transistors from operating at insufficient voltage levels. Three different types of UVLO are implemented.

The VS UVLO detects an insufficient motor voltage. To set it up, set the VS_UVLO_CMP_EN bits in the MCC_GDRV_HW register to 1. The VS_UVLO_LVL field in MCC_GDRV_CFG sets the threshold. If the fault occurs and the protection is enabled, the gate driver is disabled.

The VDRV UVLO detects an insufficient gate driver voltage. To set it up, set the VDRV_UVLO_CMP_EN bits in the MCC_GDRV_HW register to 1. The threshold for the VDRV UVLO is fixed at 6.45V. If the fault occurs and the protection is enabled, the gate driver is disabled.

The BST UVLOs detect an insufficient voltage on any of the connected bootstrap capacitors. If the fault occurs and the protection is enabled, the respective channel is disabled. It is the user's responsibility to disable the other channels if desired.

Gate Short Protection (VGS)

The gate driver also includes protection against gate-to-source voltage shorts. This is achieved by monitoring the voltage between the gate and source terminals of the power transistors. If a short circuit is detected, the gate driver is disabled to prevent damage. The VGS protection parameters, such as deglitch and blanking times, can be configured through the MCC_GDRV_PROT register. The deglitch time (VGS_DEGLITCH_UVW and VGS_DEGLITCH_Y2) determines the minimum duration of a short-circuit condition before it is recognized as a fault. The blanking time (VGS_BLANKING_UVW and VGS_BLANKING_Y2) prevents the VGS fault detection after a charge and discharge cycle to filter out any transient spikes or noise. The gate driver can be configured to automatically retry enabling the channel after a VGS fault. The number of retries is specified in the LS_RETRIES_UVW, HS_RETRIES_UVW, LS_RETRIES_Y2, and HS_RETRIES_Y2 fields of the MCC_GDRV_PROT register. If the fault persists after the retries, the gate driver remains disabled until the fault is cleared.

Fault Handling, Enabling, and Reporting

Each fault can be enabled individually in the MCC_GDRV_STATUS_EN register. If the corresponding bit of the fault is not set in this register, the fault is neither reported in the MCC_STATUS register nor does it trigger any protection mechanism. The protection features can be individually enabled or disabled through the MCC_GDRV_PROT_EN register. Each protection feature has a corresponding enable bit in this register. For example, to enable overcurrent protection for the low-side MOSFET of the UX1 channel, the user would set the LS_SHORT_EN_U bit to 1. The gate driver provides detailed fault information through the MCC_GDRV_STATUS and MCC_GDRV_FAULT registers. These registers indicate the specific type of fault that occurred (e.g., overcurrent, undervoltage, short circuit) and the affected channel. For instance, the LS_SHORT_U bit in MCC_GDRV_STATUS would be set if a low-side short-circuit fault is detected on the UX1 channel. To clear a fault, write a '1' to the corresponding bit in the MCC_GDRV_STATUS register. Afterwards the operation of the gate driver can be resumed by setting the corresponding fault fields in the MCC_GDRV_FAULT register. Note that both HS and LS must be writing together to release a half bridge.

Step-by-Step Setup: Gate Driver

Example step-by-step guide for common gate driver operations. "_X" is used as a placeholder for the different phases.

Startup:

MCC_PWM_CONFIG: Set ENABLE_X to 1, Set CHOP to 1.

MCC_GDRV_CFG: Set SINK_X and SOURCE_X to the desired drive strength, optionally enable ADAPTIVE_X.

MCC_GDRV_TIMING: Adjust T_DRIVE_X according to the connected FET and drive strength.

MCC_GDRV_BBM: Adjust BBM_X, setting it to 0 is recommended.

MCC_GDRV_STATUS_EN: Set to 0x00000000 (disables any faults during startup).

MCC_GDRV_STATUS: Set to 0xFFFFFFFF (clears existing faults).

MCC_GDRV_HW: Set to 0x03300FFF (enables the gate driver and starts charging the bootstrap capacitors).

MCC_PWM_CONFIG: Set CHOP to 7 (forwards PWM signals to gate driver).

Outputting a fixed duty cycle:

Follow gate driver startup.

MCC_PWM_MAXCNT: Set the desired PWM Frequency.

MCC_PWM_VX2_UX1_EXT: Set the desired duty cycles for phase UX1 and VX2.

MCC_PWM_Y2_WY1_EXT: Set the desired duty cycles for phase WY1 and Y2.

MCC_PWM_CONFIG: Set EXT_ENABLE_X to 1.

Protection Setup:

Follow gate driver startup.

MCC_GDRV_PROT: Set blanking and deglitch for the gate short protection and set TERM_PWM_ON_SHORT if all phases should be disabled on a fault, otherwise only the affected phase is turned off.

MCC_GDRV_OCP_X: Set the overcurrent parameters, LS_OCP_USE_VDS_X, Threshold, Blanking and Deglitch according to the setup.

MCC_GDRV_CFG: Set VS_UVLO_LVL.

MCC_GDRV_STATUS_EN: Set the fields corresponding to the required protection mechanisms, this only enables reporting of the faults in MCC_GDRV_STATUS.

MCC_GDR_PROT_EN: Set the fields corresponding to the required protection mechanisms, this turns on the protection handling.

Handling a fault:

MCC_GDRV_STATUS: Check which fault occurred and solve the underlying problem. Clear the flags by writing a 1 to them.

MCC_PWM_CONFIG: Set CHOP to 1

MCC_GDRV_PROT_EN: Set to 0x00000000

MCC_GDRV_FAULT: Set to 0x000F000F to resume gate driver operation, wait for the bootstrap capacitors to recharge.

MCC_GDRV_PROT_EN: Reenable the required protections

MCC_PWM_CONFIG: Set CHOP to 7 to start switching and resume the application.

PWM Engine Setup

The TMC9660 includes a pulse width modulation (PWM) engine to generate the desired output voltage for each motor phase. The configuration is done in the MCC_PWM_CONFIG register. By default, the PWM engine is turned off. It can be enabled by writing the value 7 in the CHOP field. This starts the PWM with the duty cycles provided by the current control output. Different modes are also available in the CHOP field. However, these modes are not used for normal operation.

The frequency of the PWM can be configured in the MCC_PWM_MAXCNT register. This value defines the clock divider for the 120MHz PWM clock. The default value for the PWM frequency is 25kHz. It can be calculated with the following formula.

$$f_{PWM} = \frac{120MHz}{MCC\_PWM\_MAXCNT}$$

The functionality of the PWM engine and the configuration is slightly different for different motor types. Depending on the selected TYPE field in the MCC_MOTOR_CONFIG register, the required configuration is described in the corresponding sections that follow.

3-Phase BLDC Motor Configuration

The SV_MODE field in the MCC_PWM_CONFIG register sets the modulation mode. The 3-phase BLDC motor supports four different options.

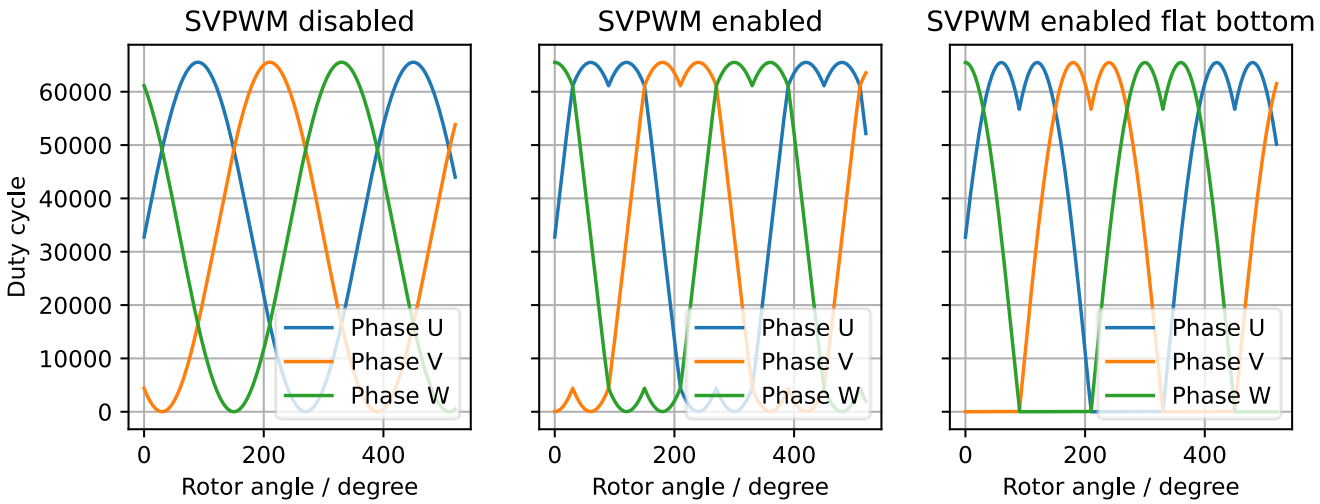

Figure 4. PWM modes for 3-phase BLDC motor

The default mode generates normal sine waves as shown in the left plot of *Figure 4*. If Space Vector PWM (SVPWM) with third harmonic injection is enabled, the generated voltage waveforms look like the plot in the middle. This modulation helps to generate higher peak voltage effectively. SVPWM is the recommended configuration. The third option is the flat bottom modulation as shown in the right plot. The last option is the flat bottom modulation with offset. The offset value can be configured in the DUTY_CYCLE_OFFSET field.

The field MEASUREMENT_MODE in the MCC_ADC_I_GEN_CONFIG register specifies the current measurement mode. If the bottom shunt measurement with automatic switching is selected, the MCC_PWM_SWITCH_LIMIT register must also be set for the PWM configuration. Usually, all three phase currents of the motor are measured for the current control. Because of the bottom shunt measurement, this is not possible for very high duty cycles. If the duty cycle of one phase is very high, the measured current is corrupted because of the settling time of the CSA. To

avoid this, the MCC_PWM_SWITCH_LIMIT sets an upper threshold for the duty cycle. If the duty cycle of one phase exceeds the switch limit, this phase current is not measured. Instead, the other two phases are used to calculate the third one. The maximum switch limit can be calculated with the following formula.

$$\text{MCC\_PWM\_SWITCH\_LIMIT}_{max} = (1 - 2 \times T_{set} \times f_{PWM}) \times 2^{16}$$

It depends on the configured PWM frequency f_{PWM} and the settling time T_{set} of the CSA. The MCC_PWM_SWITCH_LIMIT register should be set below this maximum value to avoid corrupted current measurements. The settling time of the internal CSA depends on the filter configuration and the gain factor in the CSA_SETUP register. For more information, see the *Current sensing and Analog Measurement Setup* section. *Table 9* shows the worst-case settling time for the different settings.

Table 9. CSA settling time Tset worst case

CSA SETTLING TIME	GAIN x5	GAIN x10	GAIN x20	GAIN x40
Filter 0	0.622µs	0.652µs	0.708µs	1.080µs
Filter 1	0.593µs	0.838µs	1.150µs	1.410µs
Filter 2	0.731µs	1.280µs	1.940µs	2.230µs
Filter 3	0.900µs	1.690µs	2.760µs	3.040µs

The minimum PWM switch limit is calculated with the equations listed in *Table 10*.

Table 10. Equations for minimum PWM switch limit

SV_MODE	MINIMUM PWM SWITCH LIMIT	INTERNAL VOLTAGE LIMIT
0: SVPWM disabled	$\text{MCC\_PWM\_SWITCH\_LIMIT}_{min} = 24576 + 1.5 \times MCC\_PID\_UQ\_UD\_LIMITS$	16383
1: SVPWM enabled	$\text{MCC\_PWM\_SWITCH\_LIMIT}_{min} = 32768 + 1.5 \times MCC\_PID\_UQ\_UD\_LIMITS$	18900
2: SVPWM flat bottom	$\text{MCC\_PWM\_SWITCH\_LIMIT}_{min} = 3 \times MCC\_PID\_UQ\_UD\_LIMITS$	18900

The minimum switch limit depends on the selected SV_MODE and the configured MCC_PID_UQ_UD_LIMITS. To ensure that there is no point in the waveform when two phases are above the switch limit at the same time it is recommended to set the MCC_PWM_SWITCH_LIMIT register above the calculated minimum value. If it is not possible to set the MCC_PWM_SWITCH_LIMIT register in between the minimum and maximum value, the output voltage limit should be adjusted in the MCC_PID_UQ_UD_LIMITS register to reduce the minimum switch limit. For this it is important to know that the voltage limit has an internal maximum value that also depends on the selected SV_MODE as shown in the table. Changing MCC_PID_UQ_UD_LIMITS above this value has no effect.

Each output phase must also be enabled individually using the enable fields in the MCC_PWM_CONFIG register. For a 3-phase motor this means ENABLE_UX1, ENABLE_VX2 and ENABLE_WY1 must be set to 1. The high-side and low-side outputs for the Y2 port are not needed for a BLDC motor. This port can be disabled or used for the external mode as described in the section *External PWM Mode*.

Two-Phase Stepper Motor Configuration

For a two-phase stepper motors the third harmonic injection is not available. The only option in the SV_MODE field of the MCC_PWM_CONFIG register is the default modulation with sine waves and the flat bottom modulation as shown in *Figure 5*.

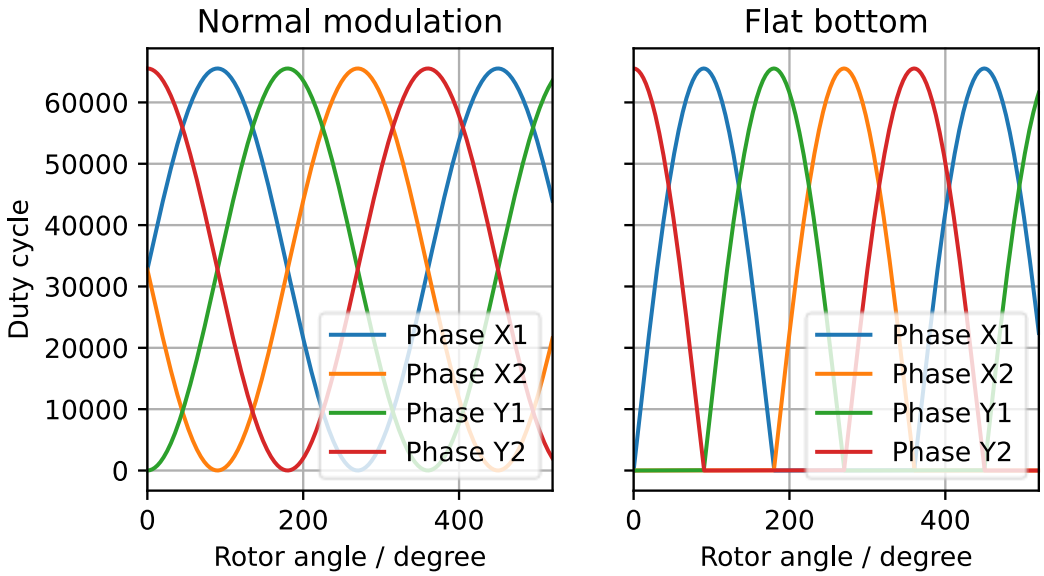

Figure 5. PWM modes for 2-phase stepper motor

The field MEASUREMENT_MODE in the MCC_ADC_I_GEN_CONFIG specifies the current measurement mode. If the bottom shunt measurement with automatic switching is selected, the MCC_PWM_SWITCH_LIMIT register must also be set for the PWM configuration. Usually, the phase currents are measured at each channel connected to the stepper motor. This means that both phase currents are measured redundantly. When the duty cycle of one phase is very high, the current measurement is corrupted. To avoid this, the MCC_PWM_SWITCH_LIMIT should be around 50% of the PWM duty cycle for the normal modulation mode. With the 16-bit scaling this would be 32768. If the flat bottom modulation is selected, the MCC_PWM_SWITCH_LIMIT can be set to zero.

To drive a stepper motor, all four output ports are needed. This means ENABLE_UX1, ENABLE_VX2, ENABLE_WY1 and ENABLE_Y2 must all be set to 1.

DC Motor Configuration

The TMC9660 can drive a DC motor using the X1 and X2 channels. The ENABLE fields in the MCC_PWM_CONFIG register must be set accordingly. The channels that are not used can be setup using the external PWM mode as described in the section *External PWM Mode*.

The SV_MODE field of the MCC_PWM_CONFIG register can be used to configure the modulation mode of the PWM. The available modes are identical to the stepper motor configuration as shown in *Figure 5*. Though the outputs are usually DC voltages instead of sine waves.

The field MEASUREMENT_MODE in the MCC_ADC_I_GEN_CONFIG specifies the current measurement mode. If the bottom shunt measurement with automatic switching is selected, the MCC_PWM_SWITCH_LIMIT register must also be set for the PWM configuration. Usually, the phase currents are measured at each channel connected to the DC motor. This means that the current is measured redundantly. When the duty cycle of one channel is very high, the current measurement is corrupted. To avoid this, the MCC_PWM_SWITCH_LIMIT should be around 50% of the PWM duty cycle for the normal modulation mode. With the 16-bit scaling this would be 32768. If the flat bottom modulation is selected, the MCC_PWM_SWITCH_LIMIT can be set to zero.

External PWM Mode

In normal use case the duty cycle for the PWM outputs is calculated automatically based on the output voltage of the current controller. Alternatively, it is also possible to bypass the control loops and directly set a specific duty cycle for each output channel. This can be enabled for each channel individually using the EXT_ENABLE fields in the MCC_PWM_CONFIG register. Note that the corresponding ENABLE field for the same channel must also be set to enable the output.

If the EXT_ENABLE mode is active for one channel, the duty cycle can be set in the register MCC_PWM_VX2_UX1_EXT and MCC_PWM_Y2_WY1_EXT for each channel accordingly. After that the high-side and low-side of this channel are driven as defined by the duty cycle. The duty cycle in these registers is automatically scaled to the PWM frequency. This means a value of 0xFFFF generates a duty cycle of 100% independently of the MCC_PWM_MAXCNT register.

Because the Y2 channel is not used when diving a BLDC motor, this channel can be used for different functions. If EXT_ENABLE_Y2 is set, a fixed duty cycle can be set for this channels high-side and low-side outputs as described. This can be done while driving a BLDC motor using the other three channels. If the Y2_HS_SRC field in the MCC_PWM_CONFIG register is set to 1, the high-side and low-side can be configured independently. In that case the Y2 low-side uses the duty cycle defined in the MCC_PWM_Y2_WY1_EXT field and the high-side duty cycle $D_{Y2,HS}$ can be set in the MCC_PWM_EXT_Y2_ALT register. Note that this register value is not automatically scaled to the active PWM frequency. To match the scaling used for the other registers, the following equation can be used to adjust the duty cycle to the PWM frequency.

$$\text{MCC\_PWM\_EXT\_Y2\_ALT} = \frac{\text{D}_{Y2,HS}}{\%} \times \frac{\text{MCC\_PWM\_MAXCNT} + 1}{100}$$

Step-by-Step Setup: PWM Engine

Example step-by-step guide to set up the PWM engine.

Prerequisites:

- Follow *Step-by-Step Setup: Gate Driver* → Startup.
- Follow *Basic Current-Sense Amplifier Configuration* and *Current Assignment* description. (Only if current measurement is required.)
- MCC_MOTOR_CONFIG: Select Motor type and number of pole pairs.

Basic configuration:

- MCC_PWM_MAXCNT: Set PWM frequency.
- MCC_PWM_SWITCH_LIMIT: Set PWM switch limit as described in this section.
- MCC_PID_UQ_UD_LIMITS: Limit output voltage if required by PWM switch limit.
- MCC_PWM_CONFIG: Enable center PWM for FOC. Select PWM mode. Enable required PWM channels.

FIELD ORIENTED CONTROL ENGINE SETUP

The TMC9660 utilizes a Motion Control Core (MCC) that includes the control structures to drive different motor types using different motion modes in closed loop. This section provides a detailed description about the Field Oriented Control engine and how to use it for efficient motion control operations.

Some general configuration is done in the MCC_MOTOR_CONFIG register. It holds the value for the selected motor type and the number of pole pairs of the motor. Supported motor types are three-phase BLDC motor, two-phase stepper motor, and single-phase DC motor. The number of pole pairs is only relevant for BLDC and stepper motors.

The MCC_MOTION_CONFIG register holds the MOTION_MODE field which is used to set the active motion mode. The control loops are disabled in the default stopped mode. The torque, velocity and position modes are used to drive the motor closed loop using the internal controllers. More information about the control structure is given in the corresponding sections about *Current Control*, *Velocity Control* and *Position Control*. The motion mode for external voltage can be used for open-loop operation when no motor position feedback is available. For more information, see the *Open-Loop Voltage Mode* section. The pseudorandom binary sequence (PRBS) motion modes are described in the *PRBS Generator* section.

A Q notation is used to describe the fixed-point number scaling of some internal registers in the MCC. The notation comprises a Q followed by one or two numbers like this “Qm.n” or “Qn”. The optional m describes the number of integer bits while the n describes the number of fraction bits.

For example, Q4.20 is referring to a 24-bit integer value. In this case 4 bits are used for the integer part, including the additional singed bit if applicable, and 20 bits for the fraction. The conversion between the real input number A_{real} and the internal register value B_{reg} is described in the following equation.

$$B_{reg} = A_{real} \times 2^{n}$$

Open-Loop Voltage Mode

If no feedback for the motor position is available, the TMC9660 can still drive any motor in open-loop configuration. For this the ramp generator is used to create a rotating angle. More information on how to configure the ramp generator can be found in *Ramp Generator* section. While in VOLTAGE_EXT motion mode, the output voltage can be set manually. If the output voltage is high enough, the rotor follows the angle provided by the ramp generator. The user can set the output voltages UQ and UD in the register MCC_VOLTAGE_EXT. More information about these voltages is provided in *FOC transformation setup* section.

This configuration creates a rotating magnetic field in the motor. The rotational velocity of this magnetic field is defined by the target velocity of the ramp generator MCC_RAMPER_V_TARGET. To convert this velocity into real world units the following conversion factor can be used.

$$k_{RPM_{openloop}} = \frac{2^{40}}{40MHz \times 60} \approx 458.13$$

More information about the conversion factor k_{RPM} can be found in the *Velocity scaling* section. This conversion factor for open loop is only applicable for the ramp generator velocity if phi_e_ramp is selected in register MCC_PHI_E_SELECTION.

Whether the rotor can follow the rotating magnetic field or not depends on the amplitude of the output voltage. If the rotor cannot follow the magnetic field, the amplitude must be increased.

When using this configuration to initialize the encoder feedback, it is recommended to set UQ to zero and UD to a positive value that is high enough to turn the motor smoothly. The rotor angle then follows the ramp generator angle very closely. After that the encoder angle can be aligned to the ramp generator angle to initialize the feedback for the rotor position.

Step-by-Step Setup: Open-Loop Voltage Mode

The basic register configuration to run a motor in open-loop voltage mode is provided.

- Follow *Step-by-Step Setup: Gate Driver* → Startup.
- Follow *Step-by-Step Setup: PWM Engine*.
- Configure ramp generator for velocity mode. See *Ramp Generator* section for details.
- MCC_MOTOR_CONFIG: Select Motor type and number of pole pairs.
- MCC_PHI_E_SELECTION: Select phi_e_ramp.
- MCC_MOTION_CONFIG: Select voltage external motion mode. Enable ramp generator in velocity mode.
- MCC_VOLTAGE_EXT: Enter output voltage amplitude for UQ and UD.
- MCC_RAMPER_V_TARGET: Enter target velocity for ramp generator to generate rotating angle phi_e_ramp.

Current Control

The current control loop is implemented as FOC. This includes the required transformations of the measured currents as well as two PI controllers. One for the magnetic flux and one for the torque. The input of the current controllers are the actual motor currents calculated based on the phase current measurement. The output is the voltage applied to the motor, which is generated by a PWM channel for each motor phase. The control structure is shown in *Figure 6*.

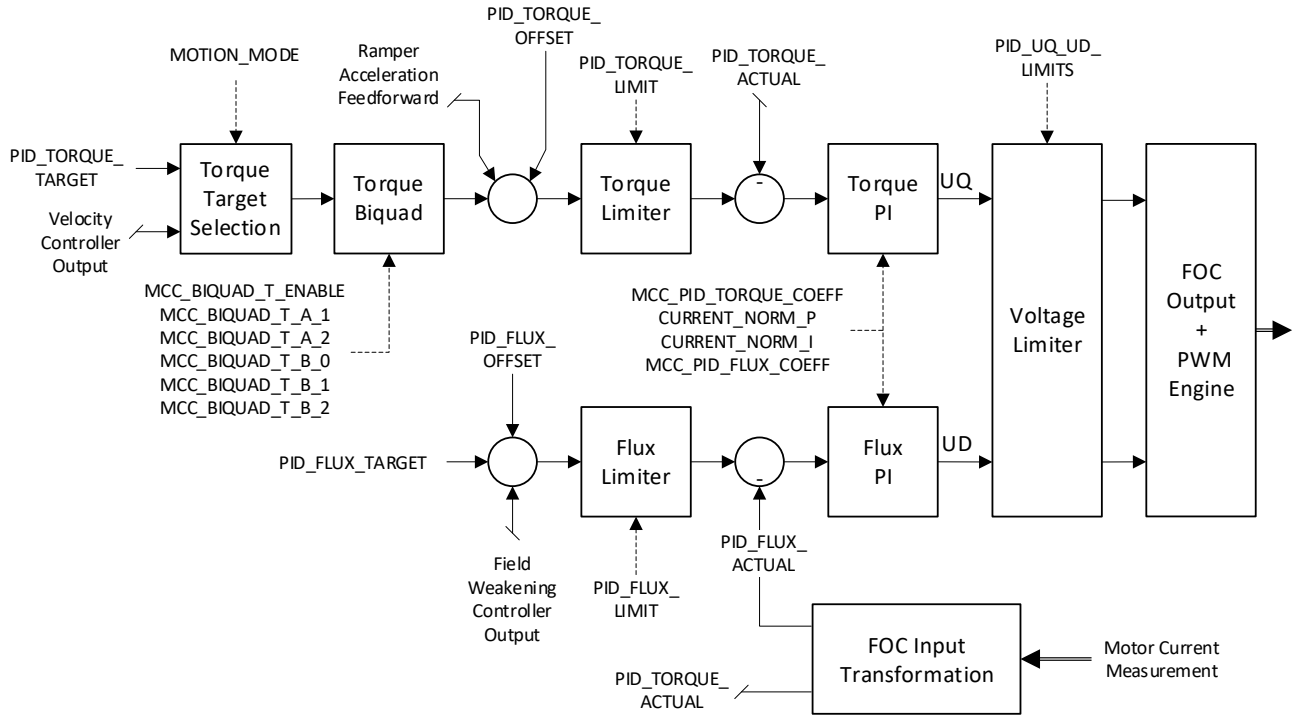

Figure 6. Torque and flux control loop

Current scaling

The TMC9660 uses an internal current scaling for the current PI controller, including the FOC transformation. The scaling applied results from the configured current sensing. As described in *Measured Current Values*. The following equation can be used to convert any internal current to milliampere.

$$I_{mA} = I_{internal} \times \frac{625}{16382 \times R_{shunt\ \Omega} \times A_{CSA\ gain}}$$

$I_{internal}$ is referring to any register that holds a current value with internal scaling. For example, MCC_PID_TORQUE_FLUX_ACTUAL and MCC_PID_TORQUE_FLUX_TARGET for the PI controller actual and target values, MCC_FOC_IBETA_IALPHA and MCC_FOC_IQ_ID for the FOC transformation outputs.

$R_{shunt\ \Omega}$ is referring to the shunt resistor and $A_{CSA\ gain}$ to the selected gain of the CSA. More information about these can be found in the *Current sensing and* Analog Measurement Setup section.

FOC transformation setup

The transformations of the motor currents are done automatically depending on the selected motor type. For a BLDC or stepper motor it is required to select the commutation angle for the electrical motor angle PHI_E. This can be done in the PHI_E_SELECTION register. PHI_E must be provided by an encoder interface like ABN or Hall to set up closed-loop current control. Alternatively, the ramp generator also provides an angle RAMPER_PHI_E that can be used for open-loop operation. It is also possible to generate the angle PHI_E externally and then provide it via the PHI_E_EXT field in the MCC_PHI_EXT register.

The input transformation consists of the Clarke and Park transformation. The input currents are taken from the registers MCC_ADC_IWY_IUX and MCC_ADC_IV. The outputs of the Clarke transformation are the currents IALPHA and IBETA which can be read in the MCC_FOC_IBETA_IALPHA register. The outputs of the Park transformation are the currents ID and IQ. They can be read in the register MCC_FOC_IQ_ID and are used as input values for the flux and torque PI controllers.

The outputs of the current control are the voltages UD and UQ and can be read in the register MCC_FOC_UQ_UD. These values are limited using the maximum voltage defined in the register MCC_PID_UQ_UD_LIMITS. The resulting voltages can be read in the MCC_FOC_UQ_UD_LIMITED register. These voltages are used as input for the inverse Park and inverse Clarke transformation. The outputs of the inverse Park transformation are the voltages UALPHA and UBETA which can be read in the register MCC_FOC_UBETA_UALPHA. The outputs of the inverse Clarke transformation are UUX, UV, UWY and can be read in the registers MCC_FOC_UWY_UUX and MCC_FOC_UV. These voltages are then used to calculate the PWM duty cycles for all output phases. They can be read in the registers MCC_PWM_VX2_UX1 and MCC_PWM_Y2_WY1.

Which transformations are used depends on the selected motor type. When using a three-phase BLDC motor, all transformations are used as described. When using a stepper motor, the Clarke and inverse Clarke transformations are skipped. They are not required for two-phase stepper motors. When the DC motor type is selected, no transformation is used at all. This also means that for a DC motor the PHI_E_SELECTION is not required.

Some of the registers mentioned here are referring to the individual motor phases. In these cases, the terms U, V and W are referring to the 3 phases of the BLDC motor if that motor type is selected. The terms X and Y are referring to the two phases of a stepper motor. This means that depending on the selected motor type not all register fields are used.

Based on the input current and output voltage, the following values for total motor power are calculated:

$$U_S = \sqrt{U_D^2 + U_Q^2} \qquad I_S = \sqrt{I_D^2 + I_Q^2} \qquad P_{Motor} = U_S \times I_S$$

The results can be read in the registers MCC_U_S_ACTUAL_I_S_ACTUAL and MCC_P_MOTOR. Note that U_S_ACTUAL is also used as actual value for the field weakening controller as described in *Field weakening*. The other values are for information only.

PI controller configuration

The P and I coefficients for the corresponding controllers must be set in the registers MCC_PID_FLUX_COEFF and MCC_PID_TORQUE_COEFF accordingly. The internal scaling of the coefficients can be adjusted using the fields CURRENT_NORM_P and CURRENT_NORM_I in the MCC_PID_CONFIG register. The normalization values for the current control are applied to both PI controllers for torque and flux and sets the shift factor at the end of the PI calculations. A shift factor of 8 results in a Q8.8 representation and shift factor of 16 results in a Q16 representation of the corresponding PI coefficient.

If the torque mode is used in the MOTION_MODE register, the target values for both PI controllers can be set in the MCC_PID_TORQUE_FLUX_TARGET register. It contains a field for the PID_TORQUE_TARGET and PID_FLUX_TARGET value. For normal FOC operation it is usual to use a PID_FLUX_TARGET value of zero and then set the desired target current with the PID_TORQUE_TARGET value. An offset can be entered for both target values individually in the MCC_PID_TORQUE_FLUX_OFFSET register. The MCC_PID_TORQUE_FLUX_LIMITS register contains limit values for the torque and flux controller. The values are applied to limit the target values that are used inside the controllers.

For other motion modes the source of the target value may change. The target values that are used for both controllers can be read in the MCC_PIDIN_TORQUE_FLUX_TARGET register before any offsets, filter and limits are applied. The register MCC_PIDIN_TORQUE_FLUX_TARGET_LIMITED on the other hand shows the target values that are actually used in the PI controllers. This includes the biquad filter for the torque controller, any offsets or feedforward values added to the target values and the limit values applied for both controllers.

As a reference it is possible to read out the actual values that are used for both controllers in the MCC_PID_TORQUE_FLUX_ACTUAL register. The scaling of these currents is analog to the selected scaling for the ADC current measurement. The error values resulting from the difference of target and actual value can be read out in the registers MCC_PID_TORQUE_ERROR and MCC_PID_FLUX_ERROR. The current integrator values for both controllers can also be read back with the MCC_PID_TORQUE_INTEGRATOR and MCC_PID_FLUX_INTEGRATOR registers. Note that the user can overwrite the integrator registers to pre-load specific values if needed.

The outputs of the current controllers are the voltages UQ und UD from the torque and flux controller accordingly. The maximum output voltage limit can be configured with the MCC_PID_UQ_UD_LIMITS register. How this voltage limit is applied depends on the selected motor type.

It is possible to bypass the entire control structure and directly set the output voltages for UQ and UD. This can be done in the register MCC_VOLTAGE_EXT. If the voltage external motion mode is used, these voltages are applied instead of the current controller output. This function is particularly useful for open-loop operation.

Step-by-Step Setup: Current Control

Example step-by-step guide to set up closed-loop torque control.

Prerequisites:

- Follow *Step-by-Step Setup: Gate Driver* → Startup.
- Follow *Basic Current-Sense Amplifier Configuration* and *Current Assignment* description.
- Follow *Step-by-Step Setup: Hall Feedback* or *Step-by-Step Setup: ABN Encoder*.
- Follow *Step-by-Step Setup: PWM Engine* → Basic configuration.

- MCC_MOTOR_CONFIG: Select Motor type and number of pole pairs.

Basic configuration:

- MCC_PHI_E_SELECTION: Select phi_e_abn or phi_e_hall.
- MCC_PID_CONFIG: Adjust current control P and I normalization if needed.
- MCC_PID_FLUX_COEFF: Set P and I coefficient for flux controller.
- MCC_PID_TORQUE_COEFF: Set P and I coefficient for torque controller.
- MCC_PID_UQ_UD_LIMITS: Set output voltage limit.
- MCC_PID_TORQUE_FLUX_LIMITS: Set target torque and flux limit.

Torque mode specific configuration:

- MCC_MOTION_CONFIG: Select torque mode as motion mode.
- MCC_PID_TORQUE_FLUX_TARGET: Set target flux to 0 and target torque to expected target current.

Field weakening

An additional and optional field weakening controller is also available. It is implemented as PI controller and can be used to generate a negative target flux as shown in *Figure 7*. This can increase the maximum achievable velocity at the expense of higher power consumption.

