Sensored trapezoidal control firmware design in EVAL_6EDL7141_TRAP_1SH

Detailed descriptions of the firmware

Author: Leonard Xiao

About this document

Scope and purpose

This document describes the detailed function blocks of the firmware, which are provided as a reference with the **EVAL_6EDL7141_TRAP_1SH** motor drive board. The board is composed of the **XMC1400 series microcontroller**, **MOTIX™ 6EDL7141 three-phase smart driver IC** and Infineon **OptiMOS™ 6** best-in-class power MOSFETs. The system is targeted at cordless power tools and other battery-powered motor drive applications. Further modifications can be made to adapt the firmware to customized boards or motors.

Intended audience

This document is aimed at designers wishing to gain deeper knowledge of the trapezoidal control firmware solution based on the EVAL_6EDL7141_TRAP_1SH board, which addresses the market of cordless power tool and other battery-powered motor drive applications. It is intended for design engineers, application engineers and students.

Table of contents

About this document **1**
Table of contents **1**
1 Introduction **3**
1.1 Installation 3
1.2 Directories and files 3
2 Architecture **5**
2.1 Block diagram 5
2.2 XMC™ resource allocation 5
2.3 Managing parameters 7
2.4 Memory map 8
2.5 Interrupts 8
2.5.1 Commutation ISR 9
2.5.2 Hall event ISR 9
2.5.3 Control-loop ISR 10
2.5.4 Systick ISR 10
2.6 State machine 10
2.7 Control schemes 12
2.7.1 Control scheme configuration 12
2.7.2 Voltage control 13
2.7.3 Speed control 13
2.7.4 Current control 13

Table of contents

2.7.5 Speed with current control 14
2.8 Ramp generator 14
2.8.1 Voltage clamping during fast braking 15
2.9 PWM schemes 15
2.10 Phase advance angle 16
2.11 Position interface 17
2.12 Measurements 19
2.12.1 Analog input measurements 20
2.12.2 DC-link current measurement 20
2.12.3 DC-link voltage measurement 20
2.12.4 Potentiometer measurement 20
2.12.5 Digital input measurement 21
2.12.6 Floating prescaler 22
2.12.7 Speed measurement 23
2.13 Voltage compensation 25
2.14 Hall pattern learning 25
2.14.1 Configuration for Hall learning 25
2.14.2 Hall learning illustration 25
2.15 PI controller 27
2.16 Protection 28
2.16.1 Trap protection 28
2.16.2 Wrong Hall protection 28
2.16.3 Hall learning protection 28
2.16.4 Motor stall protection 28
2.16.5 DC-link overvoltage/undervoltage protection 28
2.16.6 SPI fault protection 29
2.16.7 Gate driver reset protection 29
2.17 Controlling the MOTIX™ 6EDL7141 smart gate driver 29
2.18 Supporting GUI oscilloscope 29
3 Graphical user interface 30
References 31
Revision history 32

1 Introduction

This document provides relevant information about three-phase brushless DC (BLDC) motor block commutation control using Hall sensors on Infineon's (XMC1400 + MOTIX™ 6EDL7141) platform. The example firmware is offered as a ModusToolbox™ (MTB) project and consists of a BSP (Board Supported Package) and a code example. The example project consists of three-phase BLDC motor control-related algorithm firmware, primarily targeted at end applications such as battery-powered motor drives. The example project will provide a high level of configurability and modularity to address different segments.

With fundamental knowledge of this example project, it can also be adapted to IMD7xx series integrated controllers with minor modifications.

1.1 Installation

To quickly verify the firmware and benefit from the solution provided, it is recommended that you install the following software:

- BPA motor control graphical user interface (GUI)
  - A GUI providing easy configurations to quickly get your motor running
- ModusToolbox™ Tools Package
  - ModusToolbox™ is a set of multi-platform development tools and a comprehensive suite of GitHub®-hosted firmware libraries. Together, they enable an immersive development experience for customers creating converged MCU and Wireless systems.

1.2 Directories and files

The folder structure of the firmware is shown in **Figure 1**.