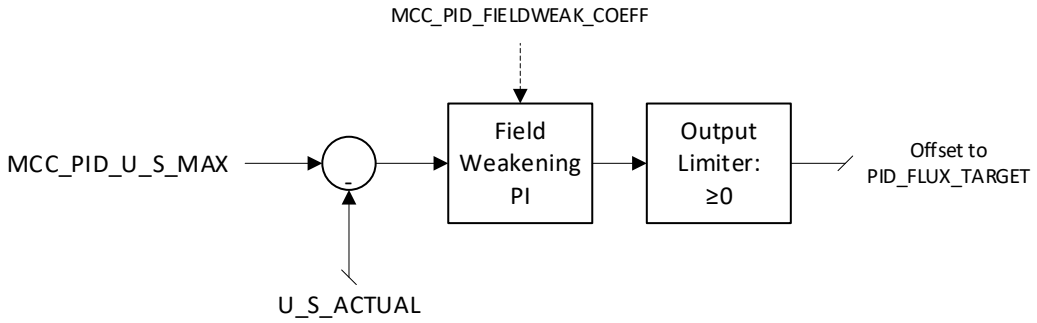

Figure 7. Field weakening control loop

To enable the field weakening the PI coefficients must be set in the MCC_PID_FIELDWEAK_COEFF register. Both coefficients are scaled with the Q16 format. In addition to that the MCC_PID_U_S_MAX register value must be set below the active voltage limit. In most cases the internal voltage limit is 16383. The only exception to this is when the SVPWM mode is enabled for a BLDC motor. In that case internal voltage limit is 18900. If the MCC_PID_UQ_UD_LIMITS register value is set below the internal voltage limit, this value is used instead. The actual voltage U_S_ACTUAL is calculated based on the absolute value of output voltages from the torque and flux controllers.

Torque feedforward

The torque PI controller has an addition optional offset input to the target value. It can be enabled in the FEEDFORWARD field of the MCC_MOTION_CONFIG register. If enabled the acceleration of the ramp generator is used as feedforward value for the torque controller. The current acceleration can be read in the MCC_RAMPER_A_ACTUAL register. Because the acceleration is using a different scaling compared to the motor current it is required to adapt the acceleration here. The register MCC_RAMPER_ACC_FF holds a value for the SHIFT factor and the GAIN to scale the acceleration before it is used as feedforward for the torque controller. The following equation for the scaling is given to calculate the feedforward value for the torque controller $I_{feedforward}$.

$$I_{feedforward} = (MCC\_RAMPER\_A\_ACTUAL \times GAIN) >> (SHIFT \times 4)$$

Open-Loop Current Control

It is also possible to generate the commutation angle phi_e with the ramp generator instead of any encoder feedback to drive the motor in open-loop mode. In addition to the open-loop voltage mode described in *Open-Loop Voltage Mode*, it is possible to combine this with the current control.

This configuration creates a rotating magnetic field in the motor. The rotational velocity of this magnetic field is defined by the target velocity of the ramp generator MCC_RAMPER_V_TARGET. To convert this velocity into real word units the following conversion factor can be used.

$$k_{RPM_{openloop}} = \frac{2^{40}}{40MHz \times 60} \approx 458.13$$

More information about the conversion factor k_{RPM} can be found in *Velocity scaling*. This conversion factor for open loop is only applicable for the ramp generator velocity if phi_e_ramp is selected in register MCC_PHI_E_SELECTION.

Step-by-Step Setup: Open-Loop Current Control

An example step-by-step guide to set up open-loop current control is provided as follows:

Prerequisites:

- Follow *Step-by-Step Setup: Gate Driver* → Startup.
- Follow *Basic Current-Sense Amplifier Configuration* and *Current Assignment* description.
- Follow *Step-by-Step Setup: PWM Engine* → Basic configuration.
- Configure ramp generator for velocity mode. See *Ramp Generator* section for details.
- MCC_MOTOR_CONFIG: Select Motor type and number of pole pairs.

Basic configuration:

- MCC_PHI_E_SELECTION: Select phi_e_ramp.
- MCC_PID_CONFIG: Adjust current control P and I normalization if needed.
- MCC_PID_FLUX_COEFF: Set P and I coefficient for flux controller.
- MCC_PID_TORQUE_COEFF: Set P and I coefficient for torque controller.
- MCC_PID_UQ_UD_LIMITS: Set output voltage limit.
- MCC_PID_TORQUE_FLUX_LIMITS: Set target torque and flux limit.
- MCC_MOTION_CONFIG: Select torque mode as motion mode. Enable ramp generator in velocity mode.
- MCC_RAMPER_V_TARGET: Enter target velocity for ramp generator to generate rotating angle phi_e_ramp.
- MCC_PID_TORQUE_FLUX_TARGET: Set target flux and torque as desired.

Velocity Control

The velocity control enables to run a motor at a target rotational speed. The velocity control loop is built around the current controller. This means to use the velocity control the current control must also be configured. The implementation is in form of a PI controller. The input velocity of the controller is the actual motor velocity. If the velocity controller is enabled, depending on the selected motion mode, the output is used as target value for the torque current controller. The control structure is shown in *Figure 8*.

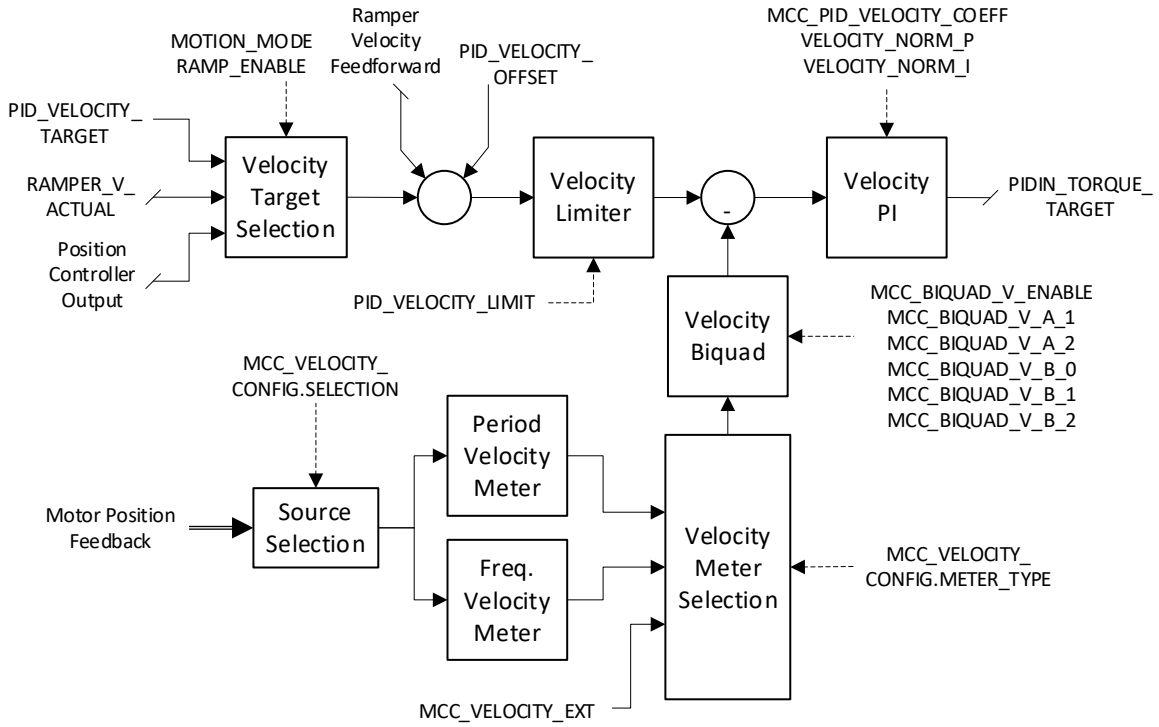

Figure 8. Velocity control loop

Velocity acquisition

To enable the velocity control, it is required to configure the feedback path for the velocity acquisition of the motor. The SELECTION field of the MCC_VELOCITY_CONFIG register is used to select the feedback source. This can be for example the rotor position based on the Hall or ABN encoder feedback. Note that multiple selections are available for each feedback system. The difference between the selections is the scaling that is applied to the raw position feedback from the encoder. This makes it possible to select the scaling that fits best to the entire control structure and the user specific setup. Either way, it is recommended to select abn_count when using an ABN encoder as feedback or hall_count for hall sensor feedback.

In the same register the METER_TYPE field can switch between two different velocity meter options. VELOCITY_PER and VELOCITY_FREQ. The VELOCITY_PER selection is using a period time measurement between two consecutive rotor positions to calculate the rotor velocity after each position change. The VELOCITY_FREQ is using a fixed frequency to update the rotor velocity based on the position change. It is recommended to select the options that is best suited to the velocity range of the desired application. To compare and evaluate which velocity meter is better suited for a given application, both values are constantly calculated and written into the MCC_VELOCITY_PER and MCC_VELOCITY_FRQ register accordingly.

The register MCC_V_MIN_POS_DEV_TIME_COUNTER_LIMIT provides some additional configuration for the VELOCITY_PER selection. The values in this register can be adjusted to the selected encoder feedback and scaling to improve velocity acquisition for a specific application. Another option for the VELOCITY_PER selection is an optional moving average filter. It can be enabled and configured in the MCC_VELOCITY_SCALING register. By default, it is disabled. Depending on the selected feedback system, the measured velocity is often quite noisy. The moving average filter can be used to reduce this noise and improve the controller behavior. The VELOCITY_FREQ selection has an additional scaling factor as configuration. It can be set freely in the MCC_VELOCITY_SCALING register. It is recommended to align the internal velocity scaling of the VELOCITY_FREQ meter to the VELOCITY_PER

meter. This way the same equations for unit conversion can be used and it is possible to switch between both selections dynamically during operation. The following formula can be used to calculate the scaling factor.

$$MCC\_VELOCITY\_SCALING = \frac{2^{24} \times f_{Velo}}{40MHz}$$

The update rate of the velocity controller f_{Velo} is given in *PI controller configuration*.

In general, the VELOCITY_PER selection is suited better for low velocity and more accurate in most cases. There are some limitations for the VELOCITY_PER meter when the input frequency of the encoder steps is very high. This is usually only the case when using an ABN encoder with high resolution in combination with a high rotational speed of the motor.

All formulas that follow require either the abn_count or hall_count in the SELECTION field of the MCC_VELOCITY_CONFIG register. The results are calculated in the unit RPM. The conversion to and from the internally used velocity scaling is described in *Velocity scaling*.

The following formula can be used to calculate the maximum motor velocity in RPM that the VELOCITY_PER meter can measure without distortion. Note that this formula is only applicable if the moving average filter is disabled. This means MOVING_AVRG_FILTER_SAMPLES must be set to zero.

$$V_{PerMax_{RPM}} = \frac{40MHz \times V\_MIN\_POS\_DEV \times 60}{53 \times CPR}$$

If the moving average filter for the VELOCITY_PER selection is enabled, the equation to calculate the maximum measurable velocity is given in the formula that follows. This is applicable when MOVING_AVRG_FILTER_SAMPLES is set to anything but zero.

$$V_{PerMax_{RPM}} = \frac{40MHz \times V\_MIN\_POS\_DEV \times 60}{105 \times CPR}$$

CPR in these equations is referring to the counts per revolution. For an ABN encoder this is the same as for the register MCC_ABN_CPR. For hall sensors the value is given by the following formula.

$$CPR_{Hall} = 6 \times N\_POLE\_PAIRS$$

If the velocity range for the given application is expected anywhere close to the calculated $V_{PerMax_{RPM}}$, it is recommended to either increase V_MIN_POS_DEV or switch to the VELOCITY_FREQ meter. Both changes can be done dynamically during operation. Considering MCC_VELOCITY_SCALING is configured accordingly.

The resolution of the VELOCITY_PER meter is getting worse with higher velocity, while the resolution of the VELOCITY_FREQ meter is constant. The following equation can be used to calculate the crossover point (COP) when the VELOCITY_FREQ selection becomes better.

$$V_{COP_{RPM}} = 60 \times \frac{f_{Velo} + \sqrt{f_{Velo}^2 + f_{Velo} \times 40MHz \times V\_MIN\_POS\_DEV \times 4}}{2 \times CPR}$$

Note that the equation for $V_{COP_{RPM}}$ relies on ideal position feedback from the encoder. The VELOCITY_PER meter is very sensitive to noisy or inaccurate encoder feedback. This means the best crossover point may be lower in the actual application. For that it is required to compare both velocity meter outputs manually in the actual application and velocity range. The registers MCC_VELOCITY_PER and MCC_VELOCITY_FRQ can be used for that.

Depending on the configuration, $V_{COP_{RPM}}$ can be higher or lower compared to $V_{PerMax_{RPM}}$. This means both thresholds need to be considered when selecting the most accurate velocity meter. The approximate curve of the

noise performance over the motor velocity is displayed for both velocity meters in the *Figure 9*. In this case a low noise performance is desirable.

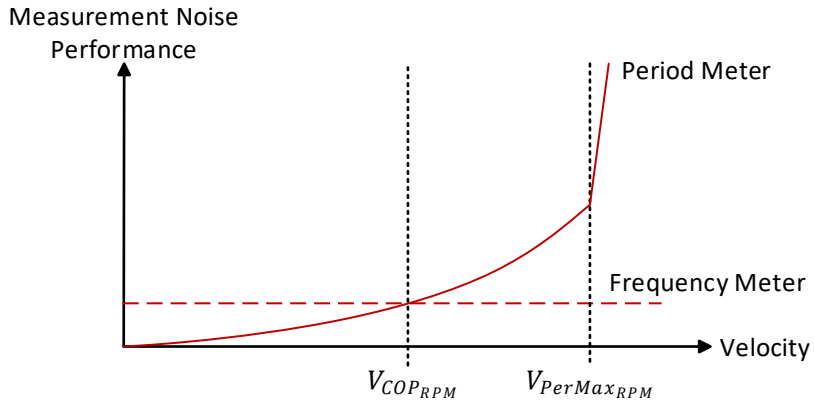

Figure 9. Velocity meter noise performance

The noise performance of the period meter becomes worse for higher velocity while the frequency meter is constant. The position of $V_{COP_{RPM}}$ and $V_{PerMax_{RPM}}$ may vary significantly depending on the actual setup and configuration.

Velocity scaling

The TMC9660 uses an internal velocity scaling for the velocity PI controller as well as for the ramp generator. This section provides the formulas to convert internal velocity values into real-world units. The scaling factor k_{RPM} can be used for that.

$$v_{RPM} = \frac{v_{internal}}{k_{RPM}}$$

The velocity $v_{internal}$ is referring to any register value that uses the internal velocity scaling. This includes for example MCC_PID_VELOCITY_ACTUAL and MCC_PID_VELOCITY_TARGET. In case the ramp generator is enabled to provide the target value for the velocity PI controller, the same scaling also applies to the registers MCC_RAMPER_V_TARGET and MCC_RAMPER_V_ACTUAL.

The scaling factor k_{RPM} depends on the configuration of the SELECTION field in the MCC_VELOCITY_CONFIG register. Following are the equations listed for all relevant options. The index PER and FREQ are referring to the corresponding velocity meter options. Note that when using the recommended value for MCC_VELOCITY_SCALING, both velocity scaling factors are identical.

$$k_{RPM_{PER,phi\_e}} = 2^{16} \times N\_POLE\_PAIRS \times \frac{2^{24}}{40MHz \times 60}$$

$$k_{RPM_{FREQ,phi\_e}} = 2^{16} \times N\_POLE\_PAIRS \times \frac{MCC\_VELOCITY\_SCALING}{f_{Velo} \times 60}$$

These scaling factors that are calculated are used for the phi_e scaling of the velocity. This means it applies to all SELECTIONS that are referring to phi_e. For example, phi_e_abn or phi_e_hall. Phi_e scaling in this context means that one electrical revolution goes from zero to 65535.

$$k_{RPM_{PER,phi\_m}} = 2^{16} \times \frac{2^{24}}{40MHz \times 60} \approx 458.13$$

$$k_{RPM_{FREQ,phi\_m}} = 2^{16} \times \frac{MCC\_VELOCITY\_SCALING}{f_{Velo} \times 60}$$

These scaling factors that are calculated are used for the phi_m scaling of the velocity. This means it applies to all SELECTIONS that are referring to phi_m. For example, phi_m_abn. Phi_m scaling in this context means that one mechanical revolution goes from zero to 65535.

Note that when using the external register to provide the angle for the velocity feedback, by selecting phi_e_ext or phi_m_ext, the scaling may change. In that case the user is responsible to provide the correctly scaled angle to the MCC_PHI_EXT register. Otherwise, the velocity scaling changes accordingly.

$$k_{RPM_{PER,abn\_count}} = MCC\_ABN\_CPR \times \frac{2^{24}}{40MHz \times 60}$$

$$k_{RPM_{FREQ,abn\_count}} = MCC\_ABN\_CPR \times \frac{MCC\_VELOCITY\_SCALING}{f_{Velo} \times 60}$$

If the abn_count is selected as velocity feedback, this scaling factor that is calculated may be used.

$$k_{RPM_{PER,hall\_count}} = 6 \times N\_POLE\_PAIRS \times \frac{2^{24}}{40MHz \times 60}$$

$$k_{RPM_{FREQ,hall\_count}} = 6 \times N\_POLE\_PAIRS \times \frac{MCC\_VELOCITY\_SCALING}{f_{Velo} \times 60}$$

If the hall_count is selected as velocity feedback, this scaling factor that is calculated may be used.

PI controller configuration

The PI coefficients must be set in the MCC_PID_VELOCITY_COEFF register. The scaling of these coefficients can be adjusted with the VELOCITY_NORM_P and VELOCITY_NORM_I field in the MCC_PID_CONFIG register. The VEL_SCALE field specifies a shift factor for the velocity controller output. It can be adjusted to better match the scaling of the velocity controller output to the torque controller input. The combination of the normalization values and the VEL_SCALE describes the scaling format for the PI coefficients. For the P normalization value, a shift factor of 0, 8 (default), 16 or 24 can be selected. For the I normalization value, a shift factor of 8, 16 (default), 24 or 32 can be selected. The VEL_SCALE value is a 4-bit register and can be configured from 0 to 15 with a default value of 8. The VEL_SCALE shift factor is applied to both PI coefficients. The sum of both shift factors results in a Q16 representation for the P coefficient and Q24 for the I coefficient with the default register values. The VEL_SMPL can be used to reduce the update rate of the velocity PI controller. By default, the controller output is calculated once per PWM cycle. The update rate of the velocity controller is calculated in the following equation.

$$f_{Velo} = \frac{f_{PWM}}{VEL\_SMPL + 1}$$

If the velocity mode is used in the MOTION_MODE register, the target value for the velocity PI controller can be set in the MCC_PID_VELOCITY_TARGET register. If the ramp generator is enabled, the target value in velocity mode is taken from the ramp generator. This means the target velocity must be entered in the MCC_RAMPER_V_TARGET register instead. An offset can be provided to the target value with the MCC_PID_VELOCITY_OFFSET register. A limit for the maximum target velocity can be set in the MCC_PID_VELOCITY_LIMIT register. For other motion modes the source of the target value may change. The target value that is used for the velocity controller can be read in the MCC_PIDIN_VELOCITY_TARGET register before any offsets, filter and limits are applied. The register MCC_PIDIN_VELOCITY_TARGET_LIMITED on the other hand shows the target value that is actually used in the PI

controller. This includes any offsets or feedforward values added to the target value and the velocity limit value applied.

For reference, the actual velocity used in the PI controller can be read in the PID_VELOCITY_ACTUAL register. The error value resulting from the difference of target and actual value can be read out in the register MCC_PID_VELOCITY_ERROR. The integrator value of the controller can also be read back with the MCC_PID_VELOCITY_INTEGRATOR register. Note that the user can overwrite the integrator register to pre-load specific values if needed.

Step-by-Step Setup: Velocity Control

Example step-by-step guide to set up closed-loop velocity control.

Prerequisites:

- Follow *Step-by-Step Setup: Gate Driver* → Startup.
- Follow *Basic Current-Sense Amplifier Configuration* and *Current Assignment* description.
- Follow *Step-by-Step Setup: Hall Feedback* or *Step-by-Step Setup: ABN Encoder*.
- Follow *Step-by-Step Setup: PWM Engine* → Basic configuration.
- Configure ramp generator for velocity mode. See *Ramp Generator* section for details. (optional)
- MCC_MOTOR_CONFIG: Select Motor type and number of pole pairs.
- Follow *Step-by-Step Setup: Current Control* → Basic configuration.

Basic configuration:

- MCC_VELOCITY_CONFIG: Set SELECTION to appropriate value. Choose fitting METER_TYPE. Enable moving average filter if applicable (VELOCITY_PER only).
- MCC_VELOCITY_SCALING: Adjust velocity scaling if needed (VELOCITY_FREQ only).
- MCC_V_MIN_POS_DEV_TIME_COUNTER_LIMIT: Set fitting configuration (VELOCITY_PER only).
- MCC_PID_CONFIG: Adjust velocity control P and I normalization if needed. Adjust velocity controller down sampling factor if needed.
- MCC_PID_VELOCITY_COEFF: Set P and I coefficient for velocity controller.
- MCC_PID_VELOCITY_LIMIT: Set target velocity limit.

Velocity mode specific configuration with ramp generator disabled:

- MCC_MOTION_CONFIG: Select velocity mode as motion mode. Disable ramp generator.
- MCC_PID_VELOCITY_TARGET: Set target velocity to expected value.

Velocity mode specific configuration with ramp generator enabled (requires ramp generator configuration):

- MCC_MOTION_CONFIG: Select velocity mode as motion mode. Enable ramp generator in velocity mode.
- MCC_RAMPER_V_TARGET: Set ramp target velocity to expected value.

Velocity feedforward

The velocity PI controller has an addition optional offset input to the target value. It can be enabled in the FEEDFORWARD field of the MCC_MOTION_CONFIG register. If enabled the velocity of the ramp generator is used as feedforward value for the velocity PI controller. The current velocity of the ramp generator can be read in the MCC_RAMPER_V_ACTUAL register.

Position Control

The position control enables to rotate a motor to a target position. The position control loop is built around the velocity controller. This means to use the position control the velocity and current control must also be configured. The implementation is in form of a PI controller. Note that the integral part of the controller is usually not needed. For most applications a P controller is sufficient for the position control. The control structure is shown in *Figure 10*.

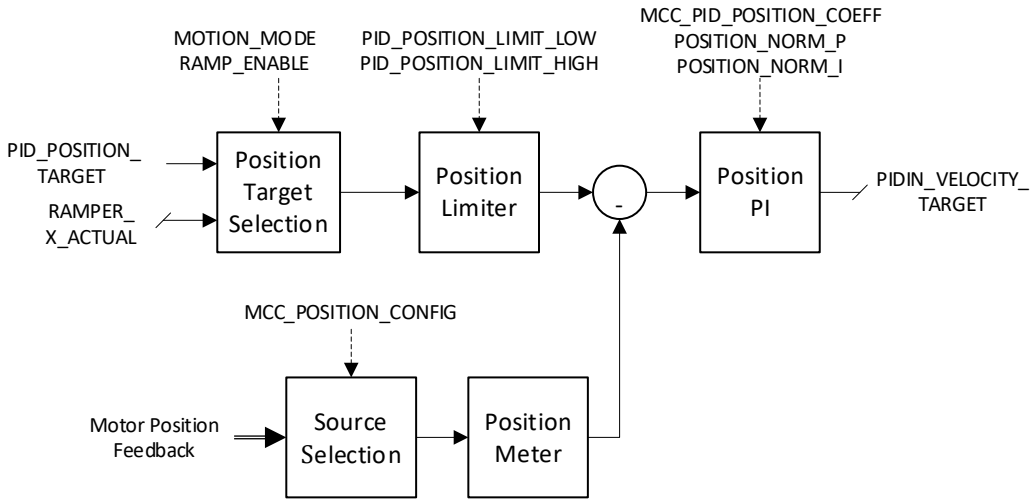

Figure 10. Position control loop

Position acquisition

To enable the position control, it is required to configure the feedback path for the position acquisition of the motor. The SELECTION field of the MCC_POSITION_CONFIG register is used to select the feedback source. This can be for example the rotor position based on the Hall or ABN encoder feedback. Note that multiple selections are available for each feedback system. The difference between the selections is the scaling that is applied to the raw position feedback from the encoder. This makes it possible to select the scaling that fits best to the entire control structure and the user specific setup. Alternatively, it is also possible to select the PHI_E_EXT or PHI_M_EXT values as input for the position meters. These values can be provided by the user in the MCC_PHI_EXT register.

Position scaling

The TMC9660 uses an internal position scaling for the position PI controller as well as for the ramp generator. This section provides the formulas to convert internal position values into real-world units. The scaling factor k_{rev} can be used for that. The result p_{rev} provides the position in full mechanical motor revolutions.

$$p_{rev} = \frac{p_{internal}}{k_{rev}}$$

The position $p_{internal}$ is referring to any register value that uses the internal position scaling. This includes for example MCC_PID_POSITION_ACTUAL and MCC_PID_POSITION_TARGET. In case the ramp generator is enabled to provide the target position for the position PI controller, the same scaling also applies to the registers MCC_RAMPER_X_TARGET and MCC_RAMPER_X_ACTUAL.

The scaling factor k_{rev} depends on the configuration of the SELECTION field in the MCC_POSITION_CONFIG register. Following are the equations listed for all relevant options.

$$k_{rev_{phi\_e}} = 2^{16} \times N\_POLE\_PAIRS$$

This factor is used for all SELECTION options that are referring to phi_e, like phi_e_abn and phi_e_hall. Phi_e scaling in this context means that one electrical revolution goes from zero to 65535.

$$k_{rev_{phi\_m}} = 2^{16}$$

This factor is used for the SELECTION options that is referring to phi_m, like phi_m_abn. Phi_m scaling in this context means that one mechanical revolution goes from zero to 65535.

Note that when using the external register to provide the angle for the position feedback, by selecting phi_e_ext or phi_m_ext, the scaling may change. In that case the user is responsible to provide the correctly scaled angle to the MCC_PHI_EXT register. Otherwise, the position scaling changes accordingly.

$$k_{rev_{abn\_count}} = MCC\_ABN\_CPR$$

$$k_{rev_{hall\_count}} = 6 \times N\_POLE\_PAIRS$$

If the abn_count or hall_count option is selected, the corresponding factor is applied.

PI controller configuration

The position PI coefficients must be set in the MCC_PID_POSITION_COEFF register. The scaling of these coefficients can be adjusted with the POSITION_NORM_P and POSITION_NORM_I field in the MCC_PID_CONFIG register. The normalization selects the shift factor at the end of the PI calculation. There are four different scaling formats available. For the P coefficient: Q16.0, Q8.8 (default), Q16 and Q24. For the I coefficient: Q8.8, Q16 (default), Q24 and Q32. The POS_SMPL can be used to reduce the update rate of the position PI controller. By default, the controller output is calculated once per PWM cycle.

If the position mode is used in the MOTION_MODE register, the target value for the position PI controller can be set in the MCC_PID_POSITION_TARGET register. If the ramp generator is enabled the target value is provided by that block instead. In that case the target value must be entered in the MCC_RAMPER_X_TARGET register. The actual position of the motor can be read in the MCC_PID_POSITION_ACTUAL register. This register value can be overwritten to align the current motor position to the application. Depending on the setting of the KEEP_POS_TARGET field in the MCC_PID_CONFIG register, the target position may also be set to the new actual position automatically. This can be done to avoid unintended motor movements when changing the actual position. An offset value can also be set for the actual motor position used by the controller in the MCC_PID_POSITION_ACTUAL_OFFSET register.

Two limit registers are provided for the target position: MCC_PID_POSITION_LIMIT_LOW and MCC_PID_POSITION_LIMIT_HIGH. These registers can be used to limit the position controller from reaching positions outside a predefined range. The target value that is used for the position controller can be read in the MCC_PIDIN_POSITION_TARGET register before any limit is applied. The register MCC_PIDIN_POSITION_TARGET_LIMITED on the other hand shows the target value that is actually used in the PI controller. This includes the upper and lower position limit applied to the target value.

The error and the integrator value of the position PI controller are provided in the registers MCC_PID_POSITION_ERROR and MCC_PID_POSITION_INTEGRATOR. Note that the user can overwrite the integrator register to pre-load specific values if needed.

While using the position mode it can become difficult to keep the motor completely still after the target position is reached. Depending on the system setup and the controller configuration it can happen that the motor jerks continuously and never come to a complete standstill. To avoid this, it is possible to configure the MCC_PID_POSITION_TOLERANCE and MCC_PID_POSITION_TOLERANCE_DELAY registers. Note that this feature only works if the target position is provided by the ramp generator. If correctly configured, the position control is disabled shortly after the target position is reached. Resulting in a jerk free standstill of the motor within the configured position tolerance band. As soon as the measured position leaves the tolerance band, the position controller is switched on again.

Step-by-Step Setup: Position Control

Example step-by-step guide to set up closed-loop position control.

Prerequisites:

- Follow *Step-by-Step Setup: Gate Driver* → Startup.
- Follow *Basic Current-Sense Amplifier Configuration* and *Current Assignment* description.
- Follow *Step-by-Step Setup: Hall Feedback* or *Step-by-Step Setup: ABN Encoder*.
- Follow *Step-by-Step Setup: PWM Engine* → Basic configuration.
- Configure ramp generator for position mode. See *Ramp Generator* section for details. (optional)
- MCC_MOTOR_CONFIG: Select Motor type and number of pole pairs.
- Follow *Step-by-Step Setup: Current Control* → Basic configuration.
- Follow *Step-by-Step Setup: Velocity Control* → Basic configuration.

Basic configuration:

- MCC_POSITION_CONFIG: Set SELECTION to appropriate value.
- MCC_PID_CONFIG: Adjust position control P normalization and down sampling factor if needed.
- MCC_PID_POSITION_COEFF: Set P coefficient for position controller. The I coefficient should be zero.
- MCC_PID_POSITION_TOLERANCE: Set position tolerance to appropriate value (only relevant with ramp generator).
- MCC_PID_POSITION_TOLERANCE_DELAY: Set delay to appropriate value (only relevant with ramp generator).
- MCC_PID_POSITION_LIMIT_LOW: Set position limit low.
- MCC_PID_POSITION_HIGH_LOW: Set position limit high.

Position mode specific configuration with ramp generator disabled:

- MCC_MOTION_CONFIG: Select position mode as motion mode. Disable ramp generator.
- MCC_PID_POSITION_TARGET: Set target position to expected value.

Position mode specific configuration with ramp generator enabled (requires Ramp Generator configuration):

- MCC_MOTION_CONFIG: Select position mode as motion mode. Enable ramp generator in position mode.
- MCC_RAMPER_X_TARGET: Set ramp target position to expected value.

Biquad Filters

The TMC9660 employs biquad filters to enhance the performance of its velocity and torque control loops. The biquad filter, essentially a second-order digital filter, operates by processing the incoming signal through a combination of current and past input samples, along with previous output samples. The filter's behavior, such as its role as a low-pass, high-pass, band-pass, or notch filter, is determined by a set of coefficients that are stored in dedicated register. The filter's output Y(n) at a given time step n is calculated using the following equation:

$$Y(n) = X(n) \times b_0 + X(n-1) \times b_1 + X(n-2) \times b_2 + Y(n-1) \times a_1 + Y(n-2) \times a_2$$

where:

- X(n) represents the current input sample.
- X(n-1) and X(n-2) are the previous input samples.
- Y(n-1) and Y(n-2) are the previous output samples.
- a_1, a_2, b_0, b_1, and b_2 are the filter coefficients.

The filter coefficients are 24-bit values normalized to a Q4.20 format.

The velocity biquad filter is used to filter the actual velocity of the motor that is used as input for the velocity controller. The exact location in the control loop can be seen in *Figure 8*. The filter coefficients can be set using these registers: MCC_BIQUAD_V_A_1, MCC_BIQUAD_V_A_2, MCC_BIQUAD_V_B_0, MCC_BIQUAD_V_B_1,

MCC_BIQUAD_V_B_2. This filter is enabled by default because the measured velocity is usually quite noisy. It can be disabled in the MCC_BIQUAD_V_ENABLE register.

The torque biquad filter is used to filter the input target value of the torque controller. This can be helpful when using the velocity or position control. In that case the output of the velocity controller is used as target torque. The exact location in the control loop can be seen in *Figure 6*. The filter coefficients can be set using these registers: MCC_BIQUAD_T_A_1, MCC_BIQUAD_T_A_2, MCC_BIQUAD_T_B_0, MCC_BIQUAD_T_B_1, MCC_BIQUAD_T_B_2. The torque filter can be enabled in the MCC_BIQUAD_T_ENABLE register.

PRBS Generator

A generator for a pseudorandom binary sequence (PRBS) is integrated into the MCC. The sequence can be configured using the MCC_PRBS_AMPLITUDE register. The output value of the generator is either the specified amplitude or the inverted negative amplitude, depending on the current random value. The binary sequence is reset when the MCC_PRBS_AMPLITUDE register is set to 0. This means it always generates the same PRBS values after a reset.

The MCC_PRBS_DOWN_SAMPLING_RATIO register can be used to configure the update rate of the PRBS generated value. By default, it is updated once every PWM cycle. The update rate is reduced according to the down sampling ratio.

The randomized output value can be used as target value for the different PI controllers. To enable this, the corresponding motion mode must be selected. The PRBS modes for flux, torque, velocity and position are available. They work identical to the normal motion modes, with the difference, that the target value for the corresponding PI controller is taken from the PRBS generator instead of the normal register for the target value. A special PRBS UD mode is also available. This mode bypasses the control loops and uses the PRBS value directly for the UD output voltage. The UQ voltage in this case is zero.

RAMP GENERATOR SETUP

The ramp generator allows motion based on target position or target velocity. It automatically calculates the motion profile taking into account the user's desired ramp profile. The EightPoint ramp generator uses three different acceleration settings each for the acceleration phase and for the deceleration phase to allow for jerk minimized ramps. Each acceleration setting corresponds to a velocity threshold setting.

The TMC9660 features a ramp generator that manipulates target position (RAMP_MODE = 0 → positioning mode) or target velocity (RAMP_MODE = 1 → velocity mode). It automatically calculates the motion profile taking into account acceleration and velocity settings. RAMP_EN = 1 activates the usage of ramper register fields for the position/velocity controller inputs instead of the target register fields. RAMP_MODE and RAMP_EN register fields are part of the MCC_MOTION_CONFIG register.

Additionally, the ramp generator handles stop events and allows on-the-fly changes in target position or target velocity and handles these according to the particular register values.

Changing the target position on-the-fly might lead to overshooting the new target when reducing the overall driving distance (RAMPER_X_TARGET – RAMPER_X_ACTUAL) or when reversing the driving direction.

The ramped values (RAMPER_X_ACTUAL, RAMPER_V_ACTUAL) can be used as target values for the velocity controller in MOTION_MODE velocity (PID_VELOCITY_TARGET ↔ RAMPER_V_ACTUAL) or MOTION_MODE position (PID_POSITION_TARGET ↔ RAMPER_X_ACTUAL). RAMP_MODE is independent from MOTION_MODE. Therefore, it is possible to mix RAMP_MODE position with MOTION_MODE velocity and vice versa. In addition, RAMPER_V_ACTUAL and RAMPER_A_ACTUAL can be used as additional feed forward values for the velocity controller and/or the torque controller. The following figure depicts the controller target value generation.