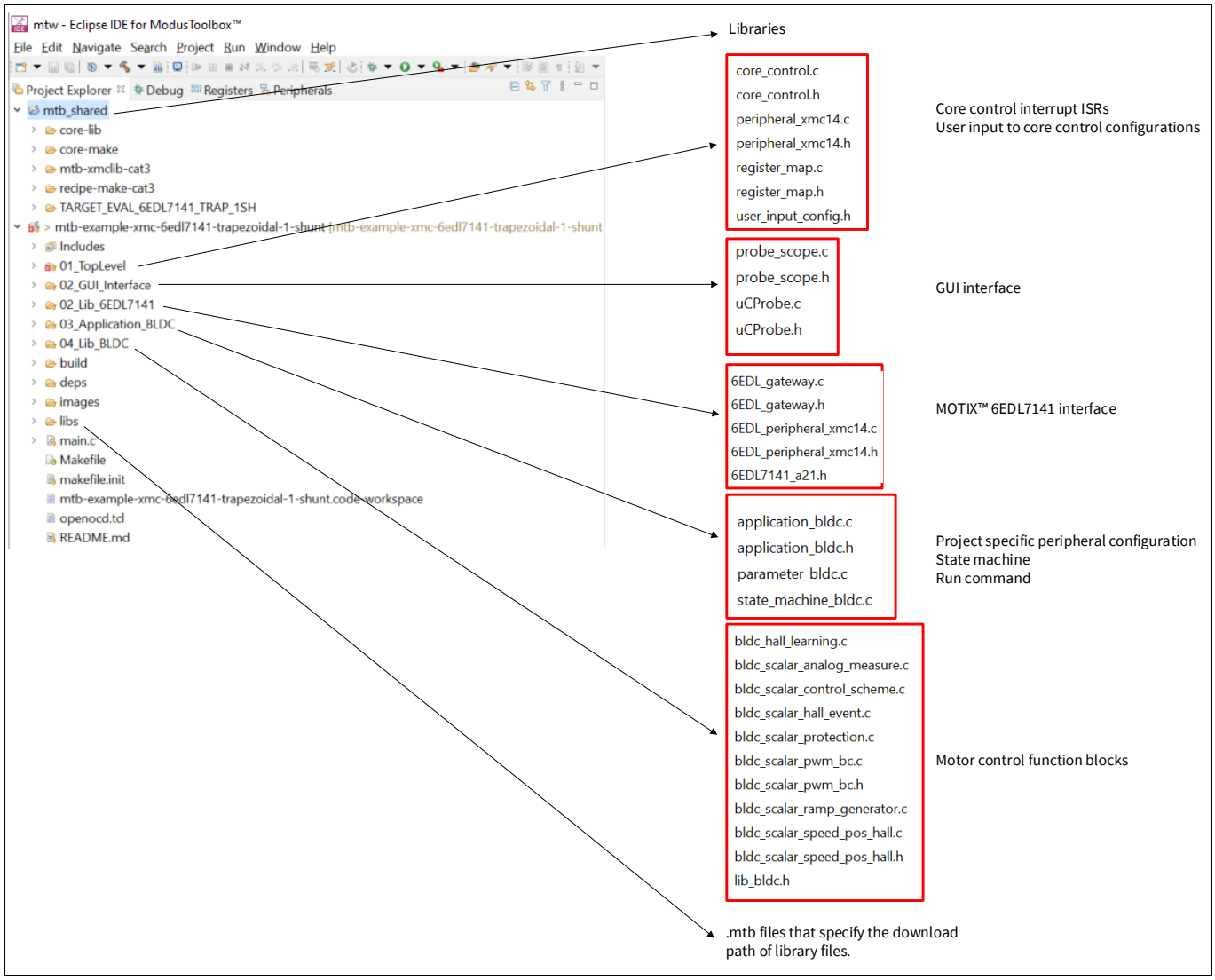


Figure 1 File/folder structure

2 Architecture

2.1 Block diagram

The major modules of this example software are shown in **Figure 2**.

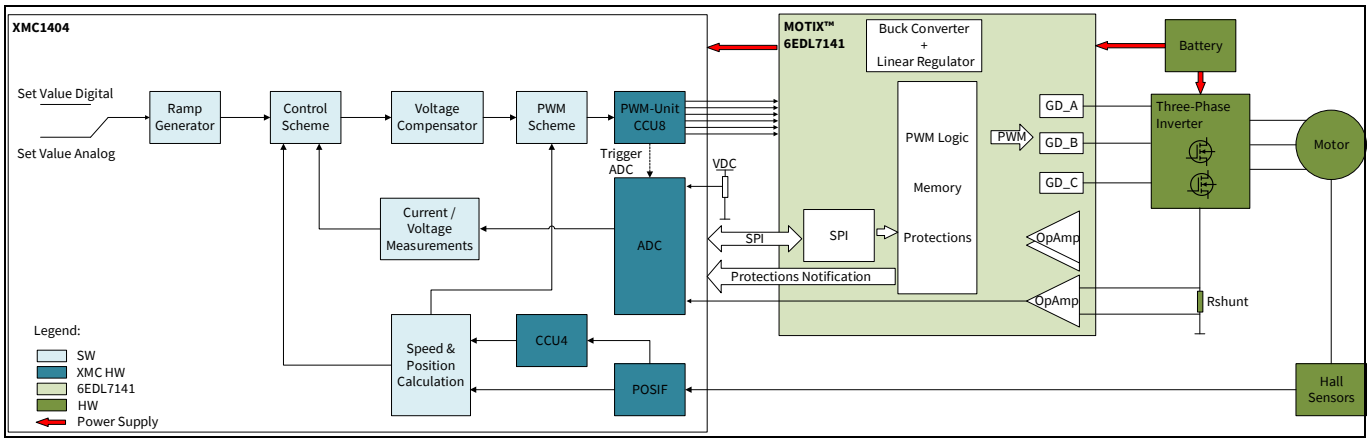


Figure 2 Block diagram

2.2 XMC™ resource allocation

The XMC™ controller resource allocations including pin assignment and peripherals are listed in **Tables 1** and **2**.

Table 1 XMC1404-VQFN64-200kB MCU pin assignment

Port	VQFN64	EVAL_6EDL7141_TRAP_1SH schematic	Configuration	Description
P0.0	41	MOSI_P0_0	USIC1_CH1.DOUT0, ALT9	SPI MOSI with MOTIX™ 6EDL7141
P0.1	42	MISO_P0_1	USIC1_CH1.DX0B	SPI MISO with MOTIX™ 6EDL7141
P0.2	43	–	–	–
P0.3	44	SCLK_P0_3	USIC1_CH1.SCLKOUT, ALT8	SPI SCLK with MOTIX™ 6EDL7141
P0.4	45	NSCS_P0_4	USIC1_CH1.SELO0, ALT8	SPI CS with MOTIX™ 6EDL7141
P0.5	46	–	–	–
P0.6	47	–	–	–
P0.7	48	XMC_EN_DRV_P0_7	GPIO output	EN_DRV to MOTIX™ 6EDL7141
P0.8/RTC_XTAL1	51	–	–	–
P0.9/RTC_XTAL2	52	–	–	–
P0.10/XTAL1	53	–	–	–

Port	VQFN64	EVAL_6EDL7141_TRAP_1SH schematic	Configuration	Description
P0.11/XTAL2	54	–	–	–
P0.12	55	XMC_nFAULT_P3_4	(Reserved)	(Reserved)
P0.13	56	–	–	–
P0.14	57	SWDIO/TMS_P0_14	SWDIO	SWDIO
P0.15	58	SWDCLK/TCK_P0_15	SWDCLK	SWDCLK
P1.0	34	INHC	PWM output	High-side gate pulse – phase W
P1.1	33	INLC	PWM output	Low-side gate pulse – phase W
P1.2	32	XMC_AZ_P1_2	GPIO output	Auto-Zero to MOTIX™ 6EDL7141
P1.3	31	XMC_nBRAKE_P1_3	GPIO output	nBRAKE to MOTIX™ 6EDL7141
P1.4	30	–	–	–
P1.5	29	–	–	–
P1.6	28	–	–	–
P1.7	27	–	–	–
P1.8	26	–	–	–
P2.0	9	BEMF_W_P2_0	–	BEMF_W (reserved)
P2.1	10	BEMF_V_P2_1	–	BEMF_V (reserved)
P2.2	11	BEMF_U_P2_2	–	BEMF_U (reserved)
P2.3	12	BAT_SENSE_P2_3	Analog input, G1CH5	Battery voltage measurement
P2.4	13	Temp_Sense_P2_4	Analog input, G1CH6	Temperature measurement
P2.5	14	Speed_Control_P2_5	Analog input, G1CH7	Potentiometer measurement
P2.6	15	P2_6	Analog input, G0CH0	Kit selection. Sample the ADC value from resistor divider and distinguish the current board. Available boards are: EVAL_6EDL7141_TRAP_1SH_v2 EVAL_6EDL7141_FOC_3SH_v1 EVAL_IMD700A_FOC_3SH_v1 … (tbc)
P2.7	16	BRAKE_XMC_P2_7	GPIO input	(Reserved)
P2.8	17	–	–	–
P2.9	18	–	–	–
P2.10	19	Amp_iV_P2_10_XMC	Analog input, G1CH2	DC-link current measurement
P2.11	20	–	–	–
P2.12	21	–	–	–
P2.13	22	XMC_EN_DRV_CLK_2_13	GPIO output	Watchdog clock output to MOTIX™ 6EDL7141
P3.0	36	INLB	PWM output	Low-side gate pulse – phase V
P3.1	37	INHB	PWM output	High-side gate pulse – phase V