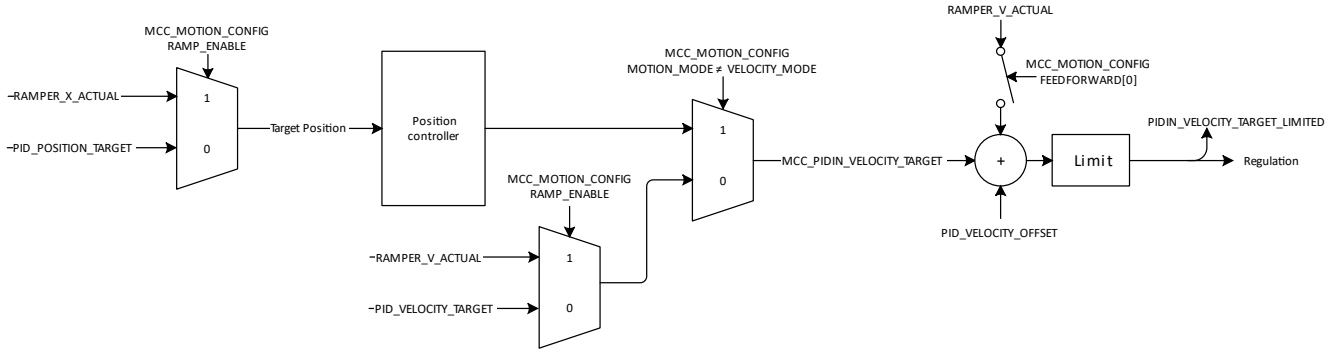

Figure 11. Controller target generation scheme

In the following both ramp modes and the corresponding registers are explained. Further information about PI controller register fields are clarified in the particular sections about current, velocity, and position control that have been explained before.

Real-World Unit Conversion

The TMC9660 uses its internal or external clock signal as a time reference for all internal operations. Thus, all time, velocity and acceleration settings are referenced to f_{CLK}. For best stability and reproducibility, it is recommended to use an external quartz oscillator as a time base, or to provide a clock signal from a microcontroller. The ramper velocity calculation depends on two other factors:

- Velocity meter selection: Register field METER_TYPE of MCC_VELOCITY_CONFIG register

- VELOCITY_PER: Measurement of velocity by time measurement between position changes. Recommended for slow velocity values.
- VELOCITY_FREQ: Velocity meter running at PWM frequency. Velocity value is calculated by the angle difference after one PWM cycle.
- VELOCITY_EXT: Velocity value is calculated externally and written to VELOCITY_EXT register field (MCC_VELOCITY_EXT register)

- Position step size: Depends on Feedback Engine settings

In the section *Velocity acquisition* information can be found when to use which velocity meter.

The following calculations give the ramper velocity in terms of steps/seconds, where the step size depends on the position feedback method. In the case of open loop, the position step size depends on the number of pole pairs of the motor.

The ramper acceleration is fully internal and only used in the ramp generator (as opposed to velocity, which can be controlled directly without the ramper). Of course, to get real units, the user also needs to relate the step size to reality using Feedback Engine calculation.

RAMPER_V… and RAMPER_A… register fields are internal units of the TMC9660. Those values need to be written to the velocity/acceleration registers fields explained in the following sections.

The following table gives a brief overview. Further more detailed information can be found in this manual in the particular sections; see the

Current scaling, *Velocity scaling*, and *Position scaling* section.

Table 11. Real-world conversion considerations

PARAMETER SYMBOL	UNIT	DESCRIPTION
f_{CLK}	[Hz]	40MHz default internal clock, usage of precise external clock recommended
step_size	[°]	Calculate the Step Size using Feedback Engine formulas
pwm_freq	[Hz]	PWM Frequency: $pwm\_freq = \frac{120MHz}{MCC\_PWM\_MAXCNT}$
n_pole_pairs		Number of motor pole pairs such that PHI_E = PHI_M × N_POLE_PAIRS
vel_scaling		Factor the user can set in the VELOCITY_SCALING register field (MCC_VELOCITY_SCALING register) to scale the velocity of the Velocity Meter (VM). Only relevant for METER_TYPE = VELOCITY_FREQ.
vel_sampling		Factor the user can set in the VELOCITY_SMPL register field to downsample velocity calculation in the Velocity Meter (VM). Only relevant for METER_TYPE = VELOCITY_FREQ.
svm_factor	[steps / s]	Velocity Meter scaling factor for METER_TYPE = VELOCITY_PER: 2^24 / f_{clk}, multiply by step_size to get [°/s].
vm_factor	[steps / s]	Velocity Meter scaling factor for METER_TYPE = VELOCITY_FREQ: $vm\_factor = vel\_scaling \times \frac{pwm\_freq}{vel\_sampling + 1}$

		Multiply by step_size to get [°/s].
openloop_factor	[rotations / s]	Velocity factor for open-loop rotation of the motor using the ramper internal PHI_E generation. Multiply by N_POLE_PAIRS / 360 to get [°/s]. rotation = one rotation of electrical field, i.e., PHI_E does 360° $2^{16} \times 2^{24} / f_{CLK}$
acceleration_factor		RAMPER_A… = RAMPER_V… × 2^17 / time / f_{CLK} where time is the ramp up time in [s] to get to the velocity, velocity is in [steps/s] and acceleration in [steps/s^2]. First set up the velocity meters and calculate velocity factors before determining the desired accelerations profiles.

RAMP_MODE: Positioning

During positioning mode, the ramp generator integrates RAMPER_V_ACTUAL to match RAMPER_X_ACTUAL with RAMPER_X_TARGET. The velocity ramp processing of RAMPER_V_ACTUAL is generated according to the defined ramp parameters. Up to seven velocity segments can be differentiated for these 8-point ramps:

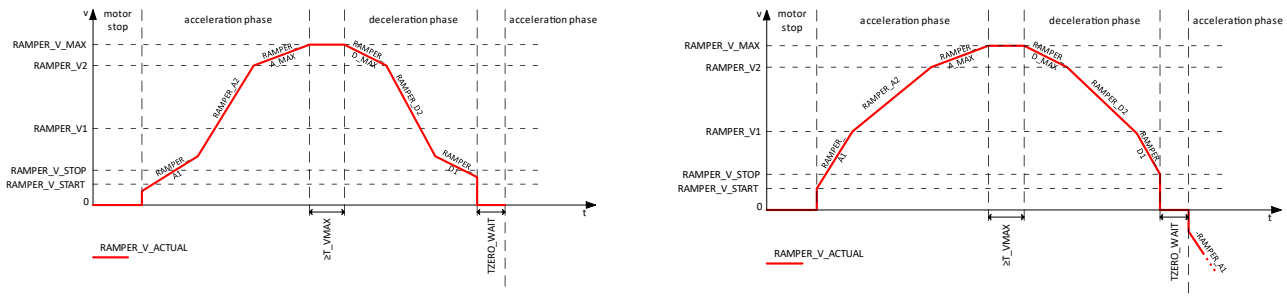

Figure 12. Typical positioning ramp structures

Following register fields are relevant:

Table 12. Ramper positioning mode parameters

REGISTER FIELD	REGISTER	SHORT EXPLANATION
RAMPER_X_ACTUAL (read only)	MCC_RAMPER_X_ACTUAL	Current position of the ramp generator. It used as a target for the position controller when in MOTION_MODE = position mode. This value is overwritten with PID_POSITION_ACTUAL value by a PID_POSITION_ACTUAL write access.
RAMPER_V_ACTUAL (read only)	MCC_RAMPER_V_ACTUAL	Current velocity of the ramp generator. It is used as a target for the velocity controller when in MOTION_MODE = velocity mode, or as feed forward to the velocity controller (Hint: feed forward is independent from the motion mode. Disable velocity feedforward when in MOTION_MODE = velocity mode)

RAMPER_A_ACTUAL (read only)	MCC_RAMPER_A_ACTUAL	Current acceleration used by the ramp generator. It can be fed forward to the torque controller.
RAMPER_X_TARGET	MCC_RAMPER_X_TARGET	Target Position of the Ramp generator. Driving distance is determined by difference to X_ACTUAL, as well as driving direction (sign (X_TARGET - X_ACTUAL)). This value is overwritten with PID_POSITION_ACTUAL value by a PID_POSITION_ACTUAL write access.
RAMPER_V_MAX	MCC_RAMPER_V_MAX	Maximum velocity (unsigned)
RAMPER_V_START	MCC_RAMPER_V_START	Initial velocity after motion start or change in driving direction (crossing velocity of 0)
RAMPER_V1	MCC_RAMPER_V1	Velocity value at which ramp generator switches between RAMPER_A1 and RAMPER_A2 during acceleration phase and between RAMPER_D1 and RAMPER_D2 during deceleration/soft-stop phase.
RAMPER_V2	MCC_RAMPER_V2	Velocity value at which ramp generator switches between RAMPER_A1 and RAMPER_A_MAX during acceleration phase and between RAMPER_D1 and RAMPER_D_MAX during deceleration/soft-stop phase.
RAMPER_V_STOP	MCC_RAMPER_V_STOP	Velocity value that used when reaching the target or before switching direction. In case |RAMPE_V_ACTUAL| is below RAMPER_V_STOP, RAMPER_V_ACTUAL is set to 0.
RAMPER_A1	MCC_RAMPER_A1	Acceleration value between V_START and V1
RAMPER_A2	MCC_RAMPER_A2	Acceleration value between V1 and V2
RAMPER_A_MAX	MCC_RAMPER_A_MAX	Acceleration value between V2 and V_MAX
RAMPER_D1	MCC_RAMPER_D1	Deceleration value between V_START and V1
RAMPER_D2	MCC_RAMPER_D2	Deceleration value between V1 and V2
RAMPER_D_MAX	MCC_RAMPER_D_MAX	Deceleration value between V2 and V_MAX
TZERO_WAIT	MCC_RAMPER_TIME_CONFIG	Forces velocity at 0 until timer runs out. Timer set after reaching target position, if RAMPER_V_ACTUAL is crossing 0 or after a hard stop/soft stop. It delays new motion by TZERO_WAIT × 12.8us.
T_VMAX	MCC_RAMPER_TIME_CONFIG	It forces a constant velocity value after reaching RAMPER_V_ACTUAL = RAMPER_V_MAX, before starting a new de-/acceleration phase, or when the sign of the acceleration RAMPER_A_ACTUAL would change. It delays any ac-/deceleration phase for T_VMAX × 12.8us.
SHIFT	MCC_RAMPER_ACC_FF	Shift factor for acceleration feed forward. See the next row.
GAIN	MCC_RAMPER_ACC_FF	Gain factor for acceleration feed forward: Result of (RAMPER_A_ACTUAL × GAIN) >> (SHIFT × 4) is used as offset for PIDIN_TORQUE_TARGET.

Early ramp termination happens whenever the steps needed for the constant velocity segment selected through T_VMAX and the steps needed for the deceleration ramp exceed the overall distance given by RAMPER_X_TARGET – RAMPER_X_ACTUAL.

This may lead to ramp where not all sections of the EightPoint ramps are executed. See the following examples:

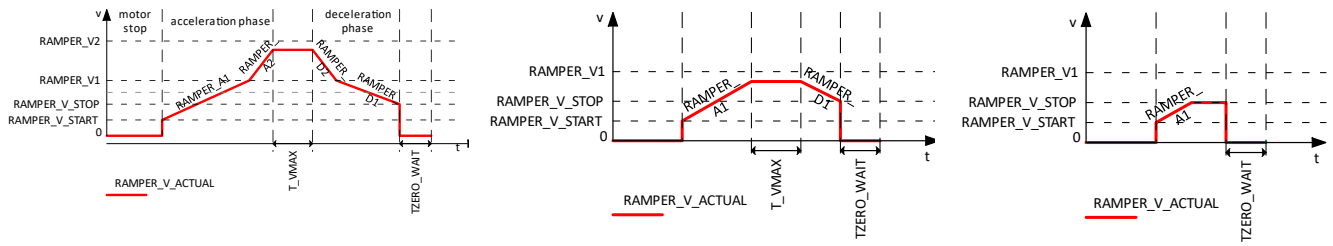

Figure 13. Adapted positioning ramp examples

On-the-fly changes of RAMPER_X_TARGET may lead to early ramp termination due to a reduced distance or an increase of velocity due to an increased distance and the possible execution of a T_VMAX segment. If the latter is not possible, an additional constant velocity phase is inserted. See the following figure:

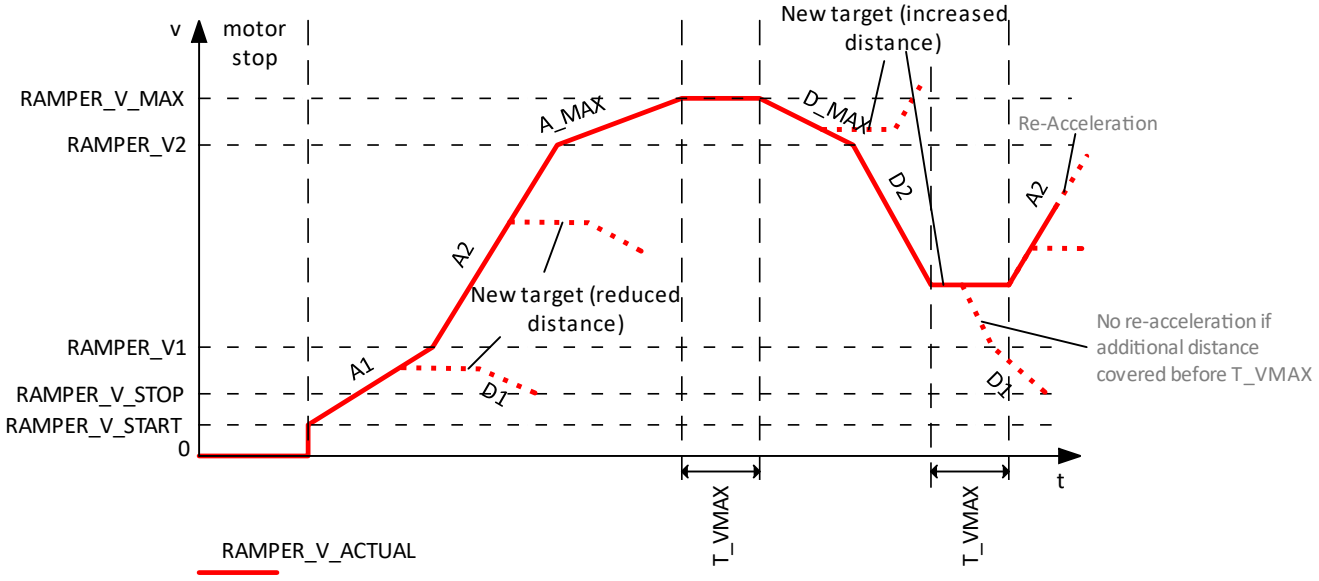

Figure 14. Possible on-the-fly changes adaptions during ramp positioning mode

RAMP_MODE: Velocity

In comparison to position mode, velocity mode utilizes only RAMPER_A1, RAMPER_A2, and RAMPER_A_MAX register fields for the ramp generation. RAMPER_D1, RAMPER_D2, and RAMPER_D_MAX are only used in case soft stop is active. RAMPER_V_START and RAMPER_V_STOP are not used.

Further on, RAMPER_V_TARGET defines the target velocity, RAMPER_V_MAX is not used in this mode. Thus, T_VMAX is used in case RAMPER_V_TARGET is reached.

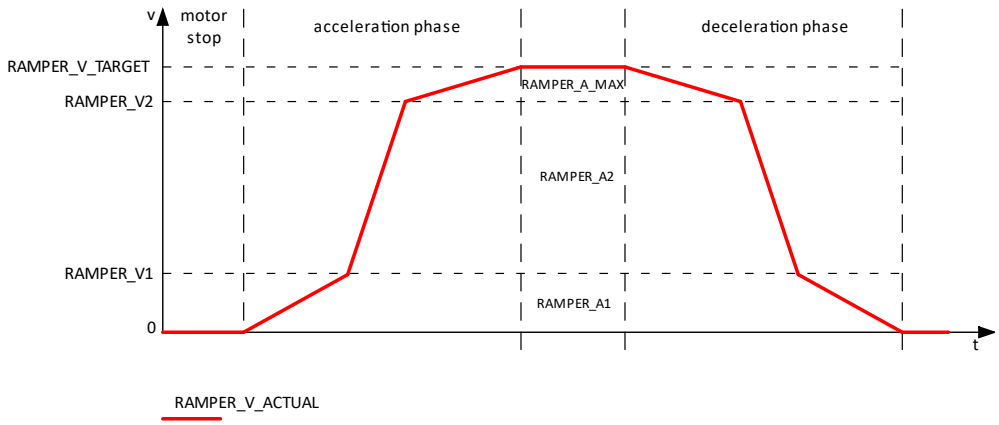

Figure 15. Velocity ramp phases

Following register fields are relevant:

Table 13. Ramp velocity mode parameters

REGISTER FIELD	REGISTER	SHORT EXPLANATION
RAMPER_X_ACTUAL (read only)	MCC_RAMPER_X_ACTUAL	Current position of the ramp generator. It used as a target for the position controller when in MOTION_MODE = position mode. This value is overwritten with PID_POSITION_ACTUAL value by a PID_POSITION_ACTUAL write access.
RAMPER_V_ACTUAL (read only)	MCC_RAMPER_V_ACTUAL	Current velocity of the ramp generator. It is used as a target for the velocity controller when in MOTION_MODE = velocity mode, or as feed forward to the velocity controller (Hint: feed forward is independent from the motion mode. Disable velocity feedforward when in MOTION_MODE = velocity mode)
RAMPER_A_ACTUAL (read only)	MCC_RAMPER_A_ACTUAL	Current acceleration used by the ramp generator. It can be fed forward to the torque controller.
RAMPER_V_TARGET	MCC_RAMPER_V_TARGET	Target velocity (signed). Sign sets the target direction.
RAMPER_V1	MCC_RAMPER_V1	Velocity value at which ramp generator switches between RAMPER_A1 andRAMPER_A2 during ac-/deceleration phase and between RAMPER_D1 and RAMPER_D2 during soft stop phase.
RAMPER_V2	MCC_RAMPER_V2	Velocity value at which ramp generator switches between RAMPER_A1 and RAMPER_A_MAX during ac-/deceleration phase and between RAMPER_D1 and RAMPER_D_MAX during soft stop phase.
RAMPER_A1	MCC_RAMPER_A1	Ac-/Deceleration value between 0 and V1
RAMPER_A2	MCC_RAMPER_A2	Ac-/Deceleration value between V1 and V2
RAMPER_A_MAX	MCC_RAMPER_A_MAX	Ac-/Deceleration value between V2 and V_MAX
RAMPER_D1	MCC_RAMPER_D1	Deceleration value between 0 and V1 (soft stop only)
RAMPER_D2	MCC_RAMPER_D2	Deceleration value between V1 and V2 (soft stop only)
RAMPER_D_MAX	MCC_RAMPER_D_MAX	Deceleration value between V2 and V_MAX (soft stop only)

TZERO_WAIT	MCC_RAMPER_TIME_CONFIG	Forces velocity at 0 until timer runs out. Timer is only set after hard stop/soft stop. It delays new motion by TZERO_WAIT × 12.8 us.
T_VMAX	MCC_RAMPER_TIME_CONFIG	It forces a constant velocity value after reaching RAMPER_V_ACTUAL = RAMPER_V_MAX, before starting a new de-/acceleration phase, or when the sign of the acceleration RAMPER_A_ACTUAL would change. It delays any ac-/deceleration phase for T_VMAX × 12.8 us.
SHIFT	MCC_RAMPER_ACC_FF	Shift factor for acceleration feed forward. See the next row.
GAIN	MCC_RAMPER_ACC_FF	Gain factor for acceleration feed forward: Result of (RAMPER_A_ACTUAL × GAIN) >> (SHIFT × 4) is used as offset for PIDIN_TORQUE_TARGET.

On-the-fly changes of RAMPER_V_TARGET may also lead to different T_VMAX phases:

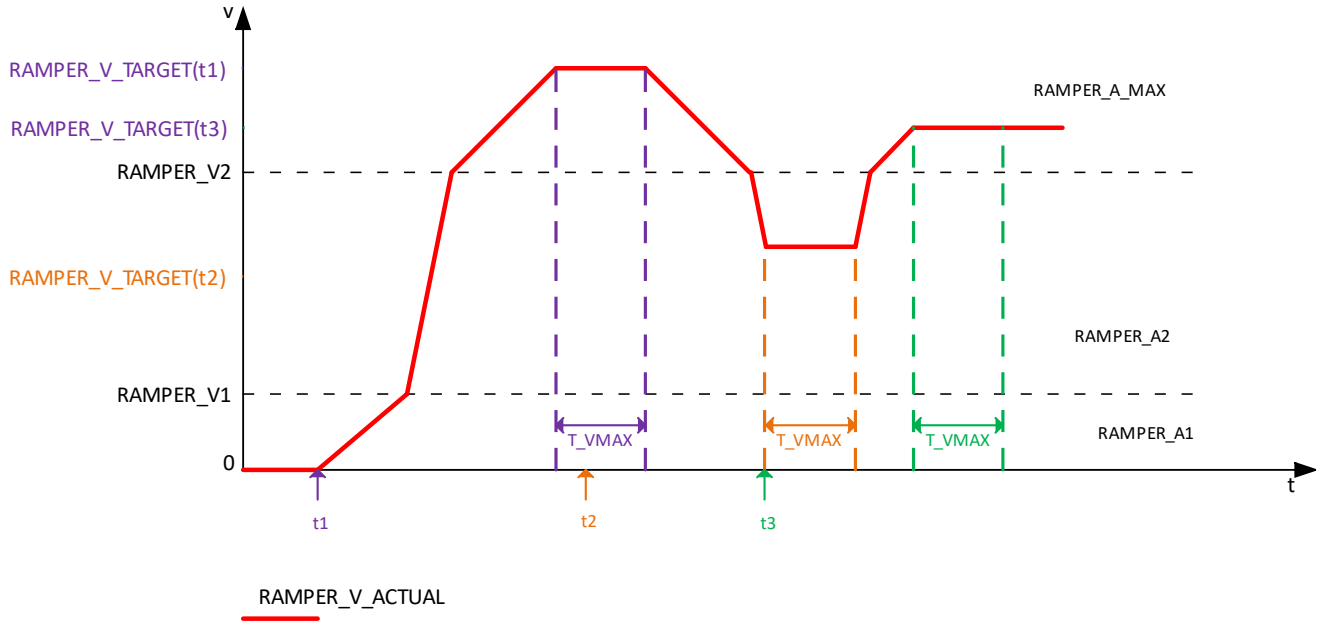

Figure 16. On-the-fly adaptations during ramp velocity mode

RAMPER_PHI_E generation

The TMC9660 provides different selectable sources of the rotor position for position measurement (see the SELECTION register field in the MCC_POSITION_CONFIG register). One option is the RAMPER_PHI_E value (MCC_RAMPER_PHI_E register) that is automatically calculated in the ramper:

RAMPER_PHI_E = RAMPER_X_ACTUAL × N_POLE_PAIRS + RAMPER_PHI_OFFSET

RAMPER_PHI_E_OFFSET can be adapted in the MCC_RAMPER_PHI_E_OFFSET register.

Stop and Reference Switch Implications

Two stop and one home reference switch are supported by the TMC9660. See the GPIO settings for the correct assignment. This reference can be used to latch certain positions and/or to stop the motion ramp. Settings for these features are assigned in the MCC_RAMPER_SWITCH_MODE:

Table 14. MCC_RAMP_SWITCH_MODE bit fields

STOP_R_POL, **STOP_L_POL**, and **STOP_H_POL** register fields
- These bit fields define the polarity of the right stop, left stop resp. home reference switches. - If set to 0, the active polarity of these switches is 1. - In case it is set to 1, the active polarities are inverted (=0).
STOP_R_ENABLE, **STOP_L_ENABLE**, and **STOP_H_ENABLE** register fields

- These enable bit fields trigger a stop event for the motion ramp in case (in correct order form)
  o Right reference switch is active and actual velocity value is positive: RAMPER_V_ACTUAL > 0
  o Left reference switch is active and actual velocity value is negative: RAMPER_V_ACTUAL < 0
  o Home reference switch is active

SG_STOP_ENABLE register field

- Enables the processing of the
  o SW_HARD_STOP,
  o STOP_ON_POS_DEVIATION, and STOP_ON_VEL_DEVIATION
- In case one of these stop events have been activated and triggered, ramp motion does not resume until this bit is cleared (by writing a 0 into this bit field)

SW_HARD_STOP register field
- Activating this bit automatically emerges a stop event (if SG_STOP_ENABLE == 1)
STOP_ON_POS_DEVIATION and **STOP_ON_VEL_DEVIATION** register fields

- Trigger a stop event in case
  o The absolute position error of the position controller exceeds MAX_POS_DEVIATION resp.
  o The absolute velocity error of the velocity controller exceeds MAX_VEL_DEVIATION
- SG_STOP_ENABLE has to be enabled

SOFTSTOP_ENABLE register field
- If this bit is active, no hard stop (RAMPER_V_ACTUAL is set immediately to v=0) is executed in case any stop event has triggered. Instead, a deceleration ramp (incl. all defined phases with RAMPER_D1, RAMPER_D2, and RAMPE_D_MAX) is executed if a stop event has triggered.
LATCH_R_ACTIVE, **LATCH_L_ACTIVE**, and **LATCH_H_ACTIVE** register fields
- If one of these bits (or combinations of them) are activated (set to 1), the value of RAMPER_X_ACTUAL is automatically written to **RAMPER_X_ACTUAL_LATCH** register field in case the particular reference switch has been activated. - Concurrently, the PID_POSITION_ACTUAL value is written to **POSITION_ACTUAL_LATCH** register field (actual feedback position). - Using this latching feature provides information about spurious stop events at mechanical reference switches.
LATCH_R_INACTIVE, **LATCH_L_INACTIVE**, and **LATCH_H_INACTIVE** register fields
- If one of these bits (or combinations of them) are activated (set to 1), the value of RAMPER_X_ACTUAL is automatically written to **RAMPER_X_ACTUAL_LATCH** register field in case the particular reference signals switches from active to inactive polarity level. - Concurrently, the PID_POSITION_ACTUAL value is written to **POSITION_ACTUAL_LATCH** register field.
SWAP_LR register field
- If active, internal processing of left and right reference switches are interchanged. - It might be useful to avoid external circuit changes.

Motion Control Status Flags

The ramp generator provides some status flags to indicate certain ramp conditions, incl. reference switch related flags. While some status flags define a certain ramp state (read-only), some flags mark an event that a certain ramp has occurred. Thus, the flag stays active as long as the state is active and the related bit is not cleared. All of the following status flags with the particular bit position are available in the MCC_RAMPER_STATUS register:

Table 15. MCC_RAMPER_STATUS bit fields

FLAG NAME (REGISTER FIELD)	BIT POSITION	ACCESS	DESCRIPTION
STATUS_STOP_L	0	R	Status of left reference switch: 0= inactive, 1= active
STATUS_STOP_R	1	R	Status of right reference switch: 0= inactive, 1= active
STATUS_STOP_H	2	R	Status of home reference switch: 0= inactive, 1= active
STATUS_LATCH_L	3	R/WC	Latching position has been executed by a triggered left reference switch. Write 1 to clear.
STATUS_LATCH_R	4	R/WC	Latching position has been executed by a triggered right reference switch. Write 1 to clear.
STATUS_LATCH_H	5	R/WC	Latching position has been executed by a triggered home reference switch. Write 1 to clear.
EVENT_STOP_L	6	R	Signals an active stop left condition based on an active left stop switch. *, **
EVENT_STOP_R	7	R	Signals an active stop right condition based on an active right stop switch. *, **
EVENT_STOP_H	8	R	Signals an active stop home condition based on an active home reference switch. *, **
EVENT_STOP_SG	9	R/WC	Signals an active stop event, incl. due to velocity and position tracking errors. Writing '1' into this bit field clears the stop condition and the motor may restart motion, unless the motion controller has been stopped. (Flag and interrupt condition is cleared upon writing '1'). **
EVENT_POS_REACHED	10	R/WC	Motion ramp has been stopped and target position has been reached (RAMPER_X_ACTUAL == RAMPER_X_TARGET). Write 1 to clear.
VELOCITY_REACHED	11	R	Maximum/target velocity is equal to actual velocity value. Positioning mode: RAMPER_V_ACTUAL == RAMPER_V_MAX Velocity mode: RAMPER_V_ACTUAL= = RAMPER_V_TARGET
POSITION_REACHED	12	R	Actual velocity is equal to 0 and target position is equal to actual position: RAMPER_X_ACTUAL == RAMPER_X_TARGET
V_ZERO	13	R	Actual velocity value RAMPER_V_ACTUAL is equal to 0.
T_ZEROWAIT_ACTIVE	14	R	T_ZERO_WAIT phase is active after ramp has been stopped. During this time, motor is in standstill.
SECOND_MOVE	15	R/WC	Ramp generator process new ramp that is moving back into opposite direction, e.g., due to on-the-fly target changes. Write 1 to clear.
STALL_IN_VEL_ERR	16	R	Ramp has been stopped because maximum velocity deviation (MAX_VEL_DEVIATION) is exceeded.
STALL_IN_POS_ERR	17	R	Ramp has been stopped because maximum position deviation (MAX_POS_DEVIATION) is exceeded.

Remarks:

* The stop condition and the interrupt condition can be removed by commanding a move to the opposite direction. In case soft_stop mode is active, the condition remains active until the motor has stopped motion into the direction of the stop switch. Disabling the stop switch or the stop function also clears the flag, but the motor continues its motion.

** This bit is ORed to the FAULTN output signal which means that the output signal is activated as soon as one of the assigned flags is activated.

STATUS FLAGS

Several status bits, flags, and events are available throughout the register map. Most of them have been explained in the sections of the particular feature and functions or got an additional section in which the flags have been described, e.g., see *ADC_STATUS Bits*, *Undervoltage Events* , *Motion Control Status Flags* or *Protection Features*.

Additionally, register MCC_STATUS_FLAGS holds different status flags for the motion control core. Each flag is set when the corresponding condition occurs. To clear a flag, the user must write 1 to the individual bit. The flags POSITION_TRACKING_ERROR and VELOCITY_TRACKING_ERROR can be configured using the registers MCC_MAX_POS_DEVIATION and MCC_MAX_VEL_DEVIATION accordingly. For more information about the individual status flags check out the corresponding register descriptions.

The FAULT_STS register hold multiple fault flags for the general status of the TMC9660. Each bit is set and cleared automatically depending on the corresponding condition. The register FAULT_R_INT holds the same status flags, but each bit is set when the condition occurs and must be cleared manually by writing 1 to the corresponding field. This way it is possible to check if the condition occurred at some point. The register FAULT_R_ENA_F holds the mask bits for the status flags. If any bit in the mask register is set and the corresponding flag in the FAULT_R_INT register is also set, the user can see the output at the FAULTN pin.

RAMDEBUG

The RamDebug feature is intended for debugging and monitoring the MCC core. It allows the collection of MCC register samples on up to four channels with a sampling rate of up to 25kHz and 4096 samples. The feature offers a trigger system similar to an oscilloscope, which allows triggering on events to capture them. The feature can be accessed using the command number 142 for UART communication, or the block number 31 for SPI communication – see the *Communication interfaces* section. The subcommands for RAMDebug are listed in *Table 18*.

Reset and Initialize RAMDebug

To configure a sample capture, the system first needs to be initialized by using the subcommand 0. After that, the channels can be configured.

Set Prescaler and Sample Count

To configure the sampling frequency and the number of samples, the prescaler and the sample count can be specified. The maximum number of samples available depends on the number of channels to be used. Note that the RAMDebug sample count must be set to the total amount of samples, not the number of samples per channel.

The sampling frequency is a value derived from the RAMDebug frequency.

$$f_{sampling} = \frac{f_{RAMDebug}}{value_{downsampling} + 1}$$

The RAMDebug frequency is derived from the PWM frequency. The RAMDebug frequency is limited to 25kHz. If the PWM frequency is set to a frequency higher than 25kHz, RAMDebug automatically prescales the PWM frequency to stay at or below 25kHz.

Table 16. PWM frequency vs. RAMDebug frequency

PWM FREQUENCY	RAMDEBUG FREQUENCY
$100kHz \geq f_{PWM} > 75kHz$	$f_{RAMDebug} = f_{PWM}/4$
$75kHz \geq f_{PWM} > 50kHz$	$f_{RAMDebug} = f_{PWM}/3$
$25kHz \geq f_{PWM} > 25kHz$	$f_{RAMDebug} = f_{PWM}/2$
$25kHz \geq f_{PWM}$	$f_{RAMDebug} = f_{PWM}$

Set Number of Pretrigger Samples

Pretrigger samples allow to observe the channel states slightly before the trigger event. The number of pretrigger samples that is desired can be specified using the subcommand 13. Just like the sample count, the number of pretrigger samples specified is the total amount of samples, not samples per channel.

Set Up the Channels

To set a channel, the RAMDebug state must be idle. To set a channel, the subcommand 4 is used. The index value specifies the channel type, and the command value specifies the channel address.

The channel type to capture a register is 2, and the channel address is the register address in the MCC block.

The channel number is defined by the order in which the channels are written.

Set a Trigger and Start the Measurement

The configuration of the trigger channel is identical to a capture channel. Additionally, a mask and shift value can be specified by subcommand 6. The mask and shift are applied to the channel value before checking the trigger. The trigger criterion is specified using type code 7, with the index specifying the trigger type and the value specifying the trigger threshold. Type code 7 starts the measurement.

For the trigger edge detection, the trigger threshold is compared to the captured trigger channel value after applying the mask and shift. If the trigger threshold and the trigger value are equal, it is considered below the threshold. For example, a trigger threshold of 10 detects a rising edge if the value transitions from 10 or less to 11 or more, but a transition from less than 10 to exactly 10 does not trigger a rising edge.

Get Status

To get the RAMDebug status, subcommand 10 is used. This allows us to check if the system is ready to be triggered, has already triggered, or if the capture is done. For a full list of status codes see the following:

Table 17. List of RAMDebug states

NUMBER	NAME	DESCRIPTION
0	Idle	RAMDebug is not running and can be configured. Use subcommand 0 to enter this state.
1	Trigger	RAMDebug is waiting for the trigger event to happen. When updating a value that RAMDebug is triggering on, ensure this state is reached before updating. Once the trigger event occurs, RAMDebug transitions to the Capture state.
2	Capture	RAMDebug has been triggered and is capturing samples. Once all samples are collected, RAMDebug transitions to the Complete state.
3	Complete	RAMDebug has finished capturing samples. The data can now be downloaded using subcommand 9.
4	Pretrigger	RAMDebug is capturing pretrigger samples. Once enough pretrigger samples are collected, RAMDebug transitions to the Trigger state.

Get Samples

As soon as the sampling is done (RAMDebug state Complete), samples can be downloaded. The sampled values are requested using subcommand 9. The index value specifies the requested sample. The samples are ordered by acquisition time and channel. For example, for a two-channel capture, index zero would return the first element of channel zero, index 1 the first element of channel one, index 2 the second value of channel zero, etc.