Port	VQFN64	EVAL_6EDL7141_TRAP_1SH schematic	Configuration	Description
P3.2	38	INLA	PWM output	Low-side gate pulse – phase U
P3.3	39	INHA	PWM output	High-side gate pulse – phase U
P3.4	40	XMC_nFAULT_P3_4	ERU1.2A3	nFAULT signal from MOTIX™ 6EDL7141
P4.0	59	–	–	–
P4.1	60	HALL0_P4_1	GPIO input	Hall sensor input – phase U
P4.2	61	HALL1_P4_2	GPIO input	Hall sensor input – phase V
P4.3	62	HALL2_P4_3	GPIO input	Hall sensor input – phase W
P4.4	63	PC_RX_P4_4	–	UART to PC
P4.5	64	PC_TX_P4_5	–	UART to PC
P4.6	3	–	–	–
P4.7	4	AUX_LED0_P4_7	GPIO output	LED, Hall signal indication
P4.8	5	AUX_LED1_P4_8	GPIO output	LED, status indication
P4.9	6	–	–	–
P4.10	7	DirectionSwitch_P4_10	GPIO input	Motor direction
P4.11	8	–	–	–

Table 2 List of XMC™ peripherals

No	Resource	Purpose	Firmware configuration
1	CCU41_CC41	Used for multi-channel pattern synchronization	Edge-aligned, single-shot mode, enable period match interrupt to enablecommutation.
2	CCU41_CC42	Used for motor speed capture	Edge-aligned, capture and clear, capture the time between Hall events
3	POSIF1	Hall and MCM configuration	Hall interface, refer to CCU41
4	CCU81_CC80	PWM generation – phase W	Center-aligned
5	CCU81_CC81	PWM generation – phase U	Center-aligned, enable one match interrupt
6	CCU81_CC82	PWM generation – phase V	Center-aligned
7	VADC GroupA QueueB	DC-link current, DC-link voltage, potentiometer measurement and user-defined measurement	Triggered by CCU81_CC82 (phase V) channel 2 compare match
8	NVIC	Used for ISRs	–
9	SYSTICK	Used for state machine	1 ms systick interrupt

2.3 Managing parameters

There are two types of parameters:

- User input – user input parameters are usually presented in engineering units (V, A, RPM, etc.), which the user can easily configure based on application need.
- Internal parameters – internal parameters are actual parameters being used by the firmware. Most internal parameters use different scaling for code execution efficiency purposes, and so may not be easily

understood. One key feature of the GUI is to calculate internal parameters from user input parameters. Meanwhile a conversion function is also available in firmware (but is commented out). In the firmware, user inputs are #define in header file 01_TopLevel/user_input_config.h, and the conversion function is in the c source file, 03_Framework_BLDC/parameter_bldc.c.

All motor control-related internal parameters are grouped in the data structure *MotorParam*, and members of *MotorParam* must be configured correctly before the motor can run. Use the GUI to enter user inputs and let the GUI calculate the internal parameter value, then click the “Write Parameter” button to write all internal parameters into the MCU’s RAM. After this step, the motor should be ready to run.

2.4 Memory map

XMC1404-VQFN64-200kB provides 200 kB of Flash memory for instruction code or constant data, starting at address 1000’1000$_H$. This excludes Flash sector 0, which is used to store system information and is always read-only. In the firmware, there are designated locations inside the MCU’s Flash memory for storing internal parameters – these are the MOTIX™ 6EDL7141 registers’ setting and motor control parameters, shown in **Table 3**. The user may want to save parameters into the MCU so that next time, when the IC powers up, the motor is ready to run immediately. Click the GUI’s “Write to Flash” button to store internal parameters into Flash memory.

For a complete memory map, see the **XMC1400 reference manual**.

Table 3 Memory map

Address space	Address range	Description
Code	10001000_H to $1000FFFF_H$	Flash (60 kB)
	10010000_H to $100100FF_H$	Store MOTIX™ 6EDL7141 registers’ settings (256 bytes)
	10010100_H to $100101FF_H$	Store motor control parameters (256 bytes)
	10010200_H to $10032FFF_H$	Flash (139.5 kB)

2.5 Interrupts

The complete interrupt table is shown in **Table 4**.

Table 4 Interrupt table

Interrupt	Function name	Interval	Service request source	Node ID	Priority
Commutation ISR	*Commutation_ISR*	Asynchronous	CCU41.SR0	21 (IRQ21_IRQn)	0
Hall event ISR	*Motor0_BLDC_SCALAR_HallEvent_ISR*	Asynchronous	POSIF1.SR0	27 (IRQ27_IRQn)	1
Control-loop ISR	*Motor0_BLDC_SCALAR_ControlLoop_ISR*	1/PWM frequency	CCU81.SR0	25 (IRQ25_IRQn)	2
Systick ISR	*SysTick_Handler*	1 ms	–	-1 (SysTick_IRQn)	3

2.5.1 Commutation ISR

The multi-channel shadow transfer event of POSIF is used as commutation ISR. It is triggered by a period match of CCU41 slice 1. This ISR function is also reused in other places where the multi-channel pattern needs to be immediately updated.

ISR tasks

- Set PWM sector according to Hall sector.
- Get the shadow multi-channel pattern value and apply the PWM modulation pattern.

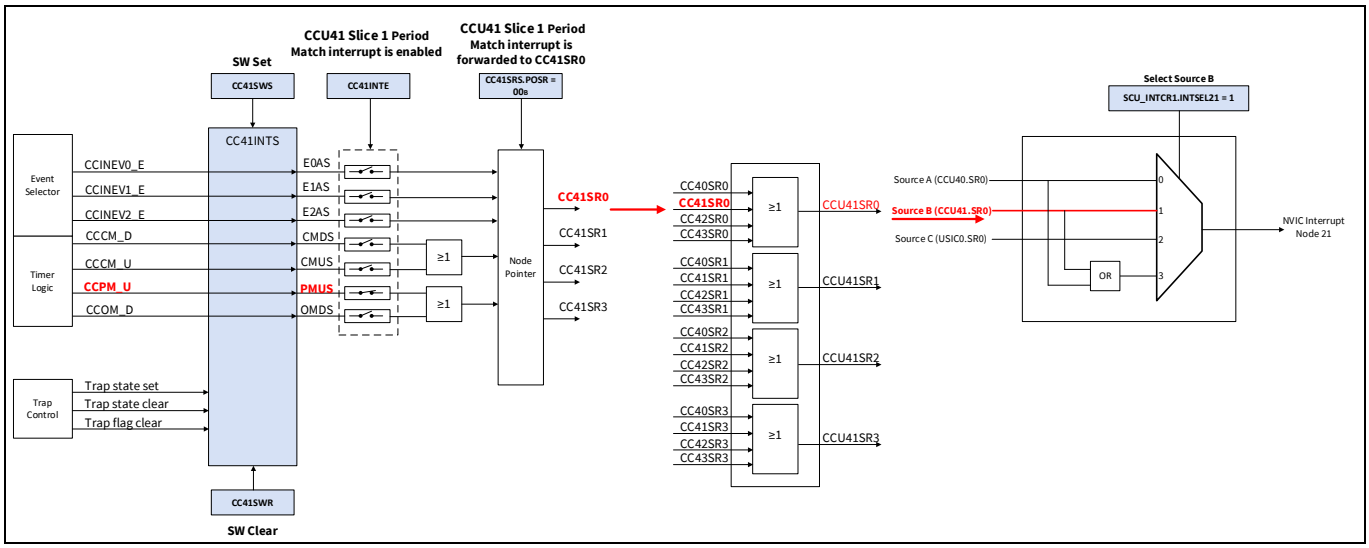


Figure 3 Source selection for NVIC interrupt node 21

2.5.2 Hall event ISR

Update of the Hall inputs interrupt.

ISR tasks

- Update new Hall pattern and read capture value from CCU4 capture slice, CCU41 slice 2.
- Report wrong Hall fault if Hall pattern is unexpected value.
- Prepare Hall and multi-channel pattern for the next commutation.
- Calculate the speed based on captured value.