Table 18. List of RAMDebug subcommands

NUMBER	NAME	INDEX	VALUE	DESCRIPTION
0	Initialize and reset	-	-	Initialize and reset RAMDebug.
1	Set sample count	-	Sample count	Sets the number of samples to collect in total (not per channel)
3	Set prescaler	-	Prescale value	Sets divider for sampling rate. Base frequency is divided by value + 1.
4	Set channel	Type: 0: Disabled 2: MCC register	MCC register address	Set what channel to capture. A maximum of 4 channels are available.
5	Set trigger channel	Type: 0: Disabled 2: MCC register	MCC register address	Set the trigger channel.
6	Set trigger mask shift	Shift value	Mask value	Set the mask and shift value to be applied to the trigger value.
7	Set trigger and start measurement	Type: 0: No trigger 1: Rising edge signed 2: Falling edge signed 3: Any edge signed 4: Rising edge signed 5: Falling edge signed 6: Any edge signed	Trigger threshold	Configure the trigger and start the measurement.
8	Get state	-	-	Read out the state of RAMDebug. 0: Idle 1: Trigger 2: Capture 3: Complete 4: Pretrigger See *Table 17*
9	Read sample	Sample number	-	Read out a captured sample
10	Get info	-	Info selection	Read out general RAMDebug information: 0: Maximum number of channels 1: Maximum amount of samples 2: RAMDebug frequency in Hz 3: Number of Samples already captured 4: RAMDebug prescaler value on trigger event
11	Get channel type	Channel number	-	Read out the configured channel type.
12	Get channel address	Channel number	-	Read out the configured channel address.
13	Set pretrigger sample count	-	Number of samples	Set the total number of pretrigger samples (not per channel).
14	Get pretrigger sample count	-	-	Read out the total number of pretrigger samples.

REGISTER MAP

The following pages show all the accessible registers of the TMC9660. Note that they are divided into different blocks that need to be addressed as described in the Communication Interfaces section.

MCC Register Overview

Shows all related registers of the MCC block.

ADDRESS & NAME	FIELDS	MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x000 **MCC_INFO_CHIP**	ID[31:24]	
	ID[23:16]	
	ID[15:8]	
	ID[7:0]	
0x020 **MCC_ADC_I1_I0_RAW**	I1[15:8]	
	I1[7:0]	
	I0[15:8]	
	I0[7:0]	
0x021 **MCC_ADC_I3_I2_RAW**	I3[15:8]	
	I3[7:0]	
	I2[15:8]	
	I2[7:0]	
0x022 **MCC_ADC_U1_U0_ RAW**	U1[15:8]	
	U1[7:0]	
	U0[15:8]	
	U0[7:0]	
0x023 **MCC_ADC_U3_U2_ RAW**	U3[15:8]	
	U3[7:0]	
	U2[15:8]	
	U2[7:0]	
0x024 **MCC_ADC_TEMP_VM_ RAW**	TEMP[15:8]	
	TEMP[7:0]	
	VM[15:8]	
	VM[7:0]	

ADDRESS & NAME	FIELDS	MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)							
0x025 **MCC_ADC_AIN1_AIN0_RAW**	AIN1[15:8]								
	AIN1[7:0]								
	AIN0[15:8]								
	AIN0[7:0]								
0x026 **MCC_ADC_AIN3_AIN2_RAW**	AIN3[15:8]								
	AIN3[7:0]								
	AIN2[15:8]								
	AIN2[7:0]								
0x040 **MCC_ADC_I_GEN_CONFIG**	TRIGGER_POS[15:8]								
	TRIGGER_POS[7:0]								
				TRIGGER_SELECT	MEASUREMENT_MODE				
	Y2_SELECT		WY1_SELECT		VX2_SELECT		UX1_SELECT		
0x041 **MCC_ADC_I0_CONFIG**	SCALE[15:8]								
	SCALE[7:0]								
	OFFSET[15:8]								
	OFFSET[7:0]								
0x042 **MCC_ADC_I1_CONFIG**	SCALE[15:8]								
	SCALE[7:0]								
	OFFSET[15:8]								
	OFFSET[7:0]								
0x043 **MCC_ADC_I2_CONFIG**	SCALE[15:8]								
	SCALE[7:0]								
	OFFSET[15:8]								
	OFFSET[7:0]								
0x044 **MCC_ADC_I3_CONFIG**	SCALE[15:8]								
	SCALE[7:0]								
	OFFSET[15:8]								
	OFFSET[7:0]								

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x045 **MCC_ADC_I1_I0_ SCALED**	I1[15:8]							
	I1[7:0]							
	I0[15:8]							
	I0[7:0]							
0x046 **MCC_ADC_I3_I2_ SCALED**	I3[15:8]							
	I3[7:0]							
	I2[15:8]							
	I2[7:0]							
0x047 **MCC_ADC_IWY_IUX**	IWY[15:8]							
	IWY[7:0]							
	IUX[15:8]							
	IUX[7:0]							
0x048 **MCC_ADC_IV**								
	IV[15:8]							
	IV[7:0]							
0x049 **MCC_ADC_STATUS**			TEMP_ DONE	VM_DONE	AIN3_DONE	AIN2_DONE	AIN1_DONE	AIN0_DONE
	U3_DONE	U2_DONE	U1_DONE	U0_DONE	I3_DONE	I2_DONE	I1_DONE	I0_DONE
			TEMP_ CLIPPED	VM_ CLIPPED	AIN3_ CLIPPED	AIN2_ CLIPPED	AIN1_ CLIPPED	AIN0_ CLIPPED
	U3_ CLIPPED	U2_ CLIPPED	U1_ CLIPPED	U0_ CLIPPED	I3_CLIPPED	I2_CLIPPED	I1_CLIPPED	I0_CLIPPED
0x060 **MCC_MOTOR_CONFIG**								
							TYPE	
	N_POLE_PAIRS							
0x061 **MCC_MOTION_CONFIG**								
	FEEDFORWARD		RAMP_ MODE	RAMP_ ENABLE	MOTION_MODE			

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x062 **MCC_PHI_E_SELECTION**								
					PHI_E_SELECTION			
0x063 **MCC_PHI_E**								
	PHI_E[15:8]							
	PHI_E[7:0]							
0x080 **MCC_PWM_CONFIG**	DUTY_CYCLE_OFFSET[15:8]							
	DUTY_CYCLE_OFFSET[7:0]							
	EXT_ENABLE_Y2	EXT_ENABLE_WY1	EXT_ENABLE_VX2	EXT_ENABLE_UX1	ENABLE_Y2	ENABLE_WY1	ENABLE_VX2	ENABLE_UX1
	Y2_HS_SRC		SV_MODE			CHOP		
0x081 **MCC_PWM_MAXCNT**								
	PWM_MAXCNT[15:8]							
	PWM_MAXCNT[7:0]							
0x083 **MCC_PWM_SWITCH_LIMIT**								
	PWM_SWITCH_LIMIT[15:8]							
	PWM_SWITCH_LIMIT[7:0]							
0x0A0 **MCC_ABN_PHI_E_PHI_M**	PHI_E[15:8]							
	PHI_E[7:0]							
	PHI_M[15:8]							
	PHI_M[7:0]							
0x0A1 **MCC_ABN_MODE**								
				DIRECTION				CLN
			DISABLE_FILTER	CLEAR_COUNT_ON_N	COMBINED_N	N_POL	B_POL	A_POL

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x0A2 **MCC_ABN_CPR**								
	ABN_CPR[23:16]							
	ABN_CPR[15:8]							
	ABN_CPR[7:0]							
0x0A3 **MCC_ABN_CPR_INV**	ABN_CPR_INV[31:24]							
	ABN_CPR_INV[23:16]							
	ABN_CPR_INV[15:8]							
	ABN_CPR_INV[7:0]							
0x0A4 **MCC_ABN_COUNT**								
	ABN_COUNT[23:16]							
	ABN_COUNT[15:8]							
	ABN_COUNT[7:0]							
0x0A5 **MCC_ABN_COUNT_N**								
	ABN_COUNT_N[23:16]							
	ABN_COUNT_N[15:8]							
	ABN_COUNT_N[7:0]							
0x0A6 **MCC_ABN_PHI_E_ OFFSET**								
	ABN_PHI_E_OFFSET[15:8]							
	ABN_PHI_E_OFFSET[7:0]							
0x0C0 **MCC_HALL_MODE**								
	FILTER							
		ORDER					EXTRAPOL ATION	POLARITY
0x0C1 **MCC_HALL_DPHI_MAX**								
	HALL_DPHI_MAX[15:8]							
	HALL_DPHI_MAX[7:0]							

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x0C2 **MCC_HALL_PHI_E_OFFSET**								
	HALL_PHI_E_OFFSET[15:8]							
	HALL_PHI_E_OFFSET[7:0]							
0x0C3 **MCC_HALL_COUNT**								
	HALL_COUNT[15:8]							
	HALL_COUNT[7:0]							
0x0C4 **MCC_HALL_PHI_E_EXTRAPOLATED_PHI_E**	PHI_E_EXTRAPOLATED[15:8]							
	PHI_E_EXTRAPOLATED[7:0]							
	PHI_E[15:8]							
	PHI_E[7:0]							
0x0C5 **MCC_HALL_POSITION_060_POSITION_000**	POSITION_060[15:8]							
	POSITION_060[7:0]							
	POSITION_000[15:8]							
	POSITION_000[7:0]							
0x0C6 **MCC_HALL_POSITION_180_POSITION_120**	POSITION_180[15:8]							
	POSITION_180[7:0]							
	POSITION_120[15:8]							
	POSITION_120[7:0]							
0x0C7 **MCC_HALL_POSITION_300_POSITION_240**	POSITION_300[15:8]							
	POSITION_300[7:0]							
	POSITION_240[15:8]							
	POSITION_240[7:0]							
0x0E0 **MCC_BIQUAD_V_A_1**								
	BIQUAD_V_A_1[23:16]							
	BIQUAD_V_A_1[15:8]							
	BIQUAD_V_A_1[7:0]							
0x0E1 **MCC_BIQUAD_V_A_2**								
	BIQUAD_V_A_2[23:16]							
	BIQUAD_V_A_2[15:8]							
	BIQUAD_V_A_2[7:0]							

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x0E2 **MCC_BIQUAD_V_B_0**								
	BIQUAD_V_B_0[23:16]							
	BIQUAD_V_B_0[15:8]							
	BIQUAD_V_B_0[7:0]							
0x0E3 **MCC_BIQUAD_V_B_1**								
	BIQUAD_V_B_1[23:16]							
	BIQUAD_V_B_1[15:8]							
	BIQUAD_V_B_1[7:0]							
0x0E4 **MCC_BIQUAD_V_B_2**								
	BIQUAD_V_B_2[23:16]							
	BIQUAD_V_B_2[15:8]							
	BIQUAD_V_B_2[7:0]							
0x0E5 **MCC_BIQUAD_V_ ENABLE**								
								BIQUAD_V_ ENABLE
0x0E6 **MCC_BIQUAD_T_A_1**								
	BIQUAD_T_A_1[23:16]							
	BIQUAD_T_A_1[15:8]							
	BIQUAD_T_A_1[7:0]							
0x0E7 **MCC_BIQUAD_T_A_2**								
	BIQUAD_T_A_2[23:16]							
	BIQUAD_T_A_2[15:8]							
	BIQUAD_T_A_2[7:0]							
0x0E8 **MCC_BIQUAD_T_B_0**								
	BIQUAD_T_B_0[23:16]							
	BIQUAD_T_B_0[15:8]							
	BIQUAD_T_B_0[7:0]							

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x0E9 **MCC_BIQUAD_T_B_1**								
	BIQUAD_T_B_1[23:16]							
	BIQUAD_T_B_1[15:8]							
	BIQUAD_T_B_1[7:0]							
0x0EA **MCC_BIQUAD_T_B_2**								
	BIQUAD_T_B_2[23:16]							
	BIQUAD_T_B_2[15:8]							
	BIQUAD_T_B_2[7:0]							
0x0EB **MCC_BIQUAD_T_ ENABLE**								
								BIQUAD_T_ ENABLE
0x100 **MCC_VELOCITY_ CONFIG**								
		MOVING_AVRG_FILTER_SAMPLES				METER_TYPE		METER_ SYNC_ PULSE
	SELECTION							
0x101 **MCC_VELOCITY_ SCALING**								
	VELOCITY_SCALING[15:8]							
	VELOCITY_SCALING[7:0]							
0x102 **MCC_V_MIN_POS_ DEV_TIME_COUNTER_ LIMIT**		V_MIN_POS_DEV[14:8]						
	V_MIN_POS_DEV[7:0]							
	TIME_COUNTER_LIMIT[15:8]							
	TIME_COUNTER_LIMIT[7:0]							
0x103 **MCC_MAX_VEL_ DEVIATION**		MAX_VEL_DEVIATION[30:24]						
	MAX_VEL_DEVIATION[23:16]							
	MAX_VEL_DEVIATION[15:8]							
	MAX_VEL_DEVIATION[7:0]							

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x120 **MCC_POSITION_ CONFIG**								
	SELECTION							
0x121 **MCC_MAX_POS_ DEVIATION**		MAX_POS_DEVIATION[30:24]						
	MAX_POS_DEVIATION[23:16]							
	MAX_POS_DEVIATION[15:8]							
	MAX_POS_DEVIATION[7:0]							
0x140 **MCC_RAMPER_ STATUS**								
							STALL_IN_ POS_ERR	STALL_IN_ VEL_ERR
	SECOND_ MOVE	T_ ZEROWAIT _ACTIVE	V_ZERO	POSITION_ REACHED	VELOCITY_ REACHED	EVENT_ POS_ REACHED	EVENT_ STOP_SG	EVENT_ STOP_H
	EVENT_ STOP_R	EVENT_ STOP_L	STATUS_ LATCH_H	STATUS_ LATCH_R	STATUS_ LATCH_L	STATUS_ STOP_H	STATUS_ STOP_R	STATUS_ STOP_L
0x141 **MCC_RAMPER_A1**								
		RAMPER_A1[22:16]						
	RAMPER_A1[15:8]							
	RAMPER_A1[7:0]							
0x142 **MCC_RAMPER_A2**								
		RAMPER_A2[22:16]						
	RAMPER_A2[15:8]							
	RAMPER_A2[7:0]							
0x143 **MCC_RAMPER_A_MAX**								
		RAMPER_A_MAX[22:16]						
	RAMPER_A_MAX[15:8]							
	RAMPER_A_MAX[7:0]							
0x144 **MCC_RAMPER_D1**								
		RAMPER_D1[22:16]						
	RAMPER_D1[15:8]							
	RAMPER_D1[7:0]							

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x145 **MCC_RAMPER_D2**								
	RAMPER_D2[22:16]							
	RAMPER_D2[15:8]							
	RAMPER_D2[7:0]							
0x146 **MCC_RAMPER_D_MAX**								
	RAMPER_D_MAX[22:16]							
	RAMPER_D_MAX[15:8]							
	RAMPER_D_MAX[7:0]							
0x147 **MCC_RAMPER_V_START**								
	RAMPER_V_START[22:16]							
	RAMPER_V_START[15:8]							
	RAMPER_V_START[7:0]							
0x148 **MCC_RAMPER_V1**						RAMPER_V1[26:24]		
	RAMPER_V1[23:16]							
	RAMPER_V1[15:8]							
	RAMPER_V1[7:0]							
0x149 **MCC_RAMPER_V2**						RAMPER_V2[26:24]		
	RAMPER_V2[23:16]							
	RAMPER_V2[15:8]							
	RAMPER_V2[7:0]							
0x14A **MCC_RAMPER_V_STOP**								
	RAMPER_V_STOP[22:16]							
	RAMPER_V_STOP[15:8]							
	RAMPER_V_STOP[7:0]							
0x14B **MCC_RAMPER_V_MAX**						RAMPER_V_MAX[26:24]		
	RAMPER_V_MAX[23:16]							
	RAMPER_V_MAX[15:8]							
	RAMPER_V_MAX[7:0]							
0x14C **MCC_RAMPER_V_TARGET**					RAMPER_V_TARGET[27:24]			
	RAMPER_V_TARGET[23:16]							
	RAMPER_V_TARGET[15:8]							
	RAMPER_V_TARGET[7:0]							

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x14D **MCC_RAMPER_SWITCH_MODE**								
					VELOCITY_OVERWRITE	STOP_ON_VEL_DEVIATION	STOP_ON_POS_DEVIATION	SW_HARD_STOP
	SOFTSTOP_ENABLE	SG_STOP_ENABLE		LATCH_H_INACTIVE	LATCH_H_ACTIVE	LATCH_R_INACTIVE	LATCH_R_ACTIVE	LATCH_L_INACTIVE
	LATCH_L_ACTIVE	SWAP_LR	STOP_H_POL	STOP_R_POL	STOP_L_POL	STOP_H_ENABLE	STOP_R_ENABLE	STOP_L_ENABLE
0x14E **MCC_RAMPER_TIME_CONFIG**	T_VMAX[15:8]							
	T_VMAX[7:0]							
	T_ZEROWAIT[15:8]							
	T_ZEROWAIT[7:0]							
0x14F **MCC_RAMPER_A_ACTUAL**								
	RAMPER_A_ACTUAL[23:16]							
	RAMPER_A_ACTUAL[15:8]							
	RAMPER_A_ACTUAL[7:0]							
0x150 **MCC_RAMPER_X_ACTUAL**	RAMPER_X_ACTUAL[31:24]							
	RAMPER_X_ACTUAL[23:16]							
	RAMPER_X_ACTUAL[15:8]							
	RAMPER_X_ACTUAL[7:0]							
0x151 **MCC_RAMPER_V_ACTUAL**					RAMPER_V_ACTUAL[27:24]			
	RAMPER_V_ACTUAL[23:16]							
	RAMPER_V_ACTUAL[15:8]							
	RAMPER_V_ACTUAL[7:0]							
0x152 **MCC_RAMPER_X_TARGET**	RAMPER_X_TARGET[31:24]							
	RAMPER_X_TARGET[23:16]							
	RAMPER_X_TARGET[15:8]							
	RAMPER_X_TARGET[7:0]							
0x153 **MCC_RAMPER_PHI_E**								
	RAMPER_PHI_E[15:8]							
	RAMPER_PHI_E[7:0]							

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x154 **MCC_RAMPER_PHI_E_OFFSET**								
	RAMPER_PHI_E_OFFSET[15:8]							
	RAMPER_PHI_E_OFFSET[7:0]							
0x155 **MCC_RAMPER_ACC_FF**								
						SHIFT		
	GAIN[15:8]							
	GAIN[7:0]							
0x156 **MCC_RAMPER_X_ACTUAL_LATCH**	RAMPER_X_ACTUAL_LATCH[31:24]							
	RAMPER_X_ACTUAL_LATCH[23:16]							
	RAMPER_X_ACTUAL_LATCH[15:8]							
	RAMPER_X_ACTUAL_LATCH[7:0]							
0x157 **MCC_POSITION_ACTUAL_LATCH**	POSITION_ACTUAL_LATCH[31:24]							
	POSITION_ACTUAL_LATCH[23:16]							
	POSITION_ACTUAL_LATCH[15:8]							
	POSITION_ACTUAL_LATCH[7:0]							
0x160 **MCC_PRBS_AMPLITUDE**	PRBS_AMPLITUDE[31:24]							
	PRBS_AMPLITUDE[23:16]							
	PRBS_AMPLITUDE[15:8]							
	PRBS_AMPLITUDE[7:0]							
0x161 **MCC_PRBS_DOWN_SAMPLING_RATIO**								
	PRBS_DOWN_SAMPLING_RATIO							
0x180 **MCC_PID_CONFIG**		VEL_SMPL						
		POS_SMPL						
	VEL_SCALE				POSITION_NORM_I		POSITION_NORM_P	
	VELOCITY_NORM_I		VELOCITY_NORM_P		CURRENT_NORM_I	CURRENT_NORM_P		KEEP_POS_TARGET

ADDRESS & NAME	FIELDS	MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x181 **MCC_PID_FLUX_ COEFF**	P[15:8]	
	P[7:0]	
	I[15:8]	
	I[7:0]	
0x182 **MCC_PID_TORQUE_ COEFF**	P[15:8]	
	P[7:0]	
	I[15:8]	
	I[7:0]	
0x183 **MCC_PID_ FIELDWEAK_COEFF**	P[15:8]	
	P[7:0]	
	I[15:8]	
	I[7:0]	
0x184 **MCC_PID_U_S_MAX**		
	U_S_MAX[15:8]	
	U_S_MAX[7:0]	
0x185 **MCC_PID_VELOCITY_ COEFF**	P[15:8]	
	P[7:0]	
	I[15:8]	
	I[7:0]	
0x186 **MCC_PID_POSITION_ COEFF**	P[15:8]	
	P[7:0]	
	I[15:8]	
	I[7:0]	
0x187 **MCC_PID_POSITION_ TOLERANCE**		PID_POSITION_TOLERANCE[30:24]
	PID_POSITION_TOLERANCE[23:16]	
	PID_POSITION_TOLERANCE[15:8]	
	PID_POSITION_TOLERANCE[7:0]	
0x188 **MCC_PID_POSITION_ TOLERANCE_DELAY**		
	PID_POSITION_TOLERANCE_DELAY[15:8]	
	PID_POSITION_TOLERANCE_DELAY[7:0]	

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x189 **MCC_PID_UQ_UD_LIMITS**								
	PID_UQ_UD_LIMITS[15:8]							
	PID_UQ_UD_LIMITS[7:0]							
0x18A **MCC_PID_TORQUE_FLUX_LIMITS**		PID_TORQUE_LIMIT[14:8]						
	PID_TORQUE_LIMIT[7:0]							
		PID_FLUX_LIMIT[14:8]						
	PID_FLUX_LIMIT[7:0]							
0x18B **MCC_PID_VELOCITY_LIMIT**		PID_VELOCITY_LIMIT[30:24]						
	PID_VELOCITY_LIMIT[23:16]							
	PID_VELOCITY_LIMIT[15:8]							
	PID_VELOCITY_LIMIT[7:0]							
0x18C **MCC_PID_POSITION_LIMIT_LOW**	PID_POSITION_LIMIT_LOW[31:24]							
	PID_POSITION_LIMIT_LOW[23:16]							
	PID_POSITION_LIMIT_LOW[15:8]							
	PID_POSITION_LIMIT_LOW[7:0]							
0x18D **MCC_PID_POSITION_LIMIT_HIGH**	PID_POSITION_LIMIT_HIGH[31:24]							
	PID_POSITION_LIMIT_HIGH[23:16]							
	PID_POSITION_LIMIT_HIGH[15:8]							
	PID_POSITION_LIMIT_HIGH[7:0]							
0x18E **MCC_PID_TORQUE_FLUX_TARGET**	PID_TORQUE_TARGET[15:8]							
	PID_TORQUE_TARGET[7:0]							
	PID_FLUX_TARGET[15:8]							
	PID_FLUX_TARGET[7:0]							
0x18F **MCC_PID_TORQUE_FLUX_OFFSET**	PID_TORQUE_OFFSET[15:8]							
	PID_TORQUE_OFFSET[7:0]							
	PID_FLUX_OFFSET[15:8]							
	PID_FLUX_OFFSET[7:0]							
0x190 **MCC_PID_VELOCITY_TARGET**	PID_VELOCITY_TARGET[31:24]							
	PID_VELOCITY_TARGET[23:16]							
	PID_VELOCITY_TARGET[15:8]							
	PID_VELOCITY_TARGET[7:0]							

ADDRESS & NAME	FIELDS	MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x191 **MCC_PID_VELOCITY_OFFSET**	PID_VELOCITY_OFFSET[31:24]	
	PID_VELOCITY_OFFSET[23:16]	
	PID_VELOCITY_OFFSET[15:8]	
	PID_VELOCITY_OFFSET[7:0]	
0x192 **MCC_PID_POSITION_TARGET**	PID_POSITION_TARGET[31:24]	
	PID_POSITION_TARGET[23:16]	
	PID_POSITION_TARGET[15:8]	
	PID_POSITION_TARGET[7:0]	
0x193 **MCC_PID_TORQUE_FLUX_ACTUAL**	PID_TORQUE_ACTUAL[15:8]	
	PID_TORQUE_ACTUAL[7:0]	
	PID_FLUX_ACTUAL[15:8]	
	PID_FLUX_ACTUAL[7:0]	
0x194 **MCC_PID_VELOCITY_ACTUAL**	PID_VELOCITY_ACTUAL[31:24]	
	PID_VELOCITY_ACTUAL[23:16]	
	PID_VELOCITY_ACTUAL[15:8]	
	PID_VELOCITY_ACTUAL[7:0]	
0x195 **MCC_PID_POSITION_ACTUAL**	PID_POSITION_ACTUAL[31:24]	
	PID_POSITION_ACTUAL[23:16]	
	PID_POSITION_ACTUAL[15:8]	
	PID_POSITION_ACTUAL[7:0]	
0x196 **MCC_PID_POSITION_ACTUAL_OFFSET**	PID_POSITION_ACTUAL_OFFSET[31:24]	
	PID_POSITION_ACTUAL_OFFSET[23:16]	
	PID_POSITION_ACTUAL_OFFSET[15:8]	
	PID_POSITION_ACTUAL_OFFSET[7:0]	
0x197 **MCC_PID_TORQUE_ERROR**		
	PID_TORQUE_ERROR[15:8]	
	PID_TORQUE_ERROR[7:0]	
0x198 **MCC_PID_FLUX_ERROR**		
	PID_FLUX_ERROR[15:8]	
	PID_FLUX_ERROR[7:0]	

ADDRESS & NAME	FIELDS	MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x199 **MCC_PID_VELOCITY_ERROR**	PID_VELOCITY_ERROR[31:24]	
	PID_VELOCITY_ERROR[23:16]	
	PID_VELOCITY_ERROR[15:8]	
	PID_VELOCITY_ERROR[7:0]	
0x19A **MCC_PID_POSITION_ERROR**	PID_POSITION_ERROR[31:24]	
	PID_POSITION_ERROR[23:16]	
	PID_POSITION_ERROR[15:8]	
	PID_POSITION_ERROR[7:0]	
0x19B **MCC_PID_TORQUE_INTEGRATOR**	PID_TORQUE_INTEGRATOR[31:24]	
	PID_TORQUE_INTEGRATOR[23:16]	
	PID_TORQUE_INTEGRATOR[15:8]	
	PID_TORQUE_INTEGRATOR[7:0]	
0x19C **MCC_PID_FLUX_INTEGRATOR**	PID_FLUX_INTEGRATOR[31:24]	
	PID_FLUX_INTEGRATOR[23:16]	
	PID_FLUX_INTEGRATOR[15:8]	
	PID_FLUX_INTEGRATOR[7:0]	
0x19D **MCC_PID_VELOCITY_INTEGRATOR**	PID_VELOCITY_INTEGRATOR[31:24]	
	PID_VELOCITY_INTEGRATOR[23:16]	
	PID_VELOCITY_INTEGRATOR[15:8]	
	PID_VELOCITY_INTEGRATOR[7:0]	
0x19E **MCC_PID_POSITION_INTEGRATOR**	PID_POSITION_INTEGRATOR[31:24]	
	PID_POSITION_INTEGRATOR[23:16]	
	PID_POSITION_INTEGRATOR[15:8]	
	PID_POSITION_INTEGRATOR[7:0]	
0x1A0 **MCC_PIDIN_TORQUE_FLUX_TARGET**	PIDIN_TORQUE_TARGET[15:8]	
	PIDIN_TORQUE_TARGET[7:0]	
	PIDIN_FLUX_TARGET[15:8]	
	PIDIN_FLUX_TARGET[7:0]	
0x1A1 **MCC_PIDIN_VELOCITY_TARGET**	PIDIN_VELOCITY_TARGET[31:24]	
	PIDIN_VELOCITY_TARGET[23:16]	
	PIDIN_VELOCITY_TARGET[15:8]	
	PIDIN_VELOCITY_TARGET[7:0]	

ADDRESS & NAME	FIELDS	MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x1A2 **MCC_PIDIN_POSITION_TARGET**	PIDIN_POSITION_TARGET[31:24]	
	PIDIN_POSITION_TARGET[23:16]	
	PIDIN_POSITION_TARGET[15:8]	
	PIDIN_POSITION_TARGET[7:0]	
0x1A3 **MCC_PIDIN_TORQUE_FLUX_TARGET_LIMITED**	PIDIN_TORQUE_TARGET_LIMITED[15:8]	
	PIDIN_TORQUE_TARGET_LIMITED[7:0]	
	PIDIN_FLUX_TARGET_LIMITED[15:8]	
	PIDIN_FLUX_TARGET_LIMITED[7:0]	
0x1A4 **MCC_PIDIN_VELOCITY_TARGET_LIMITED**	PIDIN_VELOCITY_TARGET_LIMITED[31:24]	
	PIDIN_VELOCITY_TARGET_LIMITED[23:16]	
	PIDIN_VELOCITY_TARGET_LIMITED[15:8]	
	PIDIN_VELOCITY_TARGET_LIMITED[7:0]	
0x1A5 **MCC_PIDIN_POSITION_TARGET_LIMITED**	PIDIN_POSITION_TARGET_LIMITED[31:24]	
	PIDIN_POSITION_TARGET_LIMITED[23:16]	
	PIDIN_POSITION_TARGET_LIMITED[15:8]	
	PIDIN_POSITION_TARGET_LIMITED[7:0]	
0x1A6 **MCC_FOC_IBETA_IALPHA**	IBETA[15:8]	
	IBETA[7:0]	
	IALPHA[15:8]	
	IALPHA[7:0]	
0x1A7 **MCC_FOC_IQ_ID**	IQ[15:8]	
	IQ[7:0]	
	ID[15:8]	
	ID[7:0]	
0x1A8 **MCC_FOC_UQ_UD**	UQ[15:8]	
	UQ[7:0]	
	UD[15:8]	
	UD[7:0]	
0x1A9 **MCC_FOC_UQ_UD_LIMITED**	UQ[15:8]	
	UQ[7:0]	
	UD[15:8]	
	UD[7:0]	

ADDRESS & NAME	FIELDS	MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x1AA **MCC_FOC_UBETA_UALPHA**	UBETA[15:8]	
	UBETA[7:0]	
	UALPHA[15:8]	
	UALPHA[7:0]	
0x1AB **MCC_FOC_UWY_UUX**	UWY[15:8]	
	UWY[7:0]	
	UUX[15:8]	
	UUX[7:0]	
0x1AC **MCC_FOC_UV**		
	UV[15:8]	
	UV[7:0]	
0x1AD **MCC_PWM_VX2_UX1**	VX2[15:8]	
	VX2[7:0]	
	UX1[15:8]	
	UX1[7:0]	
0x1AE **MCC_PWM_Y2_WY1**	Y2[15:8]	
	Y2[7:0]	
	WY1[15:8]	
	WY1[7:0]	
0x1AF **MCC_VELOCITY_FRQ**	VELOCITY_FRQ[31:24]	
	VELOCITY_FRQ[23:16]	
	VELOCITY_FRQ[15:8]	
	VELOCITY_FRQ[7:0]	
0x1B0 **MCC_VELOCITY_PER**	VELOCITY_PER[31:24]	
	VELOCITY_PER[23:16]	
	VELOCITY_PER[15:8]	
	VELOCITY_PER[7:0]	
0x1C0 **MCC_U_S_ACTUAL_I_S_ACTUAL**	U_S_ACTUAL[15:8]	
	U_S_ACTUAL[7:0]	
	I_S_ACTUAL[15:8]	
	I_S_ACTUAL[7:0]	

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x1C1 **MCC_P_MOTOR**	P_MOTOR[31:24]							
	P_MOTOR[23:16]							
	P_MOTOR[15:8]							
	P_MOTOR[7:0]							
0x1C2 **MCC_INPUTS_RAW**								
		HALL_W_FILT	HALL_V_FILT	HALL_U_FILT				
	ENI	REF_SW_H	REF_SW_L	REF_SW_R		HALL_W	HALL_V	HALL_U
						ENC_N	ENC_B	ENC_A
0x1C3 **MCC_OUTPUTS_RAW**								
	PWM_Y2_H	PWM_Y2_L	PWM_WY1_H	PWM_WY1_L	PWM_VX2_H	PWM_VX2_L	PWM_UX1_H	PWM_UX1_L
0x1C4 **MCC_STATUS_FLAGS**	ENI			ENC_N		ADC_I_CLIPPED		
	POSITION_REACHED	REF_SW_H	REF_SW_R	REF_SW_L			SHORT	
		PID_FW_OUTPUT_LIMIT	VELOCITY_TRACKING_ERROR	POSITION_TRACKING_ERROR	HALL_ERROR		PWM_SWITCH_LIMIT_ACTIVE	IPARK_VOLTLIM_LIMIT_U
	PID_IQ_OUTPUT_LIMIT	PID_IQ_TARGET_LIMIT	PID_ID_OUTPUT_LIMIT	PID_ID_TARGET_LIMIT	PID_V_OUTPUT_LIMIT	PID_V_TARGET_LIMIT	PID_X_OUTPUT_LIMIT	PID_X_TARGET_LIMIT
0x1E3 **MCC_GDRV_HW**				HS_AS_LS_Y2			BIAS_EN	CHARGEPUMP_EN
					BST_SW_CP_EN	BST_ILIM_MAX		
	VS_UVLO_CMP_EN	VDRV_UVLO_CMP_EN	HS_OCP_CMP_EN	LS_OCP_CMP_EN	BRIDGE_ENABLE_Y2	BRIDGE_ENABLE_W	BRIDGE_ENABLE_V	BRIDGE_ENABLE_U

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x1E4 **MCC_GDRV_CFG**								
	VS_UVLO_LVL						ADAPTIVE_MODE_Y2	ADAPTIVE_MODE_UVW
	IGATE_SOURCE_Y2				IGATE_SINK_Y2			
	IGATE_SOURCE_UVW				IGATE_SINK_UVW			
0x1E9 **MCC_GDRV_TIMING**	T_DRIVE_SOURCE_Y2							
	T_DRIVE_SINK_Y2							
	T_DRIVE_SOURCE_UVW							
	T_DRIVE_SINK_UVW							
0x1EA **MCC_GDRV_BBM**	BBM_H_Y2							
	BBM_L_Y2							
	BBM_H_UVW							
	BBM_L_UVW							
0x1EB **MCC_GDRV_PROT**				TERM_PWM_ON_SHORT				
	HS_RETRIES_Y2		LS_RETRIES_Y2		HS_RETRIES_UVW		LS_RETRIES_UVW	
			VGS_BLANKING_Y2				VGS_DEGLITCH_Y2	
			VGS_BLANKING_UVW				VGS_DEGLITCH_UVW	
0x1EC **MCC_GDRV_OCP_UVW**					HS_OCP_THRES_UVW			
		HS_OCP_BLANKING_UVW					HS_OCP_DEGLITCH_UVW	
	LS_OCP_USE_VDS_UVW				LS_OCP_THRES_UVW			
		LS_OCP_BLANKING_UVW					LS_OCP_DEGLITCH_UVW	
0x1ED **MCC_GDRV_OCP_Y2**					HS_OCP_THRES_Y2			
		HS_OCP_BLANKING_Y2					HS_OCP_DEGLITCH_Y2	
	LS_OCP_USE_VDS_Y2				LS_OCP_THRES_Y2			
		LS_OCP_BLANKING_Y2					LS_OCP_DEGLITCH_Y2	