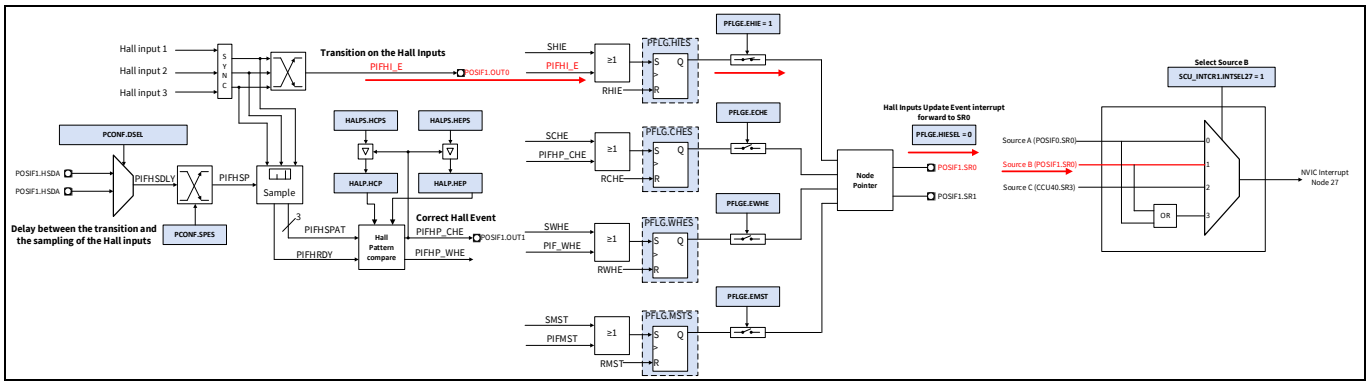


Figure 4 Source selection for NVIC interrupt node 27

2.5.3 Control-loop ISR

Control-loop ISR is based on the PWM frequency. The one match of the phase U PWM slice is used as the control-loop ISR.

ISR tasks

- Reads direct DC-link current.
- Reads DC-link voltage.
- Checks DC-link overvoltage/undervoltage (OV/UV).
- Executes the configured control scheme and updates duty cycle.
- uCProbe scope sampling.

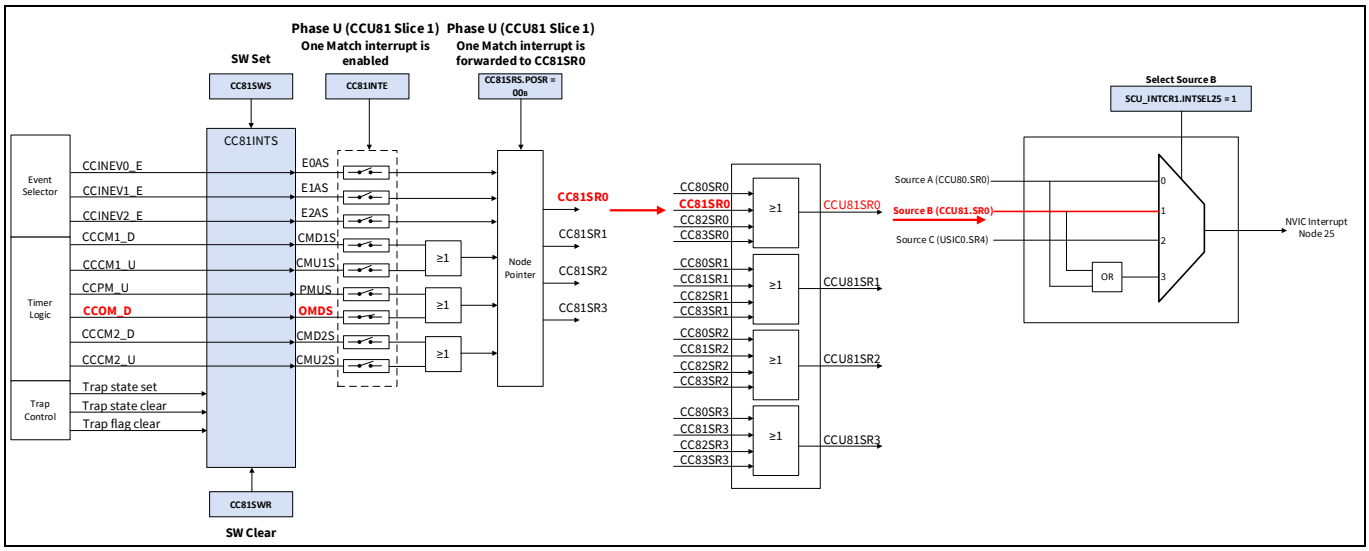


Figure 5 Source selection for NVIC interrupt node 25

2.5.4 Systick ISR

Systick ISR is used as a scheduler.