ADDRESS & NAME	FIELDS				MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)			
0x1EE **MCC_GDRV_PROT_EN**	VS_UVLO_PROT		VDRV_UVLO_PROT		HS_VGS_ON_SHORT_PROT_Y2	HS_VGS_ON_SHORT_PROT_W	HS_VGS_ON_SHORT_PROT_V	HS_VGS_ON_SHORT_PROT_U
	HS_VGS_OFF_SHORT_PROT_Y2	HS_VGS_OFF_SHORT_PROT_W	HS_VGS_OFF_SHORT_PROT_V	HS_VGS_OFF_SHORT_PROT_U	HS_SHORT_PROT_Y2	HS_SHORT_PROT_W	HS_SHORT_PROT_V	HS_SHORT_PROT_U
	BST_UVLO_PROT_Y2	BST_UVLO_PROT_W	BST_UVLO_PROT_V	BST_UVLO_PROT_U	LS_VGS_ON_SHORT_PROT_Y2	LS_VGS_ON_SHORT_PROT_W	LS_VGS_ON_SHORT_PROT_V	LS_VGS_ON_SHORT_PROT_U
	LS_VGS_OFF_SHORT_PROT_Y2	LS_VGS_OFF_SHORT_PROT_W	LS_VGS_OFF_SHORT_PROT_V	LS_VGS_OFF_SHORT_PROT_U	LS_SHORT_PROT_Y2	LS_SHORT_PROT_W	LS_SHORT_PROT_V	LS_SHORT_PROT_U
0x1EF **MCC_GDRV_STATUS_EN**	VS_UVLO_EN	VDRV_UVLWRN_EN	VDRV_UVLO_EN		HS_VGS_ON_SHORT_EN_Y2	HS_VGS_ON_SHORT_EN_W	HS_VGS_ON_SHORT_EN_V	HS_VGS_ON_SHORT_EN_U
	HS_VGS_OFF_SHORT_EN_Y2	HS_VGS_OFF_SHORT_EN_W	HS_VGS_OFF_SHORT_EN_V	HS_VGS_OFF_SHORT_EN_U	HS_SHORT_EN_Y2	HS_SHORT_EN_W	HS_SHORT_EN_V	HS_SHORT_EN_U
	BST_UVLO_EN_Y2	BST_UVLO_EN_W	BST_UVLO_EN_V	BST_UVLO_EN_U	LS_VGS_ON_SHORT_EN_Y2	LS_VGS_ON_SHORT_EN_W	LS_VGS_ON_SHORT_EN_V	LS_VGS_ON_SHORT_EN_U
	LS_VGS_OFF_SHORT_EN_Y2	LS_VGS_OFF_SHORT_EN_W	LS_VGS_OFF_SHORT_EN_V	LS_VGS_OFF_SHORT_EN_U	LS_SHORT_EN_Y2	LS_SHORT_EN_W	LS_SHORT_EN_V	LS_SHORT_EN_U
0x1F0 **MCC_GDRV_STATUS**	VS_UVLO	VDRV_UVLWRN	VDRV_UVLO		HS_VGS_ON_SHORT_Y2	HS_VGS_ON_SHORT_W	HS_VGS_ON_SHORT_V	HS_VGS_ON_SHORT_U
	HS_VGS_OFF_SHORT_Y2	HS_VGS_OFF_SHORT_W	HS_VGS_OFF_SHORT_V	HS_VGS_OFF_SHORT_U	HS_SHORT_Y2	HS_SHORT_W	HS_SHORT_V	HS_SHORT_U
	BST_UVLO_Y2	BST_UVLO_W	BST_UVLO_V	BST_UVLO_U	LS_VGS_ON_SHORT_Y2	LS_VGS_ON_SHORT_W	LS_VGS_ON_SHORT_V	LS_VGS_ON_SHORT_U
	LS_VGS_OFF_SHORT_Y2	LS_VGS_OFF_SHORT_W	LS_VGS_OFF_SHORT_V	LS_VGS_OFF_SHORT_U	LS_SHORT_Y2	LS_SHORT_W	LS_SHORT_V	LS_SHORT_U

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x1F1 **MCC_GDRV_FAULT**	VS_UVLO_STS	VDRV_UVLWRN_STS	VDRV_UVLO_STS					
					HS_FAULT_ACTIVE_Y2	HS_FAULT_ACTIVE_W	HS_FAULT_ACTIVE_V	HS_FAULT_ACTIVE_U
	BST_UVLO_STS_Y2	BST_UVLO_STS_W	BST_UVLO_STS_V	BST_UVLO_STS_U				
					LS_FAULT_ACTIVE_Y2	LS_FAULT_ACTIVE_W	LS_FAULT_ACTIVE_V	LS_FAULT_ACTIVE_U
0x200 **MCC_ADC_I1_I0_EXT**	I1[15:8]							
	I1[7:0]							
	I0[15:8]							
	I0[7:0]							
0x201 **MCC_ADC_I2_EXT**								
	I2[15:8]							
	I2[7:0]							
0x202 **MCC_PWM_VX2_UX1_EXT**	VX2[15:8]							
	VX2[7:0]							
	UX1[15:8]							
	UX1[7:0]							
0x203 **MCC_PWM_Y2_WY1_EXT**	Y2[15:8]							
	Y2[7:0]							
	WY1[15:8]							
	WY1[7:0]							
0x204 **MCC_PWM_EXT_Y2_ALT**								
	PWM_EXT_Y2_ALT[15:8]							
	PWM_EXT_Y2_ALT[7:0]							
0x205 **MCC_VOLTAGE_EXT**	UQ[15:8]							
	UQ[7:0]							
	UD[15:8]							
	UD[7:0]							

ADDRESS & NAME	FIELDS	MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x206 **MCC_PHI_EXT**	PHI_M_EXT[15:8]	
	PHI_M_EXT[7:0]	
	PHI_E_EXT[15:8]	
	PHI_E_EXT[7:0]	
0x208 **MCC_VELOCITY_EXT**	VELOCITY_EXT[31:24]	
	VELOCITY_EXT[23:16]	
	VELOCITY_EXT[15:8]	
	VELOCITY_EXT[7:0]	

ADC Register Overview

Shows all related registers of the ADC block

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x001 **ADC_SRC_CONFIG**	ADC3_MUX2_DETOUR		ADC3_MUX2_CFG		ADC3_MUX1_CFG		ADC3_MUX0_CFG	
	ADC2_MUX2_DETOUR	ADC2_MUX3_DIS	ADC2_MUX2_CFG		ADC2_MUX1_CFG		ADC2_MUX0_CFG	
	ADC1_MUX2_DETOUR		ADC1_MUX2_CFG		ADC1_MUX1_CFG		ADC1_MUX0_CFG	
	ADC0_MUX2_DETOUR	ADC0_MUX3_DIS	ADC0_MUX2_CFG		ADC0_MUX1_CFG		ADC0_MUX0_CFG	
0x002 **ADC_SETUP**								
					ADC_SHIFT_SAMPLE			
0x005 **ADC_STATUS**								
	ADC3_MUXSEQ_FAIL	ADC2_MUXSEQ_FAIL	ADC1_MUXSEQ_FAIL	ADC0_MUXSEQ_FAIL	ADC3_WTCHDG_FAIL	ADC2_WTCHDG_FAIL	ADC1_WTCHDG_FAIL	ADC0_WTCHDG_FAIL
					RDY_ADC_3	RDY_ADC_2	RDY_ADC_1	RDY_ADC_0

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x007 **CSA_SETUP**								
					CSA_AZ_FLTLNGTH_EXP			
	CSA3_FILT		CSA012_FILT			CSA3_ BYPASS	CSA3_GAIN	
		CSA012_ BYPASS	CSA012_GAIN		CSA3_EN	CSA2_EN	CSA1_EN	CSA0_EN

SYS_CTRL Register Overview

Shows all related registers of the SYS_CTRL block

ADDRESS & NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x008 **FAULT_STS**								
				LDO2_ READY	LDO1_ READY	VCCIO_ UVLO	VDDA_ UVLO	VDD_UVLO
	VSA_UVLO	CHGP_ SHORT	CHGP_OK	LDOEXT2_ SHORT	LDOEXT1_ SHORT	LDOEXT_ TSD	BCK_ SHORT	BCK_UVLO
0x009 **FAULT_R_INT**								
	UC_FAULT			LDO2_ READY_ RE_LTC	LDO1_ READY_ RE_LTC	VCCIO_ UVLO_LTC	VDDA_ UVLO_LTC	VDD_ UVLO_LTC
	VSA_ UVLO_LTC	CHGP_ SHORT_ LTC	CHGP_OK_ LTC	LDOEXT2_ SHORT_ LTC	LDOEXT1_ SHORT_ LTC	LDOEXT_ TSD_LTC	BCK_ SHORT_ RE_LTC	BCK_ UVLO_LTC
0x00A **FAULT_R_ENA_F**								
				LDO2_ READY_ RE_ENA_F	LDO1_ READY_ RE_ENA_F	VCCIO_ UVLO_ ENA_F	VDDA_ UVLO_ ENA_F	VDD_ UVLO_ ENA_F
	VSA_ UVLO_ ENA_F	CHGP_ SHORT_ ENA_F	CHGP_OK_ ENA_F	LDOEXT2_ SHORT_ ENA_F	LDOEXT1_ SHORT_ ENA_F	LDOEXT_ TSD_ENA_ F	BCK_ SHORT_ RE_ENA_F	BCK_ UVLO_ ENA_F

0x000, Block 0: MCC_INFO_CHIP

Chip ID

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **ID**	R, unsigned 0x544D0001	Chip ID, static value, should read 0x544D0001

0x020, Block 0: MCC_ADC_I1_I0_RAW

Raw phase currents I1, I0

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **I1**	R, signed 0x0000	Raw phase current I1
[15:0] **I0**	R, signed 0x0000	Raw phase current I0

0x021, Block 0: MCC_ADC_I3_I2_RAW

Raw phase currents I3, I2

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **I3**	R, signed 0x0000	Raw phase current I3
[15:0] **I2**	R, signed 0x0000	Raw phase current I2

0x022, Block 0: MCC_ADC_U1_U0_RAW

Measured phase voltages U1 and U0

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **U1**	R, signed 0x0000	Raw phase voltage U1
[15:0] **U0**	R, signed 0x0000	Raw phase voltage U0

0x023, Block 0: MCC_ADC_U3_U2_RAW

Measured phase voltage

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **U3**	R, signed 0x0000	Raw phase voltage U3
[15:0] **U2**	R, signed 0x0000	Raw phase voltage U2

0x024, Block 0: MCC_ADC_TEMP_VM_RAW

Raw Die temperature voltage and supply voltage monitoring value

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **TEMP**	R, signed 0x0000	Raw Die temperature voltage value
[15:0] **VM**	R, signed 0x0000	Raw supply voltage monitoring value

0x025, Block 0: MCC_ADC_AIN1_AIN0_RAW

Raw analog values AIN1 and AIN0

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **AIN1**	R, signed 0x0000	Raw analog input 1 value
[15:0] **AIN0**	R, signed 0x0000	Raw analog input 0 value

0x026, Block 0: MCC_ADC_AIN3_AIN2_RAW

Raw analog values AIN3 and AIN2

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **AIN3**	R, signed 0x0000	Raw analog input 3 value
[15:0] **AIN2**	R, signed 0x0000	Raw analog input 2 value

0x040, Block 0: MCC_ADC_I_GEN_CONFIG

General configuration setup of the current ADCs

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **TRIGGER_POS**	RW, unsigned 0x0000	Relative position of ADC trigger event in PWM cycle. Percentage of maxcnt (0 -> PWM_Z, 32768 -> PWM_C)
[12] **TRIGGER_SELECT**	RW 0x1	Select trigger point to start process of new ADC samples and start next FOC calculation afterwards. Per default, FOC calculation starts after internal ADC current measurement is finished which starts when internal PWM_Z signal triggers.
	0: INLINE 1: SYNC_TRIGGER	Trigger on start of each PWM cycle (PWM_Z). Trigger when ADC current measurement is finished (default).
[11:9] **MEASUREMENT_MODE**	RW 0x0	Configuration of measurement mode
	0: INLINE 1: INLINE_VW 2: INLINE_UW 3: INLINE_UV 4: BOTTOM	3 channel BLDC/2 channel Stepper Inline Shunt Measurement 2 channels with I_V and I_WY measured (BLDC) 2 channels with I_UX and I_WY measured (BLDC) 2 channels with I_UX and I_V measured (BLDC) 3/4 phase bottom shunt with automatic switching (BLDC and Stepper)
[7:6] **Y2_SELECT**	RW 0x3	Input selection of raw current ADC_I_Y2.
	0: ADC_I0 1: ADC_I1 2: ADC_I2 3: ADC_I3	
[5:4] **WY1_SELECT**	RW 0x2	Input selection of raw current ADC_I_WY1
	0: ADC_I0 1: ADC_I1 2: ADC_I2 3: ADC_I3	
[3:2] **VX2_SELECT**	RW 0x1	Input selection of raw current ADC_I_VX2
	0: ADC_I0 1: ADC_I1 2: ADC_I2 3: ADC_I3	
[1:0] **UX1_SELECT**	RW 0x0	Input selection of raw current ADC_I_UX
	0: ADC_I0 1: ADC_I1 2: ADC_I2 3: ADC_I3	

0x041, Block 0: MCC_ADC_I0_CONFIG

Current ADC channel 0 offset and scaling values

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **SCALE**	RW, signed 0xFC00	Current ADC channel 0 scaling value
[15:0] **OFFSET**	RW, signed 0x0000	Current ADC channel 0 offset value

0x042, Block 0: MCC_ADC_I1_CONFIG

Current ADC channel 1 offset and scaling values

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **SCALE**	RW, signed 0xFC00	Current ADC channel 1 scaling value
[15:0] **OFFSET**	RW, signed 0x0000	Current ADC channel 1 offset value

0x043, Block 0: MCC_ADC_I2_CONFIG

Current ADC channel 2 offset and scaling values

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **SCALE**	RW, signed 0xFC00	Current ADC channel 2 scaling value
[15:0] **OFFSET**	RW, signed 0x0000	Current ADC channel 2 offset value

0x044, Block 0: MCC_ADC_I3_CONFIG

Current ADC channel 3 offset and scaling values

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **SCALE**	RW, signed 0xFC00	Current ADC channel 3 scaling value
[15:0] **OFFSET**	RW, signed 0x0000	Current ADC channel 3 offset value

0x045, Block 0: MCC_ADC_I1_I0_SCALED

Phase current I1, I0; after applying scaling and offset.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **I1**	R, signed 0x0000	Calculated Phase Current I1 after applying scaling and offset for further processing
[15:0] **I0**	R, signed 0x0000	Calculated Phase Current I0 after applying scaling and offset for further processing

0x046, Block 0: MCC_ADC_I3_I2_SCALED

Phase current I3, I2; after applying scaling and offset.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **I3**	R, signed 0x0000	Calculated Phase Current I3 after applying scaling and offset for further processing
[15:0] **I2**	R, signed 0x0000	Calculated Phase Current I2 after applying scaling and offset for further processing

0x047, Block 0: MCC_ADC_IWY_IUX

Scaled current ADC value including signed added offset as input for the FOC.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **IWY**	R, signed 0x0000	Scaled current ADC value including signed added offset as input for the FOC phase W/Y.
[15:0] **IUX**	R, signed 0x0000	Scaled current ADC value including signed added offset as input for the FOC phase U/X.

0x048, Block 0: MCC_ADC_IV

Scaled current ADC value including signed added offset as input for the FOC.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **IV**	R, signed 0x0000	Scaled current ADC value including signed added offset as input for the FOC phase V.

0x049, Block 0: MCC_ADC_STATUS

ADC status flags

BITS & NAME	TYPE & RESET	DESCRIPTION
[29] **TEMP_DONE**	RW, W1C 0x0	ADC temperature voltage measurement finished. Flag is automatically cleared when new measurement is started.
[28] **VM_DONE**	RW, W1C 0x0	ADC supply voltage measurement VM finished. Flag is automatically cleared when new measurement is started.
[27] **AIN3_DONE**	RW, W1C 0x0	ADC measurement AIN3 finished. Flag is automatically cleared when new measurement is started.
[26] **AIN2_DONE**	RW, W1C 0x0	ADC measurement AIN2 finished. Flag is automatically cleared when new measurement is started.
[25] **AIN1_DONE**	RW, W1C 0x0	ADC measurement AIN1 finished. Flag is automatically cleared when new measurement is started.
[24] **AIN0_DONE**	RW, W1C 0x0	ADC measurement AIN0 finished. Flag is automatically cleared when new measurement is started.
[23] **U3_DONE**	RW, W1C 0x0	ADC voltage measurement U3 finished. Flag is automatically cleared when new measurement is started.
[22] **U2_DONE**	RW, W1C 0x0	ADC voltage measurement U2 finished. Flag is automatically cleared when new measurement is started.
[21] **U1_DONE**	RW, W1C 0x0	ADC voltage measurement U1 finished. Flag is automatically cleared when new measurement is started.
[20] **U0_DONE**	RW, W1C 0x0	ADC voltage measurement U0 finished. Flag is automatically cleared when new measurement is started.
[19] **I3_DONE**	RW, W1C 0x0	ADC current measurement I3 finished. Flag is automatically cleared when new measurement is started.
[18] **I2_DONE**	RW, W1C 0x0	ADC current measurement I2 finished. Flag is automatically cleared when new measurement is started.
[17] **I1_DONE**	RW, W1C 0x0	ADC current measurement I1 finished. Flag is automatically cleared when new measurement is started.
[16] **I0_DONE**	RW, W1C 0x0	ADC current measurement I0 finished. Flag is automatically cleared when new measurement is started.

BITS & NAME	TYPE & RESET	DESCRIPTION
[13] **TEMP_CLIPPED**	RW, W1C 0x0	ADC temperature voltage measurement value clipped
[12] **VM_CLIPPED**	RW, W1C 0x0	ADC supply voltage measurement VM value clipped
[11] **AIN3_CLIPPED**	RW, W1C 0x0	ADC measurement AIN3 value clipped
[10] **AIN2_CLIPPED**	RW, W1C 0x0	ADC measurement AIN2 value clipped
[9] **AIN1_CLIPPED**	RW, W1C 0x0	ADC measurement AIN1 value clipped
[8] **AIN0_CLIPPED**	RW, W1C 0x0	ADC measurement AIN0 value clipped
[7] **U3_CLIPPED**	RW, W1C 0x0	ADC voltage measurement U3 value clipped
[6] **U2_CLIPPED**	RW, W1C 0x0	ADC voltage measurement U2 value clipped
[5] **U1_CLIPPED**	RW, W1C 0x0	ADC voltage measurement U1 value clipped
[4] **U0_CLIPPED**	RW, W1C 0x0	ADC voltage measurement U0 value clipped
[3] **I3_CLIPPED**	RW, W1C 0x0	ADC current measurement I3 value clipped
[2] **I2_CLIPPED**	RW, W1C 0x0	ADC current measurement I2 value clipped
[1] **I1_CLIPPED**	RW, W1C 0x0	ADC current measurement I1 value clipped
[0] **I0_CLIPPED**	RW, W1C 0x0	ADC current measurement I0 value clipped

0x060, Block 0: MCC_MOTOR_CONFIG

Motor type configuration and number of pole pairs

BITS & NAME	TYPE & RESET	DESCRIPTION
[17:16] **TYPE**	RW 0x0	Motor type
	0: NONE 1: DC 2: STEPPER 3: BLDC	No motor Single phase DC motor Two phase Stepper motor Three phase BLDC motor
[6:0] **N_POLE_PAIRS**	RW, unsigned 0x1	Number n of pole pairs of the motor for calculation phi_e = phi_m / N_POLE_PAIRS. Min: 1

0x061, Block 0: MCC_MOTION_CONFIG

Register for selection of ramp mode, motion mode and controller feedforward.

BITS & NAME	TYPE & RESET	DESCRIPTION
[7:6] **FEEDFORWARD**	RW 0x0	Control of the Feedforward structure. Feedforward for the torque controller is available based on the acceleration of the ramp generator. Feedforward for the velocity controller is available based on the ramp generator velocity.
	0: DISABLED 1: MCC_RAMPER_V_ACTUAL 2: MCC_RAMPER_A_ACTUAL 3: BOTH	Disabled MCC_RAMPER_V_ACTUAL used as feedforward input signal for velocity controller. MCC_RAMPER_A_ACTUAL used as feedforward input signal for torque controller. Scaling applied from MCC_RAMPER_ACC_FF. Both feedforward inputs for velocity and torque are used.
[5] **RAMP_MODE**	RW 0x0	Selection of Ramp Mode.
	0: POSITION 1: VELOCITY	Position Mode Velocity Mode
[4] **RAMP_ENABLE**	RW 0x0	Enable ramp generator
[3:0] **MOTION_MODE**	RW 0x0	Configuration of motion mode.
	0: STOPPED 1: TORQUE 2: VELOCITY 3: POSITION 4: PRBS_FLUX 5: PRBS_TORQUE 6: PRBS_VELOCITY 7: PRBS_POSITION 8: VOLTAGE_EXT 9: PRBS_UD	Stopped mode Torque mode Velocity mode Position mode PRBS flux mode PRBStorque mode PRBSvelocity mode PRBSposition mode Voltage external mode PRBSud mode

0x062, Block 0: MCC_PHI_E_SELECTION

Selection of phi resp. encoder source for rotor position angle phi_e that is used as the input for the FOC for commutation.

BITS & NAME	TYPE & RESET	DESCRIPTION
[3:0] **PHI_E_SELECTION**	RW 0x0	Selection of PHI_E for commutation from available sources.
	0: RESERVED — Reserved 1: PHI_E_EXT — phi_e_ext 2: PHI_E_RAMP — phi_e_ramp 3: PHI_E_ABN — phi_e_abn 4: RAMP_X_ACTUAL — ramp_X_actual 5: PHI_E_HAL — phi_e_hal	

0x063, Block 0: MCC_PHI_E

Angle used for the inner FOC loop.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **PHI_E**	R, signed 0x0000	Angle used for the inner FOC loop.

0x080, Block 0: MCC_PWM_CONFIG

PWM Configuration: Chopper Mode, Enable Space Vector, Enable Bridge, Y2 Source, Offset.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **DUTY_CYCLE_OFFSET**	RW, unsigned 0x0000	Offset for PWM duty cycle in Flat Bottom Modulation with Offset Mode, Set in SV_MODE
[15] **EXT_ENABLE_Y2**	RW 0x0	Enable to use PWM_EXT value for channel
[14] **EXT_ENABLE_WY1**	RW 0x0	Enable to use PWM_EXT value for channel
[13] **EXT_ENABLE_VX2**	RW 0x0	Enable to use PWM_EXT value for channel
[12] **EXT_ENABLE_UX1**	RW 0x0	Enable to use PWM_EXT value for channel
[11] **ENABLE_Y2**	RW 0x0	Enables the PWM starting with the next PWM cycle. If disabled, high- and low-side are driven low.
[10] **ENABLE_WY1**	RW 0x0	Enables the PEB starting with the next PWM cycle. If disabled, high- and low-side are driven low.

BITS & NAME	TYPE & RESET	DESCRIPTION
[9] **ENABLE_VX2**	RW 0x0	Enables the pwm starting with the next PWM cycle. If disabled, high- and low-side are driven low.
[8] **ENABLE_UX1**	RW 0x0	Enables the pwm starting with the next PWM cycle. If disabled, high- and low-side are driven low.
[7:6] **Y2_HS_SRC**	RW 0x0	Source of PWM signal for Y2 high-side. 0: Y2_HS 1: Y2_ALT 2: TIM_BASIC
[5:4] **SV_MODE**	RW 0x0	Select the modulation mode for the PWM output signals. For a BLDC motor the Space Vector (SV) PWM Mode with third harmonic injection can be enabled. 0: DISABLED – DC or stepper motor: Normal modulation. BLDC motor: SVPWM disabled. 1: HARMONIC – DC or stepper motor: Normal modulation. BLDC motor: SVPWM enabled. 2: BOTTOM – DC or stepper motor: Flat bottom modulation. BLDC motor: SVPWM enabled flat bottom. 3: BOTTOM_OFFSET – DC or stepper motor: Normal modulation. BLDC motor: SVPWM enabled flat bottom with offset (DUTY_CYCLE_OFFSET).
[2:0] **CHOP**	RW 0x0	PWM chopper mode, defining how to chopper for Low-Side (LS) and High-Side (HS) outputs. 0: OFF_FREE – PWM off, LS and HS permanently off 1: OFF_LSON – PWM off, LS permanently on, HS off 2: OFF_HSON – PWM off, HS permanently on, LS off 3: OFF_FREE2 – PWM off, LS and HS permanently off 4: OFF_FREE3 – PWM off, LS and HS permanently off 5: LSPWM_HSOFF – PWM on, LS PWM, HS off 6: HSPWM_LSOFF – PWM on, HS PWM, LS off 7: CENTERED – PWM on, centered PWM for FOC

0x081, Block 0: MCC_PWM_MAXCNT

PWM counter maximum length register PWM_MAXCNT controls the PWM frequency. Default 25kHz. PWM frequency = 120MHz / PWM_MAXCNT.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **PWM_MAXCNT**	RW, unsigned 0x12BF	PWM counter maximum length register PWM_MAXCNT controls the PWM frequency. Default 25kHz. PWM frequency = 120MHz / PWM_MAXCNT.

0x083, Block 0: MCC_PWM_SWITCH_LIMIT

Determines a threshold for switching from three phase current measurement to two phase in Auto Kirchhoff.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **PWM_SWITCH_LIMIT**	RW, unsigned 0xFFFF	PWM duty cycles threshold value. Auto kirchhoff calculation is used when one duty cycle exceeds this threshold. A value of 65535 (0xFFFF) is equivalent of 100% duty cycle regardless of the current PWM period.

0x0A0, Block 0: MCC_ABN_PHI_E_PHI_M

ABN Decoder electrical angle PHI_E and mechanical angle PHI_M.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **PHI_E**	R, signed 0x0000	ABN_PHI_E = (ABN_PHI_M × N_POLE_PAIRS) + ABN_PHI_E_OFFSET
[15:0] **PHI_M**	R, signed 0x0000	ABN_PHI_M = ABN_COUNT × 2^16 / ABN_CPR

0x0A1, Block 0: MCC_ABN_MODE

ABN decoder configuration.

BITS & NAME	TYPE & RESET	DESCRIPTION
[12] **DIRECTION**	RW 0x0	Decoder count direction.
	0: POS 1: NEG	positive negative
[8] **CLN**	RW 0x0	N channel event writes ABN_COUNT_N into ABN_COUNT at N pulse instead of 0. Enable CLEAR_COUNT_ON_N to take effect
	0: OFF 1: ON	ABN_COUNT is written to 0 ABN_COUNT is written to ABN_COUNT_N
[5] **DISABLE_FILTER**	RW 0x0	Disable digital filter on '0'.
	0: FILTERED 1: UNFILTERED	Filter enabled: pulses longer 3 system clock cycles are evaluated Filter disabled
[4] **CLEAR_COUNT_ON_N**	RW 0x0	Set ABN_COUNT to 0 on Null signal.
	0: DISABLED 1: ENABLED	Disabled Enabled
[3] **COMBINED_N**	RW 0x0	Use AND of all three signals A, B, N to determine the Null signal.
	0: ONLY_N 1: ALL	Ignore A and B, just use N pulse as Null signal Use all three signals as Null signal

BITS & NAME	TYPE & RESET	DESCRIPTION
[2] **N_POL**	RW 0x0	Polarity of N pulse at Null.
	0: HIGH_ACT 1: LOW_ACT	High active Low active
[1] **B_POL**	RW 0x0	Polarity of B pulse at N. This is only used if COMBINED_N is on.
	0: HIGH_ACT 1: LOW_ACT	High active Low active
[0] **A_POL**	RW 0x0	Polarity of A pulse at N. This is only used if COMBINED_N is on.
	0: HIGH_ACT 1: LOW_ACT	High active Low active

0x0A2, Block 0: MCC_ABN_CPR

Decoder counts per revolution (CPR). ABN_CPR_INV needs to be set as well (ABN_CPR_INV = 2^32 / ABN_CPR).

BITS & NAME	TYPE & RESET	DESCRIPTION
[23:0] **ABN_CPR**	RW, unsigned 0x010000	Decoder counts per revolution (CPR). ABN_CPR_INV needs to be set as well (ABN_CPR_INV = 2^32 / ABN_CPR).

0x0A3, Block 0: MCC_ABN_CPR_INV

2^32 divided by decoder counts per revolution. (ABN_CPR_INV = 2^32 / ABN_CPR).

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **ABN_CPR_INV**	RW, unsigned 0x00010000	2^32 divided by decoder counts per revolution. (ABN_CPR_INV = 2^32 / ABN_CPR).

0x0A4, Block 0: MCC_ABN_COUNT

Raw decoder count. The digital decoder engine counts modulo ABN_CPR.

BITS & NAME	TYPE & RESET	DESCRIPTION
[23:0] **ABN_COUNT**	RW, unsigned 0x000000	Raw decoder count. The digital decoder engine counts modulo ABN_CPR.

0x0A5, Block 0: MCC_ABN_COUNT_N

ABN_COUNT latched on N pulse, when N pulse clears ABN_COUNT then ABN_COUNT_N is also 0.

BITS & NAME	TYPE & RESET	DESCRIPTION
[23:0] **ABN_COUNT_N**	RW, unsigned 0x000000	ABN_COUNT latched on N pulse, when N pulse clears ABN_COUNT then ABN_COUNT_N is also 0.

0x0A6, Block 0: MCC_ABN_PHI_E_OFFSET

Offset for ABN_PHI_E.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **ABN_PHI_E_OFFSET**	RW, signed 0x0000	Offset for ABN_PHI_E.

0x0C0, Block 0: MCC_HALL_MODE

Hall decoder configuration.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:8] **FILTER**	RW, unsigned 0x00	Define filter length for hall signals. Each hall input signal must be stable the number of clock cycles specified by this register value before new values are accepted. "Clock cycles" means usually 40MHz system clock depending on clock configuration.
[6:4] **ORDER**	RW 0x0	Ordering of the hall signals.
	0: UVW — Hall Signal Order U/V/W 1: VWU — Hall Signal Order V/W/U 2: WUV — Hall Signal Order W/U/V 3: RESERVED — reserved 4: UWV — Hall Signal Order U/W/V 5: VUW — Hall Signal Order V/U/W 6: WVU — Hall Signal Order W/V/U 7: RESERVED2 — reserved	
[1] **EXTRAPOLATION**	RW 0x0	Enable Extrapolation for PHI_E.
	0: DISABLED — Raw signal is used for HALL_PHI_E 1: ENABLED — HALL_PHI_E_EXTRAPOLATED is used for HALL_PHI_E	
[0] **POLARITY**	RW 0x0	Polarity
	0: NORMAL — off 1: INVERSED — on	

0x0C1, Block 0: MCC_HALL_DPHI_MAX

Maximum dx for extrapolation (default for digital hall: (2^16)/6).

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **HALL_DPHI_MAX**	RW, unsigned 0x2AAA	Maximum phi_e change for extrapolation (default for digital hall: (2^16)/6). Extrapolation of phi_e stops after HALL_DPHI_MAX if no new hall position is detected.

0x0C2, Block 0: MCC_HALL_PHI_E_OFFSET

Offset for electrical angle hall_phi_e of hall decoder.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **HALL_PHI_E_OFFSET**	RW, signed 0x0000	Offset for electrical angle hall_phi_e of hall decoder.

0x0C3, Block 0: MCC_HALL_COUNT

Count of passed hall states.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **HALL_COUNT**	R, signed 0x0000	Count of passed hall states.

0x0C4, Block 0: MCC_HALL_PHI_E_EXTRAPOLATED_PHI_E

Hall decoder electrical angle PHI_E.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **PHI_E_EXTRAPOLATED**	R, signed 0x0000	Extrapolated electrical angle hall_phi_e_extrapolated.
[15:0] **PHI_E**	R, signed 0x0000	Electrical angle hall_phi_e of hall decoder. Can be either raw or extrapolated, selection programmed via HALL_MODE EXTRAPOLATION bit.

0x0C5, Block 0: MCC_HALL_POSITION_060_POSITION_000

Position of the hall sensor.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **POSITION_060**	RW, signed 0x2AAA	Exact position of the hall sensor at 60°.
[15:0] **POSITION_000**	RW, signed 0x0000	Exact position of the hall sensor at 0°.

0x0C6, Block 0: MCC_HALL_POSITION_180_POSITION_120

Position of the hall sensor.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **POSITION_180**	RW, signed 0x8000	Exact position of the hall sensor at 180°.
[15:0] **POSITION_120**	RW, signed 0x5555	Exact position of the hall sensor at 120°.

0x0C7, Block 0: MCC_HALL_POSITION_300_POSITION_240

Position of the hall sensor.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **POSITION_300**	RW, signed 0xD555	Exact position of the hall sensor at 300°.
[15:0] **POSITION_240**	RW, signed 0xAAAA	Exact position of the hall sensor at 240°.

0x0E0, Block 0: MCC_BIQUAD_V_A_1

Biquad velocity filter coefficient A_1.

BITS & NAME	TYPE & RESET	DESCRIPTION
[23:0] **BIQUAD_V_A_1**	RW, signed 0x1C376B	Biquad velocity filter coefficient A_1.

0x0E1, Block 0: MCC_BIQUAD_V_A_2

Biquad velocity filter coefficient A_2.

BITS & NAME	TYPE & RESET	DESCRIPTION
[23:0] **BIQUAD_V_A_2**	RW, signed 0xF38F52	Biquad velocity filter coefficient A_2.

0x0E2, Block 0: MCC_BIQUAD_V_B_0

Biquad velocity filter coefficient B_0.

BITS & NAME	TYPE & RESET	DESCRIPTION
[23:0] **BIQUAD_V_B_0**	RW, signed 0x000E51	Biquad velocity filter coefficient B_0.

0x0E3, Block 0: MCC_BIQUAD_V_B_1

Biquad velocity filter coefficient B_1.

BITS & NAME	TYPE & RESET	DESCRIPTION
[23:0] **BIQUAD_V_B_1**	RW, signed 0x001CA1	Biquad velocity filter coefficient B_1.

0x0E4, Block 0: MCC_BIQUAD_V_B_2

Biquad velocity filter coefficient B_2.

BITS & NAME	TYPE & RESET	DESCRIPTION
[23:0] **BIQUAD_V_B_2**	RW, signed 0x000E51	Biquad velocity filter coefficient B_2.

0x0E5, Block 0: MCC_BIQUAD_V_ENABLE

Biquad velocity filter enable.

BITS & NAME	TYPE & RESET	DESCRIPTION
[0] **BIQUAD_V_ENABLE**	RW 0x1	Enable Biquad Velocity Filter

0x0E6, Block 0: MCC_BIQUAD_T_A_1

Biquad torque filter coefficient A_1.

BITS & NAME	TYPE & RESET	DESCRIPTION
[23:0] **BIQUAD_T_A_1**	RW, signed 0x000000	Biquad torque filter coefficient A_1.

0x0E7, Block 0: MCC_BIQUAD_T_A_2

Biquad torque filter coefficient A_2.

BITS & NAME	TYPE & RESET	DESCRIPTION
[23:0] **BIQUAD_T_A_2**	RW, signed 0x000000	Biquad torque filter coefficient A_2.

0x0E8, Block 0: MCC_BIQUAD_T_B_0

Biquad torque filter coefficient B_0.

BITS & NAME	TYPE & RESET	DESCRIPTION
[23:0] **BIQUAD_T_B_0**	RW, signed 0x100000	Biquad torque filter coefficient B_0.

0x0E9, Block 0: MCC_BIQUAD_T_B_1

Biquad torque filter coefficient B_1.

BITS & NAME	TYPE & RESET	DESCRIPTION
[23:0] **BIQUAD_T_B_1**	RW, signed 0x000000	Biquad torque filter coefficient B_1.

0x0EA, Block 0: MCC_BIQUAD_T_B_2

Biquad torque filter coefficient B_2.

BITS & NAME	TYPE & RESET	DESCRIPTION
[23:0] **BIQUAD_T_B_2**	RW, signed 0x000000	Biquad torque filter coefficient B_2.

0x0EB, Block 0: MCC_BIQUAD_T_ENABLE

Biquad torque filter enable.

BITS & NAME	TYPE & RESET	DESCRIPTION
[0] **BIQUAD_T_ENABLE**	RW 0x0	Enable Biquad Torque Filter

0x100, Block 0: MCC_VELOCITY_CONFIG

Velocity meter configuration.

BITS & NAME	TYPE & RESET	DESCRIPTION
[14:12] **MOVING_AVRG_FILTER_SAMPLES**	RW 0x0	Number of velocity samples for moving average filter for VELOCITY_PER.
	0: AVRG_1	No additional filter
	1: AVRG_2	Average over 2 samples
	2: AVRG_3	Average over 3 samples
	3: AVRG_4	Average over 4 samples
	4: AVRG_5	Average over 5 samples
	5: AVRG_6	Average over 6 samples
	6: AVRG_7	Average over 7 samples
	7: AVRG_8	Average over 8 samples
[10:9] **METER_TYPE**	RW 0x0	Velocity meter type selection.
	0: VELOCITY_PER	Measurement of velocity by time measurement between position changes. Use this for slow velocities.
	1: VELOCITY_FREQ	Velocity Meter running at PWM frequency. Calculates the velocity using the difference of the angle in one clock cycle.
	2: VELOCITY_EXT	Measurement of velocity by software. Uses the value in the VELOCITY_EXT register.
[8] **METER_SYNC_PULSE**	RW 0x0	Velocity Meter Synchronization Pulse. Select either the start of each PWM cycle PWM_Z(ero) or the center with PWM_C(enter).
	0: PWM_Z	synchronize sampling to PWM_Z pulse
	1: PWM_C	synchronize sampling to PWM_C pulse

BITS & NAME	TYPE & RESET	DESCRIPTION
[7:0] **SELECTION**	RW 0x00	Selects the source of the rotor position for velocity measurement. Do not select 0x0 while PHI_E_SELECTION is using phi_e_hall with extrapolation enabled. In this case, use 0x5 instead.
	0: PHI_E — phi_e selected through PHI_E_SELECTION 1: PHI_E_EXT — phi_e_ext 2: PHI_E_RAMP — phi_e_ramp 3: PHI_E_ABN — phi_e_abn 4: RAMP_X_ACTUAL — ramp_X_actual 5: PHI_E_HAL — phi_e_hal 6: PHI_M_EXT — phi_m_ext 8: ABN_COUNT — abn_count 9: PHI_M_ABN — phi_m_abn 12: HALL_COUNT — hall_count	

0x101, Block 0: MCC_VELOCITY_SCALING

Scaling factor for velocity meter output. This value is only used when VELOCITY_FREQ in MCC_VELOCITY_CONFIG - METER_TYPE is selected.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **VELOCITY_SCALING**	RW, signed 0x28F6	Scaling factor for velocity meter output. This value is only used when VELOCITY_FREQ in MCC_VELOCITY_CONFIG - METER_TYPE is selected.

0x102, Block 0: MCC_V_MIN_POS_DEV_TIME_COUNTER_LIMIT

Velocity meter configuration. These values are only used when VELOCITY_PER in MCC_VELOCITY_CONFIG - METER_TYPE is selected.

BITS & NAME	TYPE & RESET	DESCRIPTION
[30:16] **V_MIN_POS_DEV**	RW, unsigned 0x001	Minimal position deviation to calculate velocity when using mode VELOCITY_PER in MCC_VELOCITY_CONFIG - METER_TYPE.
[15:0] **TIME_COUNTER_LIMIT**	RW, unsigned 0xFFF0	Counter limit for velocity minimum deviation functionality. Velocity is calculated only when the position deviation is greater than V_MIN_POS_DEV measured during TIME_COUNTER_LIMIT ticks of the system clock (40MHz).

0x103, Block 0: MCC_MAX_VEL_DEVIATION

Velocity deviation to generate tracking error flag.

BITS & NAME	TYPE & RESET	DESCRIPTION
[30:0] **MAX_VEL_DEVIATION**	RW, unsigned 0x0010000	Maximum allowed absolute velocity deviation/error. If velocity controller error exceeds this value, a tracking error flag is activated (STALL_IN_VEL_ERR).

0x120, Block 0: MCC_POSITION_CONFIG

Position configuration

BITS & NAME	TYPE & RESET	DESCRIPTION
[7:0] **SELECTION**	RW 0x00	Selects the source of the rotor position for position measurement. Do not select 0x0 while PHI_E_SELECTION is using phi_e_hall with extrapolation enabled. In this case, use 0x5 instead.
	0: PHI_E 1: PHI_E_EXT 2: PHI_E_RAMP 3: PHI_E_ABN 4: RAMP_X_ACTUAL 5: PHI_E_HAL 6: PHI_M_EXT 8: ABN_COUNT 9: PHI_M_ABN 12: HALL_COUNT	phi_e selected through PHI_E_SELECTION phi_e_ext phi_e_ramp phi_e_abn ramp_X_actual phi_e_hal phi_m_ext abn_count phi_m_abn hall_count

0x121, Block 0: MCC_MAX_POS_DEVIATION

Absolute position deviation to generate tracking error flag.

BITS & NAME	TYPE & RESET	DESCRIPTION
[30:0] **MAX_POS_DEVIATION**	RW, unsigned 0x0008000	Maximum allowed absolute position deviation/error. If position controller error exceeds this value, a tracking error flag is activated. (STALL_IN_POS_ERR).

0x140, Block 0: MCC_RAMPER_STATUS

Ramp status and switch event status flags

BITS & NAME	TYPE & RESET	DESCRIPTION
[17] **STALL_IN_POS_ERR**	R 0x0	Position deviation/error of position controller is set in case maximum allowed absolute postion limit (see MAX_POS_DEVIATION) is exceeded. If enabled, a ramp stop is activated.
[16] **STALL_IN_VEL_ERR**	R 0x0	Velocity value deviation/error of velocity controller is set in case maximum allowed absolute position limit (see MAX_VEL_DEVIATION) is exceeded. If enabled, a ramp stop is activated.
[15] **SECOND_MOVE**	RW, W1C 0x0	Status flag that the automatic ramp required moving back in the opposite direction, e.g.,. due to on-the-fly parameter change
[14] **T_ZEROWAIT_ACTIVE**	R 0x0	Indicates that T_ZEROWAIT is active after a motor stop. During this time, the motor is in standstill.

BITS & NAME	TYPE & RESET	DESCRIPTION
[13] **V_ZERO**	R 0x0	If active, actual velocity is 0.
[12] **POSITION_REACHED**	R 0x0	Indicates that the target position is reached. This flag is active as long XACTUAL matches XTARGET.
[11] **VELOCITY_REACHED**	R 0x0	Signals that the target velocity is reached. This flag is active as long VACTUAL matches VMAX.
[10] **EVENT_POS_REACHED**	RW, W1C 0x0	Signals that the target position has been reached (position_reached becoming active). (Flag and interrupt condition are cleared upon writing '1') This bit is ORed to the interrupt output signal.
[9] **EVENT_STOP_SG**	RW, W1C 0x0	Signals an active stop event. Writing '1' clears the stall condition and the motor may restart motion, unless the motion controller has been stopped. (Flag and interrupt condition are cleared upon writing '1') This bit is ORed to the interrupt output signal.
[8] **EVENT_STOP_H**	R 0x0	Signals an active stop home condition due to stop switch. Disabling the stop switch or the stop function clears the flag, but the motor continues motion. This bit is ORed to the interrupt output signal.
[7] **EVENT_STOP_R**	R 0x0	Signals an active stop right condition due to stop switch. The stop condition and the interrupt condition can be removed by commanding a move to the opposite direction. In soft_stop mode, the condition remains active until the motor has stopped motion into the direction of the stop switch. Disabling the stop switch or the stop function also clears the flag, but the motor continues motion. This bit is ORed to the interrupt output signal.
[6] **EVENT_STOP_L**	R 0x0	Signals an active stop left condition due to stop switch. The stop condition and the interrupt condition can be removed by commanding a move to the opposite direction. In soft_stop mode, the condition remains active until the motor has stopped motion into the direction of the stop switch. Disabling the stop switch or the stop function also clears the flag, but the motor continues motion. This bit is ORed to the interrupt output signal.
[5] **STATUS_LATCH_H**	RW, W1C 0x0	Latch home ready (enable position latching using SWITCH_MODE settings latch_h_active or latch_h_inactive) (Flag is cleared upon writing '1')
[4] **STATUS_LATCH_R**	RW, W1C 0x0	Latch right ready (enable position latching using SWITCH_MODE settings latch_r_active or latch_r_inactive) (Flag is cleared upon writing '1')
[3] **STATUS_LATCH_L**	RW, W1C 0x0	Latch left ready (enable position latching using SWITCH_MODE settings latch_l_active or latch_l_inactive) (Flag is cleared upon writing '1')
[2] **STATUS_STOP_H**	R 0x0	Home reference switch status
[1] **STATUS_STOP_R**	R 0x0	Right reference switch status
[0] **STATUS_STOP_L**	R 0x0	Left reference switch status

0x141, Block 0: MCC_RAMPER_A1

First acceleration value during EigthPoint ramp mode

BITS & NAME	TYPE & RESET	DESCRIPTION
[22:0] **RAMPER_A1**	RW, unsigned 0x10000	Acceleration value if RAMPER_V_START (resp. 0) < abs(RAMPER_V_ACTUAL) < RAMPER_V1.

0x142, Block 0: MCC_RAMPER_A2

Second acceleration value during EigthPoint ramp mode

BITS & NAME	TYPE & RESET	DESCRIPTION
[22:0] **RAMPER_A2**	RW, unsigned 0x10000	Acceleration value if RAMPER_V1 < abs(RAMPER_V_ACTUAL) < RAMPER_V2.

0x143, Block 0: MCC_RAMPER_A_MAX

Maximum acceleration value in top part of EigthPoint ramp

BITS & NAME	TYPE & RESET	DESCRIPTION
[22:0] **RAMPER_A_MAX**	RW, unsigned 0x10000	Acceleration value if RAMPER_V2 < abs(RAMPER_V_ACTUAL) < RAMPER_V_MAX (resp. RAMPER_V_TARGET).

0x144, Block 0: MCC_RAMPER_D1

Lower deceleration value during EigthPoint ramp mode

BITS & NAME	TYPE & RESET	DESCRIPTION
[22:0] **RAMPER_D1**	RW, unsigned 0x10000	Last deceleration value if RAMPER_V_STOP (resp. 0) < abs(RAMPER_V_ACTUAL) < RAMPER_V1. It is used during soft-stop or with ramp in position mode, not for regular ramp velocity mode.

0x145, Block 0: MCC_RAMPER_D2

Higher deceleration value in EigthPoint ramp

BITS & NAME	TYPE & RESET	DESCRIPTION
[22:0] **RAMPER_D2**	RW, unsigned 0x10000	Deceleration value if RAMPER_V1 < abs(RAMPER_V_ACTUAL) < RAMPER_V2. It is used during soft-stop or with ramp in position mode, not for regular ramp velocity mode.

0x146, Block 0: MCC_RAMPER_D_MAX

Deceleration in top part of EightPoint ramp

BITS & NAME	TYPE & RESET	DESCRIPTION
[22:0] **RAMPER_D_MAX**	RW, unsigned 0x10000	Deceleration value if RAMPER_V2 < abs(RAMPER_V_ACTUAL) < RAMPER_V_MAX (resp. RAMPER_V_TARGET). It is used during soft-stop or with ramp in position mode, not for regular ramp velocity mode.

0x147, Block 0: MCC_RAMPER_V_START

First velocity value during EigthPoint ramp mode

BITS & NAME	TYPE & RESET	DESCRIPTION
[22:0] **RAMPER_V_START**	RW, unsigned 0x00000	Start velocity of position ramp mode when V_ACTUAL = 0 or crossing 0 during motion. Not used during ramp velocity mode.

0x148, Block 0: MCC_RAMPER_V1

First velocity value for ac-/deceleration value switching during EigthPoint ramp mode

BITS & NAME	TYPE & RESET	DESCRIPTION
[26:0] **RAMPER_V1**	RW, unsigned 0x000000	Velocity value to switch from RAMPER_A1 to RAMPER_A2 during acceleration (ramp positioning mode) and deceleration phase (RAMPER_D1 and RAMPER_D2 during ramp velocity mode only).

0x149, Block 0: MCC_RAMPER_V2

Second velocity value for ac-/deceleration value switching during EigthPoint ramp mode

BITS & NAME	TYPE & RESET	DESCRIPTION
[26:0] **RAMPER_V2**	RW, unsigned 0x000000	Velocity value to switch to from RAMPER_A2 to RAMPER_A_MAX during acceleration (ramp positioning mode) and deceleration phase (RAMPER_D2 and RAMPER_D_MAX during ramp velocity mode only).

0x14A, Block 0: MCC_RAMPER_V_STOP

Stop Velocity in EightPoint Ramp.

BITS & NAME	TYPE & RESET	DESCRIPTION
[22:0] **RAMPER_V_STOP**	RW, unsigned 0x00100	Stop velocity in ramp in position ramp mode. Velocity used before reaching the target position. Not used during ramp velocity mode.

0x14B, Block 0: MCC_RAMPER_V_MAX

Maximum velocity value for positioning in EightPoint ramp

BITS & NAME	TYPE & RESET	DESCRIPTION
[26:0] **RAMPER_V_MAX**	RW, unsigned 0x7FFFFFF	Maximum velocity value of EightPoint Ramp in ramp positioning mode.

0x14C, Block 0: MCC_RAMPER_V_TARGET

Target velocity value in EightPoint ramp

BITS & NAME	TYPE & RESET	DESCRIPTION
[27:0] **RAMPER_V_TARGET**	RW, signed 0x0000000	Target Velocity in ramp velocity mode.

0x14D, Block 0: MCC_RAMPER_SWITCH_MODE

Ramper mode configuration bits

BITS & NAME	TYPE & RESET	DESCRIPTION
[19] **VELOCITY_OVERWRITE**	RW 0x0	If enabled velocity from overwrite input (PID_VELOCITY_TARGET) is written to ramp. (Velocity mode to velocity ramp switching)
[18] **STOP_ON_VEL_DEVIATION**	RW 0x0	Enables a hard stop during ramp mode if velocity tracking error is emerged.
[17] **STOP_ON_POS_DEVIATION**	RW 0x0	Enables a hard stop during ramp mode if position tracking error is emerged.
[16] **SW_HARD_STOP**	RW 0x0	Enables a hard stop during ramp mode in case any activated reference switch has been activated.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15] **SOFTSTOP_ENABLE**	RW 0x0	The soft stop mode always uses the deceleration ramp settings RAMPER_D_MAX, RAMPER_D2, RAMPER_V2, RAMPER_V1, RAMPER_D1, RAMPEER_V_STOP and T_ZEROWAIT parameters for stopping the motor. A stop occurs when the velocity sign matches the reference switch position (REFL for negative velocities, REFR for positive velocities) and the respective switch stop function is enabled. A hard stop also uses T_ZEROWAIT before the motor becomes released.
[14] **SG_STOP_ENABLE**	RW 0x0	Enables the following stop conditions: SW_HARD_STOP STOP_ON_POS_DEVIATION STOP_ON_VEL_DEVIATION
[12] **LATCH_H_INACTIVE**	RW 0x0	Activates position latching to RAMPER_X_ACTUAL_LATCH register in case home reference switch input REFH is deactivated. The active level is defined by STOP_H_POL.
[11] **LATCH_H_ACTIVE**	RW 0x0	Activates position latching to RAMPER_X_ACTUAL_LATCH register in case home reference switch input REFH is activated. Hint: Use LATCH_H_ACTIVE to detect any spurious stop event by reading STATUS_LATCH_H.
[10] **LATCH_R_INACTIVE**	RW 0x0	Activates position latching to RAMPER_X_ACTUAL_LATCH register in case right reference switch input REFR is deactivated. The active level is defined by STOP_R_POL.
[9] **LATCH_R_ACTIVE**	RW 0x0	Activates position latching to RAMPER_X_ACTUAL_LATCH register in case right reference switch input REFR is activated. Hint: Use LATCH_R_ACTIVE to detect any spurious stop event by reading STATUS_LATCH_R.
[8] **LATCH_L_INACTIVE**	RW 0x0	Activates position latching to RAMPER_X_ACTUAL_LATCH register in case left reference switch input REFL is deactivated. The active level is defined by STOP_L_POL.
[7] **LATCH_L_ACTIVE**	RW 0x0	Activates position latching to RAMPER_X_ACTUAL_LATCH register in case left reference switch input REFL is activated. Hint: Use LATCH_L_ACTIVE to detect any spurious stop event by reading STATUS_LATCH_L.
[6] **SWAP_LR**	RW 0x0	Swap the left and the right reference switch input REFL and REFR internally.
[5] **STOP_H_POL**	RW 0x0	Defines active polarity of the home reference switch input.
	0: NORMAL	non-inverted, high active: a high level on REFH stops the motor
	1: INVERTED	inverted, low active: a low level on REFH stops the motor
[4] **STOP_R_POL**	RW 0x0	Defines active polarity of the right reference switch input.
	0: NORMAL	non-inverted, high active: a high level on REFR stops the motor
	1: INVERTED	inverted, low active: a low level on REFR stops the motor

BITS & NAME	TYPE & RESET	DESCRIPTION
[3] **STOP_L_POL**	RW 0x0	Defines active polarity of the left reference switch input. 0: NORMAL non-inverted, high active: a high level on REFL stops the motor 1: INVERTED inverted, low active: a low level on REFL stops the motor
[2] **STOP_H_ENABLE**	RW 0x0	Enables automatic motor stop during active home reference switch input.
[1] **STOP_R_ENABLE**	RW 0x0	Enables automatic motor stop during active right reference switch input.
[0] **STOP_L_ENABLE**	RW 0x0	Enables automatic motor stop during active left reference switch input.

0x14E, Block 0: MCC_RAMPER_TIME_CONFIG

Ramper timing configuration.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **T_VMAX**	RW, unsigned 0x0000	Minimum time with constant velocity before starting deceleration (TVMAX × 12.8 us) to reduce jerk. Used whenever the sign of RAMPER_A_ACTUAL would change.
[15:0] **T_ZEROWAIT**	RW, unsigned 0x0000	Defines the waiting time after ramping down to zero velocity before next movement or direction inversion can start. (12.8 us × TZEROWAIT) Only in rampmode position.

0x14F, Block 0: MCC_RAMPER_A_ACTUAL

Actual ramp acceleration value

BITS & NAME	TYPE & RESET	DESCRIPTION
[23:0] **RAMPER_A_ACTUAL**	R, signed 0x000000	Actual ramp acceleration value

0x150, Block 0: MCC_RAMPER_X_ACTUAL

Actual multi-turn position of ramp controller.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **RAMPER_X_ACTUAL**	R, signed 0x00000000	Actual position output of rampm controller. When writing PID_POSITION_ACTUAL, this value is overwritten also.

0x151, Block 0: MCC_RAMPER_V_ACTUAL

Actual velocity of Ramp controller output.

BITS & NAME	TYPE & RESET	DESCRIPTION
[27:0] **RAMPER_V_ACTUAL**	R, signed 0x0000000	Actual velocity output value of ramp controller.

0x152, Block 0: MCC_RAMPER_X_TARGET

Multi-turn target position of Ramp controller.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **RAMPER_X_TARGET**	RW, signed 0x00000000	Target position of ramp controller. When writing PID_POSITION_ACTUAL, this value is overwritten also.

0x153, Block 0: MCC_RAMPER_PHI_E

PHI_E of ramp controller.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **RAMPER_PHI_E**	R, signed 0x0000	PHI_E calculated from RAMPER_X_ACTUAL×N_POLE_PAIRS + Offset.

0x154, Block 0: MCC_RAMPER_PHI_E_OFFSET

Offset value for calculation of RAMPER_PHI_E

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **RAMPER_PHI_E_OFFSET**	RW, signed 0x0000	Offset for PHI_E calculation.

0x155, Block 0: MCC_RAMPER_ACC_FF

Gain and shift factor for Acc. feedforward.

BITS & NAME	TYPE & RESET	DESCRIPTION
[18:16] **SHIFT**	RW 0x6	Shift Factor for acceleration feedforward. Result is used as offset for PIDIN_TORQUE_TARGET. (RAMPER_A_ACTUAL × GAIN) >> (SHIFT×4)
	0: SHIFT_0 0 1: SHIFT_4 4 2: SHIFT_8 8 3: SHIFT_12 12 4: SHIFT_16 16 5: SHIFT_20 20 6: SHIFT_24 24	
[15:0] **GAIN**	RW, unsigned 0x0000	Gain Factor for acceleration feedforward. Result is used as offset for PIDIN_TORQUE_TARGET. (RAMPER_A_ACTUAL × GAIN) >> (SHIFT×4)

0x156, Block 0: MCC_RAMPER_X_ACTUAL_LATCH

Latches RAMPER_X_ACTUAL on left or right switch or Encoder trigger.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **RAMPER_X_ACTUAL_LATCH**	R, signed 0x00000000	Latched X-Actual value at stop switch event.

0x157, Block 0: MCC_POSITION_ACTUAL_LATCH

Latches PID_POSITION_ACTUAL on left or right switch or Encoder trigger.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **POSITION_ACTUAL_LATCH**	R, signed 0x00000000	Actual feedback position latch at stop switch event.

0x160, Block 0: MCC_PRBS_AMPLITUDE

Set the amplitude of the PRBS signal used by some settings of MCC_MOTION_CONFIG -> MOTION_MODE. The output PRBS signal is either +PRBS_AMPLITUDE or -PRBS_AMPLITUDE. Setting this value to 0 resets the random sequence. After resetting the random sequence is always the same.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PRBS_AMPLITUDE**	RW, signed 0x00000000	Set the amplitude of the PRBS signal used by some settings of MCC_MOTION_CONFIG -> MOTION_MODE. The output PRBS signal is either +PRBS_AMPLITUDE or -PRBS_AMPLITUDE. Setting this value to 0 resets the random sequence. After resetting the random sequence is always the same.

0x161, Block 0: MCC_PRBS_DOWN_SAMPLING_RATIO

Set downsampling rate of PWM frequency to trigger new PRBS value generation.

BITS & NAME	TYPE & RESET	DESCRIPTION
[7:0] **PRBS_DOWN_SAMPLING_RATIO**	RW, unsigned 0x00	Set downsampling rate of PWM frequency to trigger new PRBS value generation.

0x180, Block 0: MCC_PID_CONFIG

PI controller configuration.

BITS & NAME	TYPE & RESET	DESCRIPTION
[30:24] **VEL_SMPL**	RW, unsigned 0x0	Downsampling factor for Velocity controller.
[22:16] **POS_SMPL**	RW, unsigned 0x0	Downsampling factor for Position controller.
[15:12] **VEL_SCALE**	RW, unsigned 0x8	Output right shift factor of the velocity controller.
[11:10] **POSITION_NORM_I**	RW 0x1	Normalization of I Factor of Position Control.
	0: SHIFT_8 1: SHIFT_16 2: SHIFT_24 3: SHIFT_32	Shift 8 bit right Shift 16 bit right Shift 24 bit right Shift 32 bit right
[9:8] **POSITION_NORM_P**	RW 0x1	Normalization of P Factor of Position Control.
	0: SHIFT_0 1: SHIFT_8 2: SHIFT_16 3: SHIFT_24	Shift 0 bit right Shift 8 bit right Shift 16 bit right Shift 24 bit right

BITS & NAME	TYPE & RESET	DESCRIPTION
[7:6] **VELOCITY_NORM_I**	RW 0x1	Normalization of I Factor of Velocity Control.
	0: SHIFT_8 1: SHIFT_16 2: SHIFT_24 3: SHIFT_32	Shift 8 bit right + VEL_SCALE Shift 16 bit right + VEL_SCALE Shift 24 bit right + VEL_SCALE Shift 32 bit right + VEL_SCALE
[5:4] **VELOCITY_NORM_P**	RW 0x1	Normalization of P Factor of Velocity Control.
	0: SHIFT_0 1: SHIFT_8 2: SHIFT_16 3: SHIFT_24	Shift 0 bit right + VEL_SCALE Shift 8 bit right + VEL_SCALE Shift 16 bit right + VEL_SCALE Shift 24 bit right + VEL_SCALE
[3] **CURRENT_NORM_I**	RW 0x0	Normalization of I Factor of Current Control. Used for torque and flux controller.
	0: SHIFT_8 1: SHIFT_16	Shift 8 bit right Shift 16 bit right
[2] **CURRENT_NORM_P**	RW 0x0	Normalization of P Factor of Current Control. Used for torque and flux controller.
	0: SHIFT_8 1: SHIFT_16	Shift 8 bit right Shift 16 bit right
[0] **KEEP_POS_TARGET**	RW 0x0	Do not overwrite position target on position actual write
	0: OVERWRITE 1: KEEP	Overwrite Target Keep Target

0x181, Block 0: MCC_PID_FLUX_COEFF

Configuration of the PI Flux controller gains.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **P**	RW, signed 0x0000	Proportional gain for the PI Flux controller.
[15:0] **I**	RW, signed 0x0000	Integral gain for the PI Flux controller.

0x182, Block 0: MCC_PID_TORQUE_COEFF

Configuration of the PI Torque controller gains.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **P**	RW, signed 0x0000	Proportional gain for the PI Torque controller.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **I**	RW, signed 0x0000	Integral gain for the PI Torque controller.

0x183, Block 0: MCC_PID_FIELDWEAK_COEFF

Configuration of the PI Fieldweakening controller gains.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **P**	RW, signed 0x0000	Proportional gain for the PI Fieldweakening controller.
[15:0] **I**	RW, signed 0x0000	Integral gain for the PI Fieldweakening controller.

0x184, Block 0: MCC_PID_U_S_MAX

Maximum voltage allowed for fieldweakening. Target value of field weakening PI controller.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **U_S_MAX**	RW, unsigned 0x7FFF	Maximum voltage allowed for fieldweakening. Target value of field weakening PI controller.

0x185, Block 0: MCC_PID_VELOCITY_COEFF

Configuration of the PI Velocity controller gains.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **P**	RW, signed 0x0000	Proportional gain for the PI Velocity controller.
[15:0] **I**	RW, signed 0x0000	Integral gain for the PI Velocity controller.

0x186, Block 0: MCC_PID_POSITION_COEFF

Configuration of the PI Position controller gains.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **P**	RW, signed 0x0000	Proportional gain for the PI Position controller.
[15:0] **I**	RW, signed 0x0000	Integral gain for the PI Position controller.

0x187, Block 0: MCC_PID_POSITION_TOLERANCE

Position controller ignores position errors smaller than PID_POSITION_TOLERANCE if EVENT_POS_REACHED and after (PID_POSITION_TOLERANCE_DELAY × PWM period).

BITS & NAME	TYPE & RESET	DESCRIPTION
[30:0] **PID_POSITION_TOLERANCE**	RW, unsigned 0x0000000	Position controller ignores position errors smaller than PID_POSITION_TOLERANCE if EVENT_POS_REACHED and after (PID_POSITION_TOLERANCE_DELAY × PWM period).

0x188, Block 0: MCC_PID_POSITION_TOLERANCE_DELAY

Number of PWM periods the abs(PID_POSITION_ERROR) must stay within PID_POSITION_TOLERANCE after EVENT_POS_REACHED to disable the controller.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **PID_POSITION_TOLERANCE_DELAY**	RW, unsigned 0x0000	Number of PWM periods the abs(PID_POSITION_ERROR) must stay within PID_POSITION_TOLERANCE after EVENT_POS_REACHED to disable the controller.

0x189, Block 0: MCC_PID_UQ_UD_LIMITS

Set maximum output voltage limit value.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **PID_UQ_UD_LIMITS**	RW, unsigned 0x5A81	Limit U_D to PID_UQ_UD_LIMITS and limit U_Q to sqrt(PID_UQ_UD_LIMITS^2-U_D^2). If this register is set higher than 16383 (0x3FFF) the limiter uses that value instead. If MCC_PWM_CONFIG -> SV_MODE is set to use third harmonic injection (SV_MODE not 0x0) for a 3-phase BLDC motor the internal maximum voltage limit is 18900 (0x49D4) instead.

0x18A, Block 0: MCC_PID_TORQUE_FLUX_LIMITS

Set maximum target absolute current for torque and flux PI controller.

BITS & NAME	TYPE & RESET	DESCRIPTION
[30:16] **PID_TORQUE_LIMIT**	RW, unsigned 0x7FFF	PID torque limit, limits the torque target value from torque target register and velocity controller output.
[14:0] **PID_FLUX_LIMIT**	RW, unsigned 0x7FFF	PID flux limit, limits the target values from Flux weakening controller and register.

0x18B, Block 0: MCC_PID_VELOCITY_LIMIT

Set maximum absolute velocity for velocity PI controller.

BITS & NAME	TYPE & RESET	DESCRIPTION
[30:0] **PID_VELOCITY_LIMIT**	RW, unsigned 0x7FFFFFFF	PID velocity limit, limits the velocity target value from velocity target register and position controller output.

0x18C, Block 0: MCC_PID_POSITION_LIMIT_LOW

Set minimum target position for position PI controller.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PID_POSITION_LIMIT_LOW**	RW, signed 0x80000001	Position limit low, programmable position barrier.

0x18D, Block 0: MCC_PID_POSITION_LIMIT_HIGH

Set maximum target position for position PI controller.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PID_POSITION_LIMIT_HIGH**	RW, signed 0x7FFFFFFF	Position limit high, programmable position barrier.

0x18E, Block 0: MCC_PID_TORQUE_FLUX_TARGET

PID target torque and target flux (for torque mode).

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **PID_TORQUE_TARGET**	RW, signed 0x0000	PID Target torque (for torque mode).
[15:0] **PID_FLUX_TARGET**	RW, signed 0x0000	PID Target flux (for torque mode).

0x18F, Block 0: MCC_PID_TORQUE_FLUX_OFFSET

PID torque and flux offset.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **PID_TORQUE_OFFSET**	RW, signed 0x0000	Torque offset for feed forward control.
[15:0] **PID_FLUX_OFFSET**	RW, signed 0x0000	Flux offset for feed forward control.

0x190, Block 0: MCC_PID_VELOCITY_TARGET

PID Target velocity (for velocity mode).

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PID_VELOCITY_TARGET**	RW, signed 0x00000000	PID Target velocity (for velocity mode).

0x191, Block 0: MCC_PID_VELOCITY_OFFSET

PID velocity offset for feed forward control.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PID_VELOCITY_OFFSET**	RW, signed 0x00000000	PID velocity offset for feed forward control.

0x192, Block 0: MCC_PID_POSITION_TARGET

Target position register (for position mode).

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PID_POSITION_TARGET**	RW, signed 0x00000000	Target position register (for position mode).

0x193, Block 0: MCC_PID_TORQUE_FLUX_ACTUAL

PID actual torque and flux.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **PID_TORQUE_ACTUAL**	R, signed 0x0000	PID actual torque.
[15:0] **PID_FLUX_ACTUAL**	R, signed 0x0000	PID actual flux.

0x194, Block 0: MCC_PID_VELOCITY_ACTUAL

PID actual velocity.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PID_VELOCITY_ACTUAL**	R, signed 0x00000000	PID actual velocity.

0x195, Block 0: MCC_PID_POSITION_ACTUAL

PID actual position.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PID_POSITION_ACTUAL**	RW, signed 0x00000000	Actual multi turn position for positioning. WRITE on PID_POSITION_ACTUAL writes same value into PID_POSITION_TARGET to avoid unwanted move. Use offset to compensate.

0x196, Block 0: MCC_PID_POSITION_ACTUAL_OFFSET

Offset for actual position.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PID_POSITION_ACTUAL_OFFSET**	RW, signed 0x00000000	Offset for actual position.

0x197, Block 0: MCC_PID_TORQUE_ERROR

PID torque error.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **PID_TORQUE_ERROR**	R, signed 0x0000	PID torque error.

0x198, Block 0: MCC_PID_FLUX_ERROR

PID flux error.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **PID_FLUX_ERROR**	R, signed 0x0000	PID flux error.

0x199, Block 0: MCC_PID_VELOCITY_ERROR

PID velocity error.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PID_VELOCITY_ERROR**	R, signed 0x00000000	PID velocity error.

0x19A, Block 0: MCC_PID_POSITION_ERROR

PID position error.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PID_POSITION_ERROR**	R, signed 0x00000000	PID position error.

0x19B, Block 0: MCC_PID_TORQUE_INTEGRATOR

PID torque Integrator.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PID_TORQUE_INTEGRATOR**	RW, signed 0x00000000	PID torque Integrator.

0x19C, Block 0: MCC_PID_FLUX_INTEGRATOR

PID flux Integrator.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PID_FLUX_INTEGRATOR**	RW, signed 0x00000000	PID flux Integrator.

0x19D, Block 0: MCC_PID_VELOCITY_INTEGRATOR

PID velocity Integrator.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PID_VELOCITY_INTEGRATOR**	RW, signed 0x00000000	PID velocity Integrator.

0x19E, Block 0: MCC_PID_POSITION_INTEGRATOR

PID position Integrator.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PID_POSITION_INTEGRATOR**	RW, signed 0x00000000	PID position Integrator.

0x1A0, Block 0: MCC_PIDIN_TORQUE_FLUX_TARGET

PID target torque and target flux for readback.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **PIDIN_TORQUE_TARGET**	R, signed 0x0000	torque target before any filtering / limiting.
[15:0] **PIDIN_FLUX_TARGET**	R, signed 0x0000	flux target before any filtering / limiting

0x1A1, Block 0: MCC_PIDIN_VELOCITY_TARGET

PID target velocity for readback.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PIDIN_VELOCITY_TARGET**	R, signed 0x00000000	velocity target before any filtering / limiting

0x1A2, Block 0: MCC_PIDIN_POSITION_TARGET

PID target position for readback.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PIDIN_POSITION_TARGET**	R, signed 0x00000000	position target before any filtering / limiting

0x1A3, Block 0: MCC_PIDIN_TORQUE_FLUX_TARGET_LIMITED

PID target torque and target flux after PID_TORQUE_FLUX_LIMITS applied.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **PIDIN_TORQUE_TARGET_LIMITED**	R, signed 0x0000	torque target after limiter
[15:0] **PIDIN_FLUX_TARGET_LIMITED**	R, signed 0x0000	flux target after limiter

0x1A4, Block 0: MCC_PIDIN_VELOCITY_TARGET_LIMITED

PID target velocity after PID_VELOCITY_LIMIT applied.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PIDIN_VELOCITY_TARGET_LIMITED**	R, signed 0x00000000	velocity target after limiter

0x1A5, Block 0: MCC_PIDIN_POSITION_TARGET_LIMITED

PID target position after PID_POSITION_LIMIT_LOW and PID_POSITION_LIMIT_HIGH applied.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **PIDIN_POSITION_TARGET_LIMITED**	R, signed 0x00000000	Position target after limiter

0x1A6, Block 0: MCC_FOC_IBETA_IALPHA

Interim result of the FOC, IALPHA, and IBETA term.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **IBETA**	R, signed 0x0000	Interim result of the FOC, IBETA term.
[15:0] **IALPHA**	R, signed 0x0000	Interim result of the FOC, IALPHA term.