ISR tasks

- Calculates temperature with filter.
- States machine.
- Toggles status LED.
- Handles write request to MOTIX™ 6EDL7141 from GUI or control.
- Updates MOTIX™ 6EDL7141 GUI monitor value.
- Refreshes variables by reading from MOTIX™ 6EDL7141.
- Controls EN_DRV and nBRAKE pin output.
- Toggles EN_DRV_CLK pin as 500 Hz MOTIX™ 6EDL7141 watchdog clock.

2.6 State machine

The motor control state machine is executed from a systick interrupt service routine. The state chart flow is shown below.

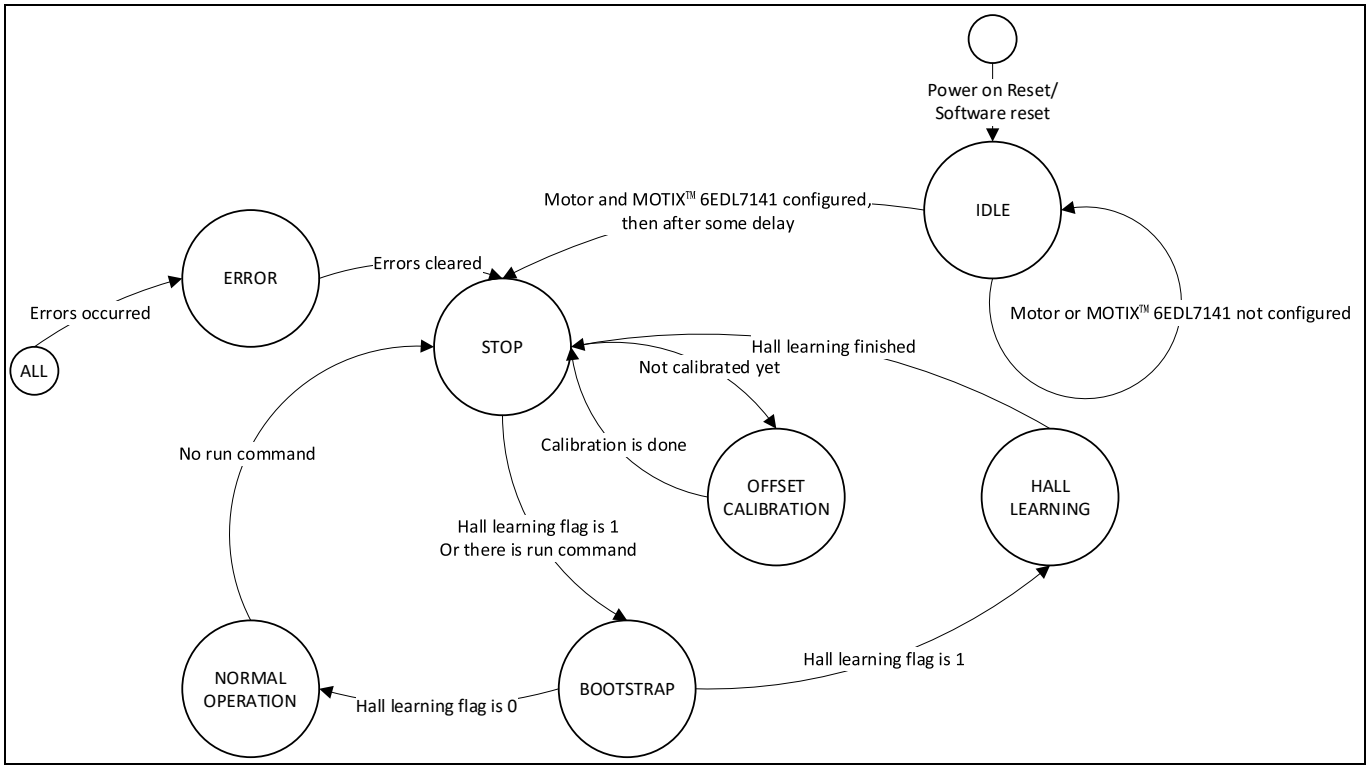


Figure 6 State machine flow chart

The state transition matrix is shown in **Table 5**.