0x1A7, Block 0: MCC_FOC_IQ_ID

Interim result of the FOC, ID, and IQ term.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **IQ**	R, signed 0x0000	Interim result of the FOC, IQ term.
[15:0] **ID**	R, signed 0x0000	Interim result of the FOC, ID term.

0x1A8, Block 0: MCC_FOC_UQ_UD

Interim result of the FOC, UD, and UQ term.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **UQ**	R, signed 0x0000	Interim result of the FOC, UQ term.
[15:0] **UD**	R, signed 0x0000	Interim result of the FOC, UD term.

0x1A9, Block 0: MCC_FOC_UQ_UD_LIMITED

Interim result of the FOC, UD, and UQ term. After PID_UQ_UD_LIMITS applied.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **UQ**	R, signed 0x0000	Interim result of the FOC, UQ term limited.
[15:0] **UD**	R, signed 0x0000	Interim result of the FOC, UD term limited.

0x1AA, Block 0: MCC_FOC_UBETA_UALPHA

Interim result of the FOC, UALPHA, and UBETA term.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **UBETA**	R, signed 0x0000	Interim result of the FOC, UBETA term.
[15:0] **UALPHA**	R, signed 0x0000	Interim result of the FOC, UALPHA term.

0x1AB, Block 0: MCC_FOC_UWY_UUX

Interim result of the FOC, UUX, and UWY term.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **UWY**	R, signed 0x0000	Interim result of the FOC, UWY term.
[15:0] **UUX**	R, signed 0x0000	Interim result of the FOC, UUX term.

0x1AC, Block 0: MCC_FOC_UV

Interim result of the FOC, UV term.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **UV**	R, signed 0x0000	Interim result of the FOC, UV term.

0x1AD, Block 0: MCC_PWM_VX2_UX1

Interim result PWM duty cycle UX1 and VX2.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **VX2**	R, unsigned 0x0000	Interim result PWM VX2.
[15:0] **UX1**	R, unsigned 0x0000	Interim result PWM UX1.

0x1AE, Block 0: MCC_PWM_Y2_WY1

Interim result PWM duty cycle WY1 and Y2.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **Y2**	R, unsigned 0x0000	Interim result PWM Y2.
[15:0] **WY1**	R, unsigned 0x0000	Interim result PWM WY1.

0x1AF, Block 0: MCC_VELOCITY_FRQ

Actual velocity measured by fixed frequency sampling.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **VELOCITY_FRQ**	R, signed 0x00000000	Actual velocity measured by fixed frequency sampling.

0x1B0, Block 0: MCC_VELOCITY_PER

Actual velocity measured by period measurement.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **VELOCITY_PER**	R, signed 0x00000000	Actual velocity measured by period measurement.

0x1C0, Block 0: MCC_U_S_ACTUAL_I_S_ACTUAL

Actual motor voltage and current.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **U_S_ACTUAL**	R, unsigned 0x0000	Actual Motor voltage. U_S_ACTUAL = sqrt(UD^2 + UQ^2)
[15:0] **I_S_ACTUAL**	R, unsigned 0x0000	Actual Motor current. I_S_ACTUAL = sqrt(ID^2 + IQ^2)

0x1C1, Block 0: MCC_P_MOTOR

Actual Power applied to motor (P = U×I).

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **P_MOTOR**	R, unsigned 0x00000000	Actual Power applied to motor. P = U_S_ACTUAL × I_S_ACTUAL

0x1C2, Block 0: MCC_INPUTS_RAW

Raw input signals PWM_IN, DIR, STP, Digital Hall Inputs, and digital ABN encoder inputs as raw signals for direct read out for system setup and validation during development phase.

BITS & NAME	TYPE & RESET	DESCRIPTION
[22] **HALL_W_FILT**	R 0x0	Hall signal W after filter and reordering.
[21] **HALL_V_FILT**	R 0x0	Hall signal V after filter and reordering.
[20] **HALL_U_FILT**	R 0x0	Hall signal U after filter and reordering.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15] **ENI**	R 0x0	DRV_ENABLE pin value.
[14] **REF_SW_H**	R 0x0	Home reference switch value. Directly from pin.
[13] **REF_SW_L**	R 0x0	Left reference switch value. Directly from pin.
[12] **REF_SW_R**	R 0x0	Right reference switch value. Directly from pin.
[10] **HALL_W**	R 0x0	Hall signal W. Directly from pin.
[9] **HALL_V**	R 0x0	Hall signal V. Directly from pin.
[8] **HALL_U**	R 0x0	Hall signal U. Directly from pin.
[2] **ENC_N**	R 0x0	Encoder signal N. Directly from pin.
[1] **ENC_B**	R 0x0	Encoder signal B. Directly from pin.
[0] **ENC_A**	R 0x0	Encoder signal A. Directly from pin.

0x1C3, Block 0: MCC_OUTPUTS_RAW

Raw output signals for each PWM channel high- and low-side.

BITS & NAME	TYPE & RESET	DESCRIPTION
[7] **PWM_Y2_H**	R 0x0	Value of PWM phase Y2 high side.
[6] **PWM_Y2_L**	R 0x0	Value of PWM phase Y2 low side.
[5] **PWM_WY1_H**	R 0x0	Value of PWM phase WY1 high side.

BITS & NAME	TYPE & RESET	DESCRIPTION
[4] **PWM_WY1_L**	R 0x0	Value of PWM phase WY1 low side.
[3] **PWM_VX2_H**	R 0x0	Value of PWM phase VX2 high side.
[2] **PWM_VX2_L**	R 0x0	Value of PWM phase VX2 low side.
[1] **PWM_UX1_H**	R 0x0	Value of PWM phase UX1 high side.
[0] **PWM_UX1_L**	R 0x0	Value of PWM phase UX1 low side.

0x1C4, Block 0: MCC_STATUS_FLAGS

Status Flag BitVector Vector, individual status bits are set to '1' on error condition pulse '1' and remain '1' until they are cleared through register access; error condition '1' have priority over register write to clear access.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31] **ENI**	RW, W1C 0x0	Change on DRV_ENABLE pin sets this bit to 1.
[28] **ENC_N**	RW, W1C 0x0	Filtered encoder signal N = 1 sets this bit to 1.
[26] **ADC_I_CLIPPED**	RW, W1C 0x0	Set to 1 if ADC current measurement is clipped.
[23] **POSITION_REACHED**	RW, W1C 0x0	Ramper position reached. Only when motion controller is enabled and in use.
[22] **REF_SW_H**	RW, W1C 0x0	Set to 1 if home reference switch is 1. Directly connected to reference switch pin.
[21] **REF_SW_R**	RW, W1C 0x0	Set to 1 if right reference switch is 1. Directly connected to reference switch pin.
[20] **REF_SW_L**	RW, W1C 0x0	Set to 1 if left reference switch is 1. Directly connected to reference switch pin.
[17] **SHORT**	RW, W1C 0x0	Set to 1 if any of the HS_FAULT or LS_FAULT of the MCC_GDRV_FAULT register are triggered.

BITS & NAME	TYPE & RESET	DESCRIPTION
[14] **PID_FW_OUTPUT_LIMIT**	RW, W1C 0x0	Set to 1 if field weakening PI controller output limit is active. Limit values are 0 and -PID_FLUX_LIMIT.
[13] **VELOCITY_TRACKING_ERROR**	RW, W1C 0x0	Set to 1 if PI velocity controller error value is higher than MAX_VEL_DEVIATION.
[12] **POSITION_TRACKING_ERROR**	RW, W1C 0x0	Set to 1 if PI position controller error value is higher than MAX_POS_DEVIATION.
[11] **HALL_ERROR**	RW, W1C 0x0	1 on hall_vector=000 or hall_vector=111
[9] **PWM_SWITCH_LIMIT_ACTIVE**	RW, W1C 0x0	One current is calculated with kirchhoff for 3 phase BLDC motor.
[8] **IPARK_VOLTLIM_LIMIT_U**	RW, W1C 0x0	Ud or Uq are limited by PID_UQ_UD_LIMITS or internal maximum value
[7] **PID_IQ_OUTPUT_LIMIT**	RW, W1C 0x0	Set to 1 if PI torque controller output limiter is active. Output value is limited by PID_UQ_UD_LIMITS.
[6] **PID_IQ_TARGET_LIMIT**	RW, W1C 0x0	Set to 1 if PI torque controller target value limiter is active. Target value is limited by PID_TORQUE_LIMIT.
[5] **PID_ID_OUTPUT_LIMIT**	RW, W1C 0x0	Set to 1 if PI flux controller output limiter is active. Output value is limited by PID_UQ_UD_LIMITS.
[4] **PID_ID_TARGET_LIMIT**	RW, W1C 0x0	Set to 1 if PI flux controller target value limiter is active. Target value is limited by PID_FLUX_LIMIT.
[3] **PID_V_OUTPUT_LIMIT**	RW, W1C 0x0	Set to 1 if PI velocity controller output limiter is active. Output value is limited by PID_TORQUE_LIMIT.
[2] **PID_V_TARGET_LIMIT**	RW, W1C 0x0	Set to 1 if PI velocity controller target value limiter is active. Target value is limited by PID_VELOCITY_LIMIT.
[1] **PID_X_OUTPUT_LIMIT**	RW, W1C 0x0	Set to 1 if PI position controller output limiter is active. Output value is limited by PID_VELOCITY_LIMIT.
[0] **PID_X_TARGET_LIMIT**	RW, W1C 0x0	Set to 1 if PI position controller target value limiter is active. Target value is limited by PID_POSITION_LIMIT_HIGH or PID_POSITION_LIMIT_LOW.

0x1E3, Block 0: MCC_GDRV_HW

Controls the general gate driver setup.

BITS & NAME	TYPE & RESET	DESCRIPTION
[28] **HS_AS_LS_Y2**	RW 0x0	Treats the FET connected to Y2 HS as being on the low side instead of the high side, set to 1 if the FETs on Y2 are operated independently and the HS FET is between the load and ground instead of the load and the supply voltage.
[25] **BIAS_EN**	RW 0x0	Enables the internal analog bias voltage for the gatedriver. Enable before using any gatedriver functionality
[24] **CHARGEPUMP_EN**	RW 0x0	Gatedriver chargepump, enable to charge the bootstrap capacitors independently from the PWM switching events. Recommended to be turned on to allow for 100% duty cycle.
[11] **BST_SW_CP_EN**	RW 0x0	Internal chargepump for an active switch to charge the bootstrap capacitors. If turned off the caps are charged by a diode, thus this should be turned on to improve efficiency.
	0: OFF 1: ON	Chargepump Disabled Chargepump Enabled
[10:8] **BST_ILIM_MAX**	RW 0x0	Defines the maximum BST cap charge current.
	0: LIM_45MA 1: LIM_91MA 2: LIM_141MA 3: LIM_191MA 4: LIM_267MA 5: LIM_292MA 6: LIM_341MA 7: LIM_391MA	45mA 91mA 141mA 191mA 267mA 292mA 341mA 391mA
[7] **VS_UVLO_CMP_EN**	RW 0x0	Turn on the VS comparator which checks for VS undervoltage conditions.
[6] **VDRV_UVLO_CMP_EN**	RW 0x0	Turn on the VDRV comparator which checks for VDRV undervoltage conditions.
[5] **HS_OCP_CMP_EN**	RW 0x0	Turn on the high-side overcurrent comparator. Required to be set to detect OCP events.
[4] **LS_OCP_CMP_EN**	RW 0x0	Turn on the low-side overcurrent comparator. Required to be set to detect OCP events.
[3] **BRIDGE_ENABLE_Y2**	RW 0x0	Enable Bridge Y2
	0: DISABLED 1: ENABLED	Both high- and low-side are not driven and pulled down. Bridge is driven according to the PWM configuration
[2] **BRIDGE_ENABLE_W**	RW 0x0	Enable Bridge WY1
	0: DISABLED 1: ENABLED	Both high- and low-side are not driven and pulled down. Bridge is driven according to the PWM configuration

BITS & NAME	TYPE & RESET	DESCRIPTION
[1] **BRIDGE_ENABLE_V**	RW 0x0	Enable Bridge VX2
	0: DISABLED 1: ENABLED	Both high- and low-side are not driven and pulled down. Bridge is driven according to the PWM configuration
[0] **BRIDGE_ENABLE_U**	RW 0x0	Enable Bridge UX1
	0: DISABLED 1: ENABLED	Both high- and low-side are not driven and pulled down. Bridge is driven according to the PWM configuration

0x1E4, Block 0: MCC_GDRV_CFG

Controls the gate driver behavior and current settings.

BITS & NAME	TYPE & RESET	DESCRIPTION
[23:20] **VS_UVLO_LVL**	RW 0x0	Sets the VS undervoltage level.
	0: VSUVLO_44 1: VSUVLO_46 2: VSUVLO_48 3: VSUVLO_50 4: VSUVLO_52 5: VSUVLO_54 6: VSUVLO_56 7: VSUVLO_58 8: VSUVLO_60 9: VSUVLO_63 10: VSUVLO_66 11: VSUVLO_69 12: VSUVLO_72 13: VSUVLO_75 14: VSUVLO_78 15: VSUVLO_81	4.4V 4.6V 4.8V 5.0V 5.2V 5.4V 5.6V 5.8V 6.0V 6.3V 6.6V 6.9V 7.2V 7.5V 7.8V 8.1V
[17] **ADAPTIVE_MODE_Y2**	RW 0x1	If enabled, the discharge cycle of the high-/low-side FET is shortened monitoring the gate voltage of it. If enabled, the T_DRIVE_SINK acts as an upper boundary instead of a fixed time.
[16] **ADAPTIVE_MODE_UVW**	RW 0x1	If enabled, the discharge cycle of the high-/low-side FET is shortened monitoring the gate voltage of it. If enabled, the T_DRIVE_SINK acts as an upper boundary instead of a fixed time.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:12] **IGATE_SOURCE_Y2**	RW 0x0	Limits the maximum source current that is used to charge the MOSFET gate.
	0: SOURCE_25MA 25mA 1: SOURCE_50MA 50mA 2: SOURCE_80MA 80mA 3: SOURCE_105MA 105mA 4: SOURCE_135MA 135mA 5: SOURCE_160MA 160mA 6: SOURCE_190MA 190mA 7: SOURCE_215MA 215mA 8: SOURCE_290MA 290mA 9: SOURCE_360MA 360mA 10: SOURCE_430MA 430mA 11: SOURCE_500MA 500mA 12: SOURCE_625MA 625mA 13: SOURCE_755MA 755mA 14: SOURCE_885MA 885mA 15: SOURCE_1000MA 1000mA	
[11:8] **IGATE_SINK_Y2**	RW 0x0	Limits the maximum sink current that is used to discharge the MOSFET gate.
	0: SINK_50MA 50mA 1: SINK_100MA 100mA 2: SINK_160MA 160mA 3: SINK_210MA 210mA 4: SINK_270MA 270mA 5: SINK_320MA 320mA 6: SINK_380MA 380mA 7: SINK_430MA 430mA 8: SINK_580MA 580mA 9: SINK_720MA 720mA 10: SINK_860MA 860mA 11: SINK_1000MA 1000mA 12: SINK_1250MA 1250mA 13: SINK_1510MA 1510mA 14: SINK_1770MA 1770mA 15: SINK_2000MA 2000mA	
[7:4] **IGATE_SOURCE_UVW**	RW 0x0	Limits the maximum source current that is used to charge the MOSFET gate.
	0: SOURCE_25MA 25mA 1: SOURCE_50MA 50mA 2: SOURCE_80MA 80mA 3: SOURCE_105MA 105mA 4: SOURCE_135MA 135mA 5: SOURCE_160MA 160mA 6: SOURCE_190MA 190 mA 7: SOURCE_215MA 215mA 8: SOURCE_290MA 290mA 9: SOURCE_360MA 360mA 10: SOURCE_430MA 430mA 11: SOURCE_500MA 500mA 12: SOURCE_625MA 625mA 13: SOURCE_755MA 755mA 14: SOURCE_885MA 885mA 15: SOURCE_1000MA 1000mA	

BITS & NAME	TYPE & RESET	DESCRIPTION
[3:0] **IGATE_SINK_UVW**	RW 0x0	Limits the maximum sink current that is used to discharge the MOSFET gate.
	0: SINK_50MA 50mA 1: SINK_100MA 100mA 2: SINK_160MA 160mA 3: SINK_210MA 210mA 4: SINK_270MA 270mA 5: SINK_320MA 320mA 6: SINK_380MA 380mA 7: SINK_430MA 430mA 8: SINK_580MA 580mA 9: SINK_720MA 720mA 10: SINK_860MA 860mA 11: SINK_1000MA 1000mA 12: SINK_1250MA 1250mA 13: SINK_1510MA 1510mA 14: SINK_1770MA 1770mA 15: SINK_2000MA 2000mA	

0x1E9, Block 0: MCC_GDRV_TIMING

Sets the T_DRIVE sink and source times for all channels.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:24] **T_DRIVE_SOURCE_Y2**	RW, unsigned 0xFF	Charge time of the MOSFET. During this time, the full gate drive current is applied. The applied time is defined as "(1s / PWM_CLK) × (2 × TDRIVE + 5)"
[23:16] **T_DRIVE_SINK_Y2**	RW, unsigned 0xFF	Discharge time of the MOSFET. During this time, the full gate drive current is applied. The applied time is defined as "(1s / PWM_CLK) × (2 × TDRIVE + 5)".
[15:8] **T_DRIVE_SOURCE_UVW**	RW, unsigned 0xFF	Charge time of the MOSFET. During this time, the full gate drive current is applied. The applied time is defined as "(1s / PWM_CLK) × (2 × TDRIVE + 5)"
[7:0] **T_DRIVE_SINK_UVW**	RW, unsigned 0xFF	Discharge time of the MOSFET. During this time, the full gate drive current is applied. The applied time is defined as "(1s / PWM_CLK) × (2 × TDRIVE + 5)".

0x1EA, Block 0: MCC_GDRV_BBM

Controls the BBM_L and BBM_H times for all channels.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:24] **BBM_H_Y2**	RW, unsigned 0x14	Break Before Make time "(1s / PWM_CLK) × (BBM + 1)" for high-side MOSFET gate control. Applies before switching from low to high. Works with both external and internal gate driver, however with internal gate driver it should usually be set to zero in favor of T_DRIVE.

BITS & NAME	TYPE & RESET	DESCRIPTION
[23:16] **BBM_L_Y2**	RW, unsigned 0x14	Break Before Make time "(1s / PWM_CLK) × (BBM + 1)" for low-side MOSFET gate control. Applies before switching from high to low. Works with both external and internal gate driver, however with internal gate driver it should usually be set to zero in favor of T_DRIVE.
[15:8] **BBM_H_UVW**	RW, unsigned 0x14	Break Before Make time "(1s / PWM_CLK) × (BBM + 1)" for high-side MOSFET gate control. Applies before switching from low to high. Works with both external and internal gate driver, however with internal gate driver it should usually be set to zero in favor of T_DRIVE.
[7:0] **BBM_L_UVW**	RW, unsigned 0x14	Break Before Make time "(1s / PWM_CLK) × (BBM + 1)" for low-side MOSFET gate control. Applies before switching from high to low. Works with both external and internal gate driver, however with internal gate driver it should usually be set to zero in favor of T_DRIVE.

0x1EB, Block 0: MCC_GDRV_PROT

Contains the general and VGS related protection settings.

BITS & NAME	TYPE & RESET	DESCRIPTION
[28] **TERM_PWM_ON_SHORT**	RW 0x0	If a fault on any phase occur, force the PWM on phases to low.
	0: OFF 1: ON	Keep PWM running for other phases Terminate PWM for other phases
[23:22] **HS_RETRIES_Y2**	RW 0x0	Specifies the number of retries that occurs after any fault has been detected. A retry happens every PWM cycle.
	0: OFF 1: ONE 2: TWO 3: THREE	No Retries 1 Retry 2 Retries 3 Retries
[21:20] **LS_RETRIES_Y2**	RW 0x0	Specifies the number of retries that occurs after any fault has been detected. A retry happens every PWM cycle.
	0: OFF 1: ONE 2: TWO 3: THREE	No Retries 1 Retry 2 Retries 3 Retries
[19:18] **HS_RETRIES_UVW**	RW 0x0	Specifies the number of retries that occurs after any fault has been detected. A retry happens every PWM cycle.
	0: OFF 1: ONE 2: TWO 3: THREE	No Retries 1 Retry 2 Retries 3 Retries
[17:16] **LS_RETRIES_UVW**	RW 0x0	Specifies the number of retries that occurs after any fault has been detected. A retry happens every PWM cycle.
	0: OFF 1: ONE 2: TWO 3: THREE	No Retries 1 Retry 2 Retries 3 Retries

BITS & NAME	TYPE & RESET	DESCRIPTION
[13:12] **VGS_BLANKING_Y2**	RW 0x0	VGS Short Blanking Time
	0: BLK_OFF Off 1: BLK_250NS 0.25us 2: BLK_500NS 0.5us 3: BLK_1000NS 1us	
[10:8] **VGS_DEGLITCH_Y2**	RW 0x0	VGS Short Deglitch Time
	0: DEG_OFF Off 1: DEG_250NS 0.25us 2: DEG_500NS 0.5us 3: DEG_1000NS 1us 4: DEG_2000NS 2us 5: DEG_4000NS 4us 6: DEG_6000NS 6us 7: DEG_8000NS 8us	
[5:4] **VGS_BLANKING_UVW**	RW 0x0	VGS Short Blanking Time
	0: BLK_OFF off 1: BLK_250NS 0.25us 2: BLK_500NS 0.5us 3: BLK_1000NS 1us	
[2:0] **VGS_DEGLITCH_UVW**	RW 0x0	VGS Short Deglitch Time
	0: DEG_OFF off 1: DEG_250NS 0.25us 2: DEG_500NS 0.5us 3: DEG_1000NS 1us 4: DEG_2000NS 2us 5: DEG_4000NS 4us 6: DEG_6000NS 6us 7: DEG_8000NS 8us	

0x1EC, Block 0: MCC_GDRV_OCP_UVW

Configures the overcurrent protection for phases U, V and W.

BITS & NAME	TYPE & RESET	DESCRIPTION
[27:24] **HS_OCP_THRES_UVW**	RW 0x0	Threshold of the overcurrent protection.
	0: THRES_63MV 63mV 1: THRES_125MV 125mV 2: THRES_187MV 187mV 3: THRES_248MV 248mV 4: THRES_312MV 312mV 5: THRES_374MV 374mV 6: THRES_434MV 434mV 7: THRES_504MV 504mV 8: THRES_705MV 705mV 9: THRES_940MV 940mV 10: THRES_1180MV 1180mV 11: THRES_1410MV 1410mV 12: THRES_1650MV 1650mV 13: THRES_1880MV 1880mV 14: THRES_2110MV 2110mV 15: THRES_2350MV 2350mV	
[22:20] **HS_OCP_BLANKING_UVW**	RW 0x0	OCP Blanking Time
	0: BLK_OFF Off 1: BLK_250NS 0.25us 2: BLK_500NS 0.5us 3: BLK_1000NS 1us 4: BLK_2000NS 2us 5: BLK_4000NS 4us 6: BLK_6000NS 6us 7: BLK_8000NS 8us	
[18:16] **HS_OCP_DEGLITCH_UVW**	RW 0x0	OCP Deglitch Time
	0: DEG_OFF Off 1: DEG_250NS 0.25us 2: DEG_500NS 0.5us 3: DEG_1000NS 1us 4: DEG_2000NS 2us 5: DEG_4000NS 4us 6: DEG_6000NS 6us 7: DEG_8000NS 8us	
[15] **LS_OCP_USE_VDS_UVW**	RW 0x0	Switches between shunt and RDSon measurement

BITS & NAME	TYPE & RESET	DESCRIPTION
[11:8] **LS_OCP_THRES_UVW**	RW 0x0	Threshold of the low-side overcurrent protection.
	0: THRES_80_63MV 80mV (SHUNT), 63mV (VDS) 1: THRES_165_125MV 165mV (SHUNT), 125mV (VDS) 2: THRES_250_187MV 250mV (SHUNT), 187mV (VDS) 3: THRES_330_248MV 330mV (SHUNT), 248mV (VDS) 4: THRES_415_312MV 415mV (SHUNT), 312mV (VDS) 5: THRES_500_374MV 500mV (SHUNT), 374mV (VDS) 6: THRES_582_434MV 582mV (SHUNT), 434mV (VDS) 7: THRES_660_504MV 660mV (SHUNT), 504mV (VDS) 8: THRES_125_705MV 125mV (SHUNT), 705mV (VDS) 9: THRES_250_940MV 250mV (SHUNT), 940mV (VDS) 10: THRES_375_1180MV 375mV (SHUNT), 1180mV (VDS) 11: THRES_500_1410MV 500mV (SHUNT), 1410mV (VDS) 12: THRES_625_1650MV 625mV (SHUNT), 1650mV (VDS) 13: THRES_750_1880MV 750mV (SHUNT), 1880mV (VDS) 14: THRES_873_2110MV 873mV (SHUNT), 2110mV (VDS) 15: THRES_1000_2350MV 1000mV (SHUNT), 2350mV (VDS)	
[6:4] **LS_OCP_BLANKING_UVW**	RW 0x0	OCP Blanking Time
	0: BLK_OFF Off 1: BLK_250NS 0.25us 2: BLK_500NS 0.5us 3: BLK_1000NS 1us 4: BLK_2000NS 2us 5: BLK_4000NS 4us 6: BLK_6000NS 6us 7: BLK_8000NS 8us	
[2:0] **LS_OCP_DEGLITCH_UVW**	RW 0x0	OCP Deglitch Time
	0: DEG_OFF Off 1: DEG_250NS 0.25us 2: DEG_500NS 0.5us 3: DEG_1000NS 1us 4: DEG_2000NS 2us 5: DEG_4000NS 4us 6: DEG_6000NS 6us 7: DEG_8000NS 8us	

0x1ED, Block 0: MCC_GDRV_OCP_Y2

Configures the overcurrent protection for phase Y2.

BITS & NAME	TYPE & RESET	DESCRIPTION
[27:24] **HS_OCP_THRES_Y2**	RW 0x0	Threshold of the overcurrent protection.
	0: THRES_63MV 63mV 1: THRES_125MV 125mV 2: THRES_187MV 187mV 3: THRES_248MV 248mV 4: THRES_312MV 312mV 5: THRES_374MV 374mV 6: THRES_434MV 434mV 7: THRES_504MV 504mV 8: THRES_705MV 705mV 9: THRES_940MV 940mV 10: THRES_1180MV 1180mV 11: THRES_1410MV 1410mV 12: THRES_1650MV 1650mV 13: THRES_1880MV 1880mV 14: THRES_2110MV 2110mV 15: THRES_2350MV 2350mV	
[22:20] **HS_OCP_BLANKING_Y2**	RW 0x0	OCP Blanking Time
	0: BLK_OFF off 1: BLK_250NS 0.25us 2: BLK_500NS 0.5us 3: BLK_1000NS 1us 4: BLK_2000NS 2us 5: BLK_4000NS 4us 6: BLK_6000NS 6us 7: BLK_8000NS 8us	
[18:16] **HS_OCP_DEGLITCH_Y2**	RW 0x0	OCP Deglitch Time
	0: DEG_OFF off 1: DEG_250NS 0.25us 2: DEG_500NS 0.5us 3: DEG_1000NS 1us 4: DEG_2000NS 2us 5: DEG_4000NS 4us 6: DEG_6000NS 6us 7: DEG_8000NS 8us	
[15] **LS_OCP_USE_VDS_Y2**	RW 0x0	Switches between shunt and RDSon measurement

BITS & NAME	TYPE & RESET	DESCRIPTION
[11:8] **LS_OCP_THRES_Y2**	RW 0x0	Threshold of the low-side overcurrent protection.
	0: THRES_80_63MV 80mV (SHUNT), 63mV (VDS) 1: THRES_165_125MV 165mV (SHUNT), 125mV (VDS) 2: THRES_250_187MV 250mV (SHUNT), 187mV (VDS) 3: THRES_330_248MV 330mV (SHUNT), 248mV (VDS) 4: THRES_415_312MV 415mV (SHUNT), 312mV (VDS) 5: THRES_500_374MV 500mV (SHUNT), 374mV (VDS) 6: THRES_582_434MV 582mV (SHUNT), 434mV (VDS) 7: THRES_660_504MV 660mV (SHUNT), 504mV (VDS) 8: THRES_125_705MV 125mV (SHUNT), 705mV (VDS) 9: THRES_250_940MV 250mV (SHUNT), 940mV (VDS) 10: THRES_375_1180MV 375mV (SHUNT), 1180mV (VDS) 11: THRES_500_1410MV 500mV (SHUNT), 1410mV (VDS) 12: THRES_625_1650MV 625mV (SHUNT), 1650mV (VDS) 13: THRES_750_1880MV 750mV (SHUNT), 1880mV (VDS) 14: THRES_873_2110MV 873mV (SHUNT), 2110mV (VDS) 15: THRES_1000_2350MV 1000mV (SHUNT), 2350mV (VDS)	
[6:4] **LS_OCP_BLANKING_Y2**	RW 0x0	OCP Blanking Time
	0: BLK_OFF off 1: BLK_250NS 0.25us 2: BLK_500NS 0.5us 3: BLK_1000NS 1us 4: BLK_2000NS 2us 5: BLK_4000NS 4us 6: BLK_6000NS 6us 7: BLK_8000NS 8us	
[2:0] **LS_OCP_DEGLITCH_Y2**	RW 0x0	OCP Deglitch Time
	0: DEG_OFF off 1: DEG_250NS 0.25us 2: DEG_500NS 0.5us 3: DEG_1000NS 1us 4: DEG_2000NS 2us 5: DEG_4000NS 4us 6: DEG_6000NS 6us 7: DEG_8000NS 8us	

0x1EE, Block 0: MCC_GDRV_PROT_EN

Enables the protection mechanism for the different fault events, note that the fault needs to be set in MCC_GDRV_STATUS_EN too.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31] **VS_UVLO_PROT**	RW 0x0	Disables the gate driver on VS undervoltage event occurs and the respective MCC_GDRV_STATUS_EN bit is enabled.
[29] **VDRV_UVLO_PROT**	RW 0x0	Disables the gate driver on VDRV undervoltage event occurs and the respective MCC_GDRV_STATUS_EN bit is enabled.

BITS & NAME	TYPE & RESET	DESCRIPTION
[27] **HS_VGS_ON_SHORT_PROT_Y2**	RW 0x0	Disables the gate driver if a gate short event occurs while the gate is turned on and the respective MCC_GDRV_STATUS_EN bit is enabled.
[26] **HS_VGS_ON_SHORT_PROT_W**	RW 0x0	Disables the gate driver if a gate short event occurs while the gate is turned on and the respective MCC_GDRV_STATUS_EN bit is enabled.
[25] **HS_VGS_ON_SHORT_PROT_V**	RW 0x0	Disables the gate driver if a gate short event occurs while the gate is turned on and the respective MCC_GDRV_STATUS_EN bit is enabled.
[24] **HS_VGS_ON_SHORT_PROT_U**	RW 0x0	Disables the gate driver if a gate short event occurs while the gate is turned on and the respective MCC_GDRV_STATUS_EN bit is enabled.
[23] **HS_VGS_OFF_SHORT_PROT_Y2**	RW 0x0	Disables the gate driver if a gate short event occurs while the gate is turned off and the respective MCC_GDRV_STATUS_EN bit is enabled.
[22] **HS_VGS_OFF_SHORT_PROT_W**	RW 0x0	Disables the gate driver if a gate short event occurs while the gate is turned off and the respective MCC_GDRV_STATUS_EN bit is enabled.
[21] **HS_VGS_OFF_SHORT_PROT_V**	RW 0x0	Disables the gate driver if a gate short event occurs while the gate is turned off and the respective MCC_GDRV_STATUS_EN bit is enabled.
[20] **HS_VGS_OFF_SHORT_PROT_U**	RW 0x0	Disables the gate driver if a gate short event occurs while the gate is turned off and the respective MCC_GDRV_STATUS_EN bit is enabled.
[19] **HS_SHORT_PROT_Y2**	RW 0x0	Disables the channel if an overcurrent event occurs and the respective MCC_GDRV_STATUS_EN bit is enabled.
[18] **HS_SHORT_PROT_W**	RW 0x0	Disables the channel if an overcurrent event occurs and the respective MCC_GDRV_STATUS_EN bit is enabled.
[17] **HS_SHORT_PROT_V**	RW 0x0	Disables the channel if an overcurrent event occurs and the respective MCC_GDRV_STATUS_EN bit is enabled.
[16] **HS_SHORT_PROT_U**	RW 0x0	Disables the channel if an overcurrent event occurs and the respective MCC_GDRV_STATUS_EN bit is enabled.
[15] **BST_UVLO_PROT_Y2**	RW 0x0	Disables the affected phase if a bootstrap capacitor undervoltage event occurs and the respective MCC_GDRV_STATUS_EN bit is enabled.
[14] **BST_UVLO_PROT_W**	RW 0x0	Disables the affected phase if a bootstrap capacitor undervoltage event occurs and the respective MCC_GDRV_STATUS_EN bit is enabled.

BITS & NAME	TYPE & RESET	DESCRIPTION
[13] **BST_UVLO_PROT_V**	RW 0x0	Disables the affected phase if a bootstrap capacitor undervoltage event occurs and the respective MCC_GDRV_STATUS_EN bit is enabled.
[12] **BST_UVLO_PROT_U**	RW 0x0	Disables the affected phase if a bootstrap capacitor undervoltage event occurs and the respective MCC_GDRV_STATUS_EN bit is enabled.
[11] **LS_VGS_ON_SHORT_PROT_Y2**	RW 0x0	Disables the gate driver if a gate short event occurs while the gate is turned on and the respective MCC_GDRV_STATUS_EN bit is enabled.
[10] **LS_VGS_ON_SHORT_PROT_W**	RW 0x0	Disables the gate driver if a gate short event occurs while the gate is turned on and the respective MCC_GDRV_STATUS_EN bit is enabled.
[9] **LS_VGS_ON_SHORT_PROT_V**	RW 0x0	Disables the gate driver if a gate short event occurs while the gate is turned on and the respective MCC_GDRV_STATUS_EN bit is enabled.
[8] **LS_VGS_ON_SHORT_PROT_U**	RW 0x0	Disables the gate driver if a gate short event occurs while the gate is turned on and the respective MCC_GDRV_STATUS_EN bit is enabled.
[7] **LS_VGS_OFF_SHORT_PROT_Y2**	RW 0x0	Disables the gate driver if a gate short event occurs while the gate is turned off and the respective MCC_GDRV_STATUS_EN bit is enabled.
[6] **LS_VGS_OFF_SHORT_PROT_W**	RW 0x0	Disables the gate driver if a gate short event occurs while the gate is turned off and the respective MCC_GDRV_STATUS_EN bit is enabled.
[5] **LS_VGS_OFF_SHORT_PROT_V**	RW 0x0	Disables the gate driver if a gate short event occurs while the gate is turned off and the respective MCC_GDRV_STATUS_EN bit is enabled.
[4] **LS_VGS_OFF_SHORT_PROT_U**	RW 0x0	Disables the gate driver if a gate short event occurs while the gate is turned off and the respective MCC_GDRV_STATUS_EN bit is enabled.
[3] **LS_SHORT_PROT_Y2**	RW 0x0	Disables the channel if an overcurrent event occurs and the respective MCC_GDRV_STATUS_EN bit is enabled.
[2] **LS_SHORT_PROT_W**	RW 0x0	Disables the channel if an overcurrent event occurs and the respective MCC_GDRV_STATUS_EN bit is enabled.
[1] **LS_SHORT_PROT_V**	RW 0x0	Disables the channel if an overcurrent event occurs and the respective MCC_GDRV_STATUS_EN bit is enabled.
[0] **LS_SHORT_PROT_U**	RW 0x0	Disables the channel if an overcurrent event occurs and the respective MCC_GDRV_STATUS_EN bit is enabled.

0x1EF, Block 0: MCC_GDRV_STATUS_EN

Enables the reporting of the different fault events. Required to be activated to get any feedback of the fault status and enable the internal protections.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31] **VS_UVLO_EN**	RW 0x1	If not set, any reporting or action regarding the VS undervoltage event is disabled.
[30] **VDRV_UVLWRN_EN**	RW 0x1	If not set, any reporting or action regarding the VDRV undervoltage warning event is disabled.
[29] **VDRV_UVLO_EN**	RW 0x1	If not set, any reporting or action regarding the VDRV undervoltage event is disabled.
[27] **HS_VGS_ON_SHORT_EN_Y2**	RW 0x1	If not set, any reporting or action regarding the gate short event while the gate is turned on is disabled.
[26] **HS_VGS_ON_SHORT_EN_W**	RW 0x1	If not set, any reporting or action regarding the gate short event while the gate is turned on is disabled.
[25] **HS_VGS_ON_SHORT_EN_V**	RW 0x1	If not set, any reporting or action regarding the gate short event while the gate is turned on is disabled.
[24] **HS_VGS_ON_SHORT_EN_U**	RW 0x1	If not set, any reporting or action regarding the gate short event while the gate is turned on is disabled.
[23] **HS_VGS_OFF_SHORT_EN_Y2**	RW 0x1	If not set, any reporting or action regarding the gate short event while the gate is turned off is disabled.
[22] **HS_VGS_OFF_SHORT_EN_W**	RW 0x1	If not set, any reporting or action regarding the gate short event while the gate is turned off is disabled.
[21] **HS_VGS_OFF_SHORT_EN_V**	RW 0x1	If not set, any reporting or action regarding the gate short event while the gate is turned off is disabled.
[20] **HS_VGS_OFF_SHORT_EN_U**	RW 0x1	If not set, any reporting or action regarding the gate short event while the gate is turned off is disabled.
[19] **HS_SHORT_EN_Y2**	RW 0x1	If not set, any reporting or action regarding the overcurrent event is disabled.
[18] **HS_SHORT_EN_W**	RW 0x1	If not set, any reporting or action regarding the overcurrent event is disabled.

BITS & NAME	TYPE & RESET	DESCRIPTION
[17] **HS_SHORT_EN_V**	RW 0x1	If not set, any reporting or action regarding the overcurrent event is disabled.
[16] **HS_SHORT_EN_U**	RW 0x1	If not set, any reporting or action regarding the overcurrent event is disabled.
[15] **BST_UVLO_EN_Y2**	RW 0x1	If not set, any reporting or action regarding the bootstrap capacitor undervoltage event is disabled.
[14] **BST_UVLO_EN_W**	RW 0x1	If not set, any reporting or action regarding the bootstrap capacitor undervoltage event is disabled.
[13] **BST_UVLO_EN_V**	RW 0x1	If not set, any reporting or action regarding the bootstrap capacitor undervoltage event is disabled.
[12] **BST_UVLO_EN_U**	RW 0x1	If not set, any reporting or action regarding the bootstrap capacitor undervoltage event is disabled.
[11] **LS_VGS_ON_SHORT_EN_Y2**	RW 0x1	If not set, any reporting or action regarding the gate short event while the gate is turned on is disabled.
[10] **LS_VGS_ON_SHORT_EN_W**	RW 0x1	If not set, any reporting or action regarding the gate short event while the gate is turned on is disabled.
[9] **LS_VGS_ON_SHORT_EN_V**	RW 0x1	If not set, any reporting or action regarding the gate short event while the gate is turned on is disabled.
[8] **LS_VGS_ON_SHORT_EN_U**	RW 0x1	If not set, any reporting or action regarding the gate short event while the gate is turned on is disabled.
[7] **LS_VGS_OFF_SHORT_EN_Y2**	RW 0x1	If not set, any reporting or action regarding the gate short event while the gate is turned off is disabled.
[6] **LS_VGS_OFF_SHORT_EN_W**	RW 0x1	If not set, any reporting or action regarding the gate short event while the gate is turned off is disabled.
[5] **LS_VGS_OFF_SHORT_EN_V**	RW 0x1	If not set, any reporting or action regarding the gate short event while the gate is turned off is disabled.
[4] **LS_VGS_OFF_SHORT_EN_U**	RW 0x1	If not set, any reporting or action regarding the gate short event while the gate is turned off is disabled.
[3] **LS_SHORT_EN_Y2**	RW 0x1	If not set, any reporting or action regarding the overcurrent event is disabled.

BITS & NAME	TYPE & RESET	DESCRIPTION
[2] **LS_SHORT_EN_W**	RW 0x1	If not set, any reporting or action regarding the overcurrent event is disabled.
[1] **LS_SHORT_EN_V**	RW 0x1	If not set, any reporting or action regarding the overcurrent event is disabled.
[0] **LS_SHORT_EN_U**	RW 0x1	If not set, any reporting or action regarding the overcurrent event is disabled.

0x1F0, Block 0: MCC_GDRV_STATUS

Contains the status of the different fault events.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31] **VS_UVLO**	RW, W1C 0x0	Undervoltage condition of VS (supply voltage)
[30] **VDRV_UVLWRN**	RW, W1C 0x0	Low voltage warning condition of VDRV (gatedrive voltage)
[29] **VDRV_UVLO**	RW, W1C 0x0	Undervoltage condition of VDRV (gatedrive voltage)
[27] **HS_VGS_ON_SHORT_Y2**	RW, W1C 0x0	Status of the gate short detection while the gate is turned on
[26] **HS_VGS_ON_SHORT_W**	RW, W1C 0x0	Status of the gate short detection while the gate is turned on
[25] **HS_VGS_ON_SHORT_V**	RW, W1C 0x0	Status of the gate short detection while the gate is turned on
[24] **HS_VGS_ON_SHORT_U**	RW, W1C 0x0	Status of the gate short detection while the gate is turned on
[23] **HS_VGS_OFF_SHORT_Y2**	RW, W1C 0x0	Status of the gate short detection while the gate is turned off
[22] **HS_VGS_OFF_SHORT_W**	RW, W1C 0x0	Status of the gate short detection while the gate is turned off
[21] **HS_VGS_OFF_SHORT_V**	RW, W1C 0x0	Status of the gate short detection while the gate is turned off

BITS & NAME	TYPE & RESET	DESCRIPTION
[20] **HS_VGS_OFF_SHORT_U**	RW, W1C 0x0	Status of the gate short detection while the gate is turned off
[19] **HS_SHORT_Y2**	RW, W1C 0x0	Status of the overcurrent protection
[18] **HS_SHORT_W**	RW, W1C 0x0	Status of the overcurrent protection
[17] **HS_SHORT_V**	RW, W1C 0x0	Status of the overcurrent protection
[16] **HS_SHORT_U**	RW, W1C 0x0	Status of the overcurrent protection
[15] **BST_UVLO_Y2**	RW, W1C 0x0	Undervoltage condition of the bootstrap cap on phase Y2
[14] **BST_UVLO_W**	RW, W1C 0x0	Undervoltage condition of the bootstrap cap on phase W
[13] **BST_UVLO_V**	RW, W1C 0x0	Undervoltage condition of the bootstrap cap on phase V
[12] **BST_UVLO_U**	RW, W1C 0x0	Undervoltage condition of the bootstrap cap on phase U
[11] **LS_VGS_ON_SHORT_Y2**	RW, W1C 0x0	Status of the gate short detection while the gate is turned on
[10] **LS_VGS_ON_SHORT_W**	RW, W1C 0x0	Status of the gate short detection while the gate is turned on
[9] **LS_VGS_ON_SHORT_V**	RW, W1C 0x0	Status of the gate short detection while the gate is turned on
[8] **LS_VGS_ON_SHORT_U**	RW, W1C 0x0	Status of the gate short detection while the gate is turned on
[7] **LS_VGS_OFF_SHORT_Y2**	RW, W1C 0x0	Status of the gate short detection while the gate is turned off
[6] **LS_VGS_OFF_SHORT_W**	RW, W1C 0x0	Status of the gate short detection while the gate is turned off
[5] **LS_VGS_OFF_SHORT_V**	RW, W1C 0x0	Status of the gate short detection while the gate is turned off

BITS & NAME	TYPE & RESET	DESCRIPTION
[4] **LS_VGS_OFF_SHORT_U**	RW, W1C 0x0	Status of the gate short detection while the gate is turned off
[3] **LS_SHORT_Y2**	RW, W1C 0x0	Status of the overcurrent protection
[2] **LS_SHORT_W**	RW, W1C 0x0	Status of the overcurrent protection
[1] **LS_SHORT_V**	RW, W1C 0x0	Status of the overcurrent protection
[0] **LS_SHORT_U**	RW, W1C 0x0	Status of the overcurrent protection

0x1F1, Block 0: MCC_GDRV_FAULT

Contains the status of the different fault events.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31] **VS_UVLO_STS**	R 0x0	Undervoltage condition of VS (supply voltage)
[30] **VDRV_UVLWRN_STS**	R 0x0	Low voltage warning condition of VDRV (gatedrive voltage)
[29] **VDRV_UVLO_STS**	R 0x0	Undervoltage condition of VDRV (gatedrive voltage)
[19] **HS_FAULT_ACTIVE_Y2**	RW, W1C 0x0	Set if any fault occured on the phase that triggered the hardware protection, clear to resume normal operation. Make sure to clear both HS_FAULT_ACTIVE and LS_FAULT_ACTIVE simultaneously to resume operation.
[18] **HS_FAULT_ACTIVE_W**	RW, W1C 0x0	Set if any fault occured on the phase that triggered the hardware protection, clear to resume normal operation. Make sure to clear both HS_FAULT_ACTIVE and LS_FAULT_ACTIVE simultaneously to resume operation.
[17] **HS_FAULT_ACTIVE_V**	RW, W1C 0x0	Set if any fault occured on the phase that triggered the hardware protection, clear to resume normal operation. Make sure to clear both HS_FAULT_ACTIVE and LS_FAULT_ACTIVE simultaneously to resume operation.
[16] **HS_FAULT_ACTIVE_U**	RW, W1C 0x0	Set if any fault occured on the phase that triggered the hardware protection, clear to resume normal operation. Make sure to clear both HS_FAULT_ACTIVE and LS_FAULT_ACTIVE simultaneously to resume operation.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15] **BST_UVLO_STS_Y2**	R 0x0	Undervoltage condition of the bootstrap cap on phase Y2
[14] **BST_UVLO_STS_W**	R 0x0	Undervoltage condition of the bootstrap cap on phase W
[13] **BST_UVLO_STS_V**	R 0x0	Undervoltage condition of the bootstrap cap on phase V
[12] **BST_UVLO_STS_U**	R 0x0	Undervoltage condition of the bootstrap cap on phase U
[3] **LS_FAULT_ACTIVE_Y2**	RW, W1C 0x0	Set if any fault occurred on the phase that triggered the hardware protection, clear to resume normal operation. Make sure to clear both HS_FAULT_ACTIVE and LS_FAULT_ACTIVE simultaneously to resume operation.
[2] **LS_FAULT_ACTIVE_W**	RW, W1C 0x0	Set if any fault occurred on the phase that triggered the hardware protection, clear to resume normal operation. Make sure to clear both HS_FAULT_ACTIVE and LS_FAULT_ACTIVE simultaneously to resume operation.
[1] **LS_FAULT_ACTIVE_V**	RW, W1C 0x0	Set if any fault occurred on the phase that triggered the hardware protection, clear to resume normal operation. Make sure to clear both HS_FAULT_ACTIVE and LS_FAULT_ACTIVE simultaneously to resume operation.
[0] **LS_FAULT_ACTIVE_U**	RW, W1C 0x0	Set if any fault occurred on the phase that triggered the hardware protection, clear to resume normal operation. Make sure to clear both HS_FAULT_ACTIVE and LS_FAULT_ACTIVE simultaneously to resume operation.

0x200, Block 0: MCC_ADC_I1_I0_EXT

External writable ADC value for phase I1, I0

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **I1**	RW, signed 0x0000	External writable ADC value for phase I1
[15:0] **I0**	RW, signed 0x0000	External writable ADC value for phase I0

0x201, Block 0: MCC_ADC_I2_EXT

External writable ADC value for phase I2. Note that depending on the selection in register MCC_ADC_I_GEN_CONFIG -> TRIGGER_SELECT, writing to this register may not trigger a new processing of the ADC data. So when updating all three external ADC values, it is recommended to update this register first.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **I2**	RW, signed 0x0000	External writable ADC value for phase I2

0x202, Block 0: MCC_PWM_VX2_UX1_EXT

External writable PWM value.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **VX2**	RW, unsigned 0x0000	External writable PWM value for phase VX2.
[15:0] **UX1**	RW, unsigned 0x0000	External writable PWM value for phase UX1.

0x203, Block 0: MCC_PWM_Y2_WY1_EXT

External writable PWM value.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **Y2**	RW, unsigned 0x0000	External writable PWM value for phase Y2.
[15:0] **WY1**	RW, unsigned 0x0000	External writable PWM value for phase WY1.

0x204, Block 0: MCC_PWM_EXT_Y2_ALT

External writable PWM compare value for phase Y2_ALT.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:0] **PWM_EXT_Y2_ALT**	RW, unsigned 0x0000	Duty cycle to be used in independent Y2 mode. This is not scaled to the PWM_MAXCNT value and therefore needs to be set accordingly. Duty cycle = PWM_EXT_Y2_ALT / (PWM_MAXCNT + 1)

0x205, Block 0: MCC_VOLTAGE_EXT

External writable parameter for open-loop voltage control mode, useful during system setup.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **UQ**	RW, signed 0x0000	U_Q component in Voltage mode.
[15:0] **UD**	RW, signed 0x0000	U_D component in Voltage mode.

0x206, Block 0: MCC_PHI_EXT

Angle phi_e_ext and phi_m_ext for external writing into this register.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:16] **PHI_M_EXT**	RW, signed 0x0000	Angle phi_m_ext for external input.
[15:0] **PHI_E_EXT**	RW, signed 0x0000	Angle phi_e_ext for external input.

0x208, Block 0: MCC_VELOCITY_EXT

Actual velocity for external override.

BITS & NAME	TYPE & RESET	DESCRIPTION
[31:0] **VELOCITY_EXT**	RW, signed 0x00000000	Actual velocity for SW override.

0x008, Block 2: FAULT_STS

General system status flags. Bits are set and cleared automatically.

BITS & NAME	TYPE & RESET	DESCRIPTION
[12] **LDO2_READY**	R 0x0	LDO2 (V_EXT2) has completed soft-start.
[11] **LDO1_READY**	R 0x0	LDO1 (V_EXT1) has completed soft-start.
[10] **VCCIO_UVLO**	R 0x0	VCCIO Undervoltage.

BITS & NAME	TYPE & RESET	DESCRIPTION
[9] **VDDA_UVLO**	R 0x0	VDDA Undervoltage.
[8] **VDD_UVLO**	R 0x0	VDD Undervoltage.
[7] **VSA_UVLO**	R 0x0	VSA Undervoltage.
[6] **CHGP_SHORT**	R 0x0	VDRV charge pump short status.
[5] **CHGP_OK**	R 0x0	VDRV charge pump power-up is currently okay.
[4] **LDOEXT2_SHORT**	R 0x0	LDO2 (V_EXT2) is shorted.
[3] **LDOEXT1_SHORT**	R 0x0	LDO1 (V_EXT1) is shorted.
[2] **LDOEXT_TSD**	R 0x0	LDO thermal shutdown status.
[1] **BCK_SHORT**	R 0x0	VBUCK shorted.
[0] **BCK_UVLO**	R 0x0	VBUCK Undervoltage.

0x009, Block 2: FAULT_R_INT

General system status flags. Bits are set automatically if corresponding bit in FAULT_STS is set. Each bit must be cleared manually.

BITS & NAME	TYPE & RESET	DESCRIPTION
[15] **UC_FAULT**	RW 0x0	Force fault pin assertion.
[12] **LDO2_READY_RE_LTC**	RW, W1C 0x0	LDO2READY latched bit. Write 1 to clear this bit.
[11] **LDO1_READY_RE_LTC**	RW, W1C 0x0	LDO1READY latched bit. Write 1 to clear this bit.

BITS & NAME	TYPE & RESET	DESCRIPTION
[10] **VCCIO_UVLO_LTC**	RW, W1C 0x0	VCCIO_UVLO latched bit. Write 1 to clear this bit.
[9] **VDDA_UVLO_LTC**	RW, W1C 0x0	VDDA_UVLO latched bit. Write 1 to clear this bit.
[8] **VDD_UVLO_LTC**	RW, W1C 0x0	VDD_UVLO latched bit. Write 1 to clear this bit.
[7] **VSA_UVLO_LTC**	RW, W1C 0x0	VSA_UVLO latched bit. Write 1 to clear this bit.
[6] **CHGP_SHORT_LTC**	RW, W1C 0x0	CHGP_SHORT latched bit. Write 1 to clear this bit. The VDRV charge pump does not turn off because of a short, but rather it enters current limitation. However, once the charge pump is turned off, the bit has to be zero in order to turn it on again.
[5] **CHGP_OK_LTC**	RW, W1C 0x0	CHGP_OK latched bit. Write 1 to clear this bit.
[4] **LDOEXT2_SHORT_LTC**	RW, W1C 0x0	LDO2EXT_SHORT latched bit. Write 1 to clear this bit.
[3] **LDOEXT1_SHORT_LTC**	RW, W1C 0x0	LDO1EXT_SHORT latched bit. Write 1 to clear this bit.
[2] **LDOEXT_TSD_LTC**	RW, W1C 0x0	LDOEXT_TSD latched bit. Write 1 to clear this bit.
[1] **BCK_SHORT_RE_LTC**	RW, W1C 0x0	BUCK_SHORT latched bit. Write 1 to clear this bit.
[0] **BCK_UVLO_LTC**	RW, W1C 0x0	BUCK_UVLO latched bit. Write 1 to clear this bit.

0x00A, Block 2: FAULT_R_ENA_F

Mask for register FAULT_R_INT. If any bit in this mask register is set and the corresponding flag in the FAULT_R_INT register is also set, the status can be seen at the FAULTN pin.

BITS & NAME	TYPE & RESET	DESCRIPTION
[12] **LDO2_READY_RE_ENA_F**	RW 0x0	LDO2_READY_LTC mask bit for fault pin.
[11] **LDO1_READY_RE_ENA_F**	RW 0x0	LDO1_READY_LTC mask bit for fault pin.

BITS & NAME	TYPE & RESET	DESCRIPTION
[10] **VCCIO_UVLO_ENA_F**	RW 0x1	VCCIO_UVLO_LTC mask bit for fault pin.
[9] **VDDA_UVLO_ENA_F**	RW 0x1	VDDA_UVLO_LTC mask bit for fault pin.
[8] **VDD_UVLO_ENA_F**	RW 0x1	VDD_UVLO_LTC mask bit for fault pin.
[7] **VSA_UVLO_ENA_F**	RW 0x1	VSA_UVLO_LTC mask bit for fault pin.
[6] **CHGP_SHORT_ENA_F**	RW 0x0	CHGP_SHORT_LTC mask bit for fault pin.
[5] **CHGP_OK_ENA_F**	RW 0x0	CHGP_OK_LTC mask bit for fault pin.
[4] **LDOEXT2_SHORT_ENA_F**	RW 0x0	LDO2EXT_SHORT_LTC mask bit for fault pin.
[3] **LDOEXT1_SHORT_ENA_F**	RW 0x0	LDO1EXT_SHORT_LTC mask bit for fault pin.
[2] **LDOEXT_TSD_ENA_F**	RW 0x0	LDOEXT_LTC thermal shutdown mask bit for fault pin.
[1] **BCK_SHORT_RE_ENA_F**	RW 0x1	BUCK_SHORT mask bit for fault pin.
[0] **BCK_UVLO_ENA_F**	RW 0x1	BUCK_UVLO mask bit for fault pin.

0x001, Block 1: ADC_SRC_CONFIG

ADC sources config

BITS & NAME	TYPE & RESET	DESCRIPTION
[31] **ADC3_MUX2_DETOUR**	RW 0x1	MUX2 measurement is skipped for a second MUX1 measurement (AIN3 value) during one PWM period in the ADC3 acquisition sequence. Sequence position of this second measurement is still defined by ADC3_MUX2 _CFG
	0: NO_CHANGE no changes 1: ADC3_MUX2_DETOUR MUX2 measurement is skipped for a second MUX1 (AIN3) measurement	

BITS & NAME	TYPE & RESET	DESCRIPTION
[29:28] **ADC3_MUX2_CFG**	RW 0x3	Measurement position of MUX2 input (phase voltage value of Y2) in ADC3 current acquisition sequence
	0: ADC3_MUX2_OFF 1: ADC3_MUX2_1ST 2: ADC3_MUX2_2ND 3: ADC3_MUX2_3RD	skip MUX input 2 MUX input 2 is sampled first after trigger MUX input 2 is sampled second after trigger MUX input 2 is sampled third after trigger
[27:26] **ADC3_MUX1_CFG**	RW 0x2	Measurement position of MUX1 input (AIN3 value) in ADC3 current acquisition sequence
	0: ADC3_MUX1_OFF 1: ADC3_MUX1_1ST 2: ADC3_MUX1_2ND 3: ADC3_MUX1_3RD	skip MUX input 1 MUX input 1 is sampled first after trigger MUX input 1 is sampled second after trigger MUX input 1 is sampled third after trigger
[25:24] **ADC3_MUX0_CFG**	RW 0x1	Measurement position of MUX0 input (CSA value of Y2) in ADC3 current acquisition sequence
	0: ADC3_MUX0_OFF 1: ADC3_MUX0_1ST 2: ADC3_MUX0_2ND 3: ADC3_MUX0_3RD	skip MUX input 0 MUX input 0 is sampled first after trigger MUX input 0 is sampled second after trigger MUX input 0 is sampled third after trigger
[23] **ADC2_MUX2_DETOUR**	RW 0x1	MUX2 measurement is skipped for a second MUX1 measurement (AIN2 value) during one PWM period in the ADC2 acquisition sequence. Sequence position of this second measurement is still defined by ADC2_MUX2 _CFG
	0: NO_CHANGE 1: ADC2_MUX2_DETOUR	no changes MUX2 measurement is skipped for a second MUX1 (AIN2) measurement
[22] **ADC2_MUX3_DIS**	RW 0x0	Disable measurement of MUX3 input (junction temperature VTJ value) in ADC2 current acquisition sequence
[21:20] **ADC2_MUX2_CFG**	RW 0x3	Measurement position of MUX2 input (phase voltage value of WY1) in ADC2 current acquisition sequence
	0: ADC2_MUX2_OFF 1: ADC2_MUX2_1ST 2: ADC2_MUX2_2ND 3: ADC2_MUX2_3RD	skip MUX input 2 MUX input 2 is sampled first after trigger MUX input 2 is sampled second after trigger MUX input 2 is sampled third after trigger
[19:18] **ADC2_MUX1_CFG**	RW 0x2	Measurement position of MUX1 input (AIN2 value) in ADC2 current acquisition sequence
	0: ADC2_MUX1_OFF 1: ADC2_MUX1_1ST 2: ADC2_MUX1_2ND 3: ADC2_MUX1_3RD	skip MUX input 1 MUX input 1 is sampled first after trigger MUX input 1 is sampled second after trigger MUX input 1 is sampled third after trigger
[17:16] **ADC2_MUX0_CFG**	RW 0x1	Measurement position of MUX0 input (CSA value of WY1) in ADC2 current acquisition sequence
	0: ADC2_MUX0_OFF 1: ADC2_MUX0_1ST 2: ADC2_MUX0_2ND 3: ADC2_MUX0_3RD	skip MUX input 0 MUX input 0 is sampled first after trigger MUX input 0 is sampled second after trigger MUX input 0 is sampled third after trigger

BITS & NAME	TYPE & RESET	DESCRIPTION
[15] **ADC1_MUX2_DETOUR**	RW 0x1	MUX2 measurement is skipped for a second MUX1 measurement (AIN1 value) during one PWM period in the ADC1 acquisition sequence. Sequence position of this second measurement is still defined by ADC1_MUX2 _CFG
	0: NO_CHANGE no changes 1: ADC1_MUX2_DETOUR MUX2 measurement is skipped for a second MUX1 (AIN1) measurement	
[13:12] **ADC1_MUX2_CFG**	RW 0x3	Measurement position of MUX2 input (phase voltage value of VX2) in ADC1 current acquisition sequence
	0: ADC1_MUX2_OFF skip MUX input 2 1: ADC1_MUX2_1ST MUX input 2 is sampled first after trigger 2: ADC1_MUX2_2ND MUX input 2 is sampled second after trigger 3: ADC1_MUX2_3RD MUX input 2 is sampled third after trigger	
[11:10] **ADC1_MUX1_CFG**	RW 0x2	Measurement position of MUX1 input (AIN1 value) in ADC1 current acquisition sequence
	0: ADC1_MUX1_OFF skip MUX input 1 1: ADC1_MUX1_1ST MUX input 1 is sampled first after trigger 2: ADC1_MUX1_2ND MUX input 1 is sampled second after trigger 3: ADC1_MUX1_3RD MUX input 1 is sampled third after trigger	
[9:8] **ADC1_MUX0_CFG**	RW 0x1	Measurement position of MUX1 input (CSA value of VX2) in ADC1 current acquisition sequence
	0: ADC1_MUX0_OFF skip MUX input 0 1: ADC1_MUX0_1ST MUX input 0 is sampled first after trigger 2: ADC1_MUX0_2ND MUX input 0 is sampled second after trigger 3: ADC1_MUX0_3RD MUX input 0 is sampled third after trigger	
[7] **ADC0_MUX2_DETOUR**	RW 0x1	MUX2 measurement is skipped for a second MUX1 measurement (AIN0 value) during one PWM period in the ADC0 acquisition sequence. Sequence position of this second measurement is still defined by ADC0_MUX2 _CFG
	0: NO_CHANGE no changes 1: ADC0_MUX2_DETOUR MUX2 measurement is skipped for a second MUX1 (AIN0) measurement	
[6] **ADC0_MUX3_DIS**	RW 0x0	Disable measurement of MUX3 input (supply voltage value) in ADC0 current acquisition sequence
[5:4] **ADC0_MUX2_CFG**	RW 0x3	Measurement position of MUX2 input (phase voltage value of UX1) in ADC0 current acquisition sequence
	0: ADC0_MUX2_OFF skip MUX input 2 1: ADC0_MUX2_1ST MUX input 2 is sampled first after trigger 2: ADC0_MUX2_2ND MUX input 2 is sampled second after trigger 3: ADC0_MUX2_3RD MUX input 2 is sampled third after trigger	
[3:2] **ADC0_MUX1_CFG**	RW 0x2	Measurement position of MUX1 input (AIN0 value) in ADC0 current acquisition sequence
	0: ADC0_MUX1_OFF skip MUX input 1 1: ADC0_MUX1_1ST MUX input 1 is sampled first after trigger 2: ADC0_MUX1_2ND MUX input 1 is sampled second after trigger 3: ADC0_MUX1_3RD MUX input 1 is sampled third after trigger	

BITS & NAME	TYPE & RESET	DESCRIPTION
[1:0] **ADC0_MUX0_CFG**	RW 0x1	Measurement position of MUX0 input (CSA value of UX1) in ADC0 current acquisition sequence
	0: ADC0_MUX0_OFF 1: ADC0_MUX0_1ST 2: ADC0_MUX0_2ND 3: ADC0_MUX0_3RD	skip MUX input 0 MUX input 0 is sampled first after trigger MUX input 0 is sampled second after trigger MUX input 0 is sampled third after trigger

0x002, Block 1: ADC_SETUP

ADC setup

BITS & NAME	TYPE & RESET	DESCRIPTION
[19:16] **ADC_SHIFT_SAMPLE**	RW 0x0	Shift ADC sample time in steps of 100ns, base = 500ns
	0: ADC_SHIFT_500NS 500ns 1: ADC_SHIFT_600NS 600ns 2: ADC_SHIFT_700NS 700ns 3: ADC_SHIFT_800NS 800ns 4: ADC_SHIFT_900NS 900ns 5: ADC_SHIFT_1000NS 1000ns 6: ADC_SHIFT_1100NS 1100ns 7: ADC_SHIFT_1200NS 1200ns 8: ADC_SHIFT_1300NS 1300ns 9: ADC_SHIFT_1400NS 1400ns 10: ADC_SHIFT_1500NS 1500ns 11: ADC_SHIFT_1600NS 1600ns 12: ADC_SHIFT_1700NS 1700ns 13: ADC_SHIFT_1800NS 1800ns 14: ADC_SHIFT_1900NS 1900ns 15: ADC_SHIFT_2000NS 2000ns	

0x005, Block 1: ADC_STATUS

ADC status bits

BITS & NAME	TYPE & RESET	DESCRIPTION
[15] **ADC3_MUXSEQ_FAIL**	R 0x0	ADC3 acquisition sequence is not set correctly, check particular selection bits in ADC_SRC_CONFIG register
	0: ADC3_SEQ_OK 1: ADC3_SEQ_FAIL	No fail Fail
[14] **ADC2_MUXSEQ_FAIL**	R 0x0	ADC2 acquisition sequence is not set correctly, check particular selection bits in ADC_SRC_CONFIG register
	0: ADC2_SEQ_OK 1: ADC2_SEQ_FAIL	No fail Fail
[13] **ADC1_MUXSEQ_FAIL**	R 0x0	ADC1 acquisition sequence is not set correctly, check particular selection bits in ADC_SRC_CONFIG register
	0: ADC1_SEQ_OK 1: ADC1_SEQ_FAIL	No fail Fail

BITS & NAME	TYPE & RESET	DESCRIPTION
[12] **ADC0_MUXSEQ_FAIL**	R 0x0	ADC0 acquisition sequence is not set correctly, check particular selection bits in ADC_SRC_CONFIG register
	0: ADC0_SEQ_OK 1: ADC0_SEQ_FAIL	No fail Fail
[11] **ADC3_WTCHDG_FAIL**	R 0x0	ADC3 is not responding correctly
	0: ADC3_OK 1: ADC3_FAIL	No fail Fail
[10] **ADC2_WTCHDG_FAIL**	R 0x0	ADC2 is not responding correctly
	0: ADC2_OK 1: ADC2_FAIL	No fail Fail
[9] **ADC1_WTCHDG_FAIL**	R 0x0	ADC1 is not responding correctly
	0: ADC1_OK 1: ADC1_FAIL	No fail Fail
[8] **ADC0_WTCHDG_FAIL**	R 0x0	ADC0 is not responding correctly
	0: ADC0_OK 1: ADC0_FAIL	No fail Fail
[3] **RDY_ADC_3**	R 0x0	ADC3 status
	0: ADC3_NRDY 1: ADC3_RDY	ADC3 calibration ongoing ADC3 is calibrated and ready for normal operation
[2] **RDY_ADC_2**	R 0x0	ADC2 status
	0: ADC2_NRDY 1: ADC2_RDY	ADC2 calibration ongoing ADC2 is calibrated and ready for normal operation
[1] **RDY_ADC_1**	R 0x0	ADC1 status
	0: ADC1_NRDY 1: ADC1_RDY	ADC1 calibration ongoing ADC1 is calibrated and ready for normal operation
[0] **RDY_ADC_0**	R 0x0	ADC0 status
	0: ADC0_NRDY 1: ADC0_RDY	ADC0 calibration ongoing ADC0 is calibrated and ready for normal operation

0x007, Block 1: CSA_SETUP

CSA setup

BITS & NAME	TYPE & RESET	DESCRIPTION
[19:16] **CSA_AZ_FLTLNGTH_EXP**	RW 0x0	Filter length exponent of digital filter (moving average style) for all raw AZ values
	0: CSA_AZ_FILT_OFF 1: CSA_AZ_FILT_2 2: CSA_AZ_FILT_4 3: CSA_AZ_FILT_8	filtLength=1 (no filter) filtLength=2 (filter over 2 values) filtLength=4 (filter over 4 values) filtLength=8 (filter over 8 values)

BITS & NAME	TYPE & RESET	DESCRIPTION
[15:14] **CSA3_FILT**	RW 0x0	BW filter settings for CSA3 (adapt PWM cycle and ADC_SHIFT_SAMPLE where applicable)
	0: CSA3_FILT_0U55 0.55us 1: CSA3_FILT_0U75 0.75us 2: CSA3_FILT_1U00 1.00us 3: CSA3_FILT_1U35 1.35us	
[13:12] **CSA012_FILT**	RW 0x0	BW filter settings for CSA0...2 (adapt PWM cycle and ADC_SHIFT_SAMPLE where applicable)
	0: CSA012_FILT_0U55 0.55us 1: CSA012_FILT_0U75 0.75us 2: CSA012_FILT_1U00 1.00us 3: CSA012_FILT_1U35 1.35us	
[10] **CSA3_BYPASS**	RW 0x0	Bypass of CSA3; results in gain=-1
	0: CSA3_BYPASS_OFF no bypass 1: CSA3_BYPASS_EN x1 (bypass CSA)	
[9:8] **CSA3_GAIN**	RW 0x0	Gain for CSA3 in case no bypass is active
	0: CSA3_GAIN_X5 x5 1: CSA3_GAIN_X10 x10 2: CSA3_GAIN_X20 x20 3: CSA3_GAIN_X40 x40	
[6] **CSA012_BYPASS**	RW 0x0	Bypass of CSA3; results in gain=-1
	0: CSA012_BYPASS_OF F no bypass 1: CSA012_BYPASS_EN x1 (bypass CSA)	
[5:4] **CSA012_GAIN**	RW 0x0	Gain for CSA0...2 in case no bypass is active
	0: CSA012_GAIN_X5 x5 1: CSA012_GAIN_X10 x10 2: CSA012_GAIN_X20 x20 3: CSA012_GAIN_X40 x40	
[3] **CSA3_EN**	RW 0x0	CSA3 Enable bit
	0: CSA3_OFF CSA3 disabled 1: CSA3_EN CSA3 enabled	
[2] **CSA2_EN**	RW 0x0	CSA2 Enable bit
	0: CSA2_OFF CSA2 disabled 1: CSA2_EN CSA2 enabled	
[1] **CSA1_EN**	RW 0x0	CSA1 Enable bit
	0: CSA1_OFF CSA1 disabled 1: CSA1_EN CSA1 enabled	
[0] **CSA0_EN**	RW 0x0	CSA0 Enable bit
	0: CSA0_OFF CSA0 disabled 1: CSA0_EN CSA0 enabled	

REVISION HISTORY

Revision History

Revision Number	Revision Date	Change(s)
0	04/25	Initial release