Table 5 State machine transition matrix

State	State functionality	Transition event	Next state
IDLE	Check to ensure motor and MOTIX™ 6EDL7141 parameters are configured Calculate derived parameters, initialize XMC™ peripheral, update Hall pattern and Hall sector	Motor parameters configured and MOTIX™ 6EDL7141 parameters configured Initialization completed	STOP
		Trap error SPI communication fault, OV/UV error	ERROR
STOP	Update parameters Check offset calibration status	Offset not calibrated yet	OFFSET CALIBRATION
		Hall learning flag is active Or there is run command directly from GUI or from potentiometer	BOOTSTRAP
		Trap error Wrong Hall event MOTIX™ 6EDL7141 reset fault SPI communication fault OV/UV error	ERROR
OFFSET CALIBRATION	Do offset calibration	Offset calibration completed	STOP
		Trap error Wrong Hall event MOTIX™ 6EDL7141 reset fault	ERROR

State	State functionality	Transition event	Next state
		SPI communication fault OV/UV error	
BOOTSTRAP	Bootstrapping per phase	Bootstrapping completed and Hall learning flag is 1	HALL LEARNING
		Bootstrapping completed and Hall learning flag is 0	NORMAL OPERATION
		Trap error Wrong Hall event MOTIX™ 6EDL7141 reset fault SPI communication fault OV/UV error	ERROR
HALL LEARNING	Hall pattern identification algorithm	Hall pattern identified successfully	STOP
		Trap error Wrong Hall event MOTIX™ 6EDL7141 reset fault SPI communication fault OV/UV error Timeout – not able to detect pattern	ERROR
NORMAL OPERATION	Control loop execution Ramp-up/-down Stall detection	Stop command received	STOP
		Trap error MOTIX™ 6EDL7141 reset fault SPI communication fault Wrong Hall event OV/UV error Stall detection	ERROR
ERROR	Set stop command, wait for errors to be cleared	Errors are cleared	STOP

2.7 Control schemes

The firmware is designed to control the three-phase BLDC motor on one of the following control schemes: voltage control, speed control, current control, speed with current control. A different control scheme can be switched at motor stop state by changing value in *MotorParam.ControlScheme*.

2.7.1 Control scheme configuration

MotorParam.ControlScheme – this variable can configure the control scheme, as below:

Voltage control – *VOLTAGE_CONTROL*

Speed control – *SPEED_CONTROL*

Current control – *CURRENT_CONTROL*

Speed with current control – *SPEED_CURRENT_CONTROL*

2.7.2 Voltage control

Voltage control provides a behavior that is comparable to a brushed DC motor. Although the position of the stator field is controlled to be synchronous with the rotor by the use of Hall sensors, speed and torque occur as a result of the construction of the individual motor and the mechanical load. Voltage setpoint input will connect to ramp output (if ramp is enabled) or user-set value/analog input based on configuration.

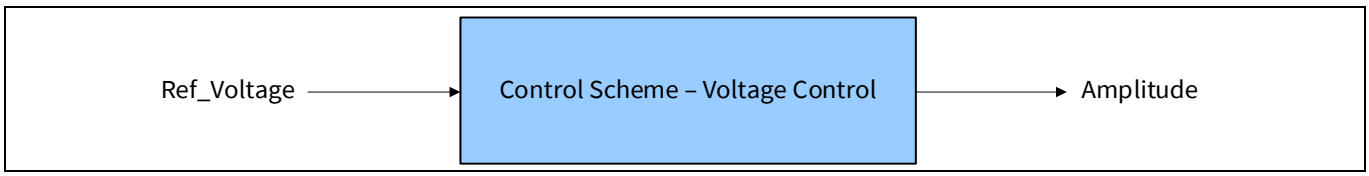


Figure 7 Control scheme – voltage control

Ramp.output is the variable used for the Ref_Voltage.

2.7.3 Speed control

The speed control scheme is a closed-loop control, which adjusts the voltage according to the speed reference value. In the case of dynamic load changes, the voltage at the motor is adjusted automatically and the speed is maintained at a constant. The actual speed value is derived from the Hall sensor/phase voltage. Speed setpoint input will connect to ramp output (if ramp is enabled) or user-set value/analog input based on configuration.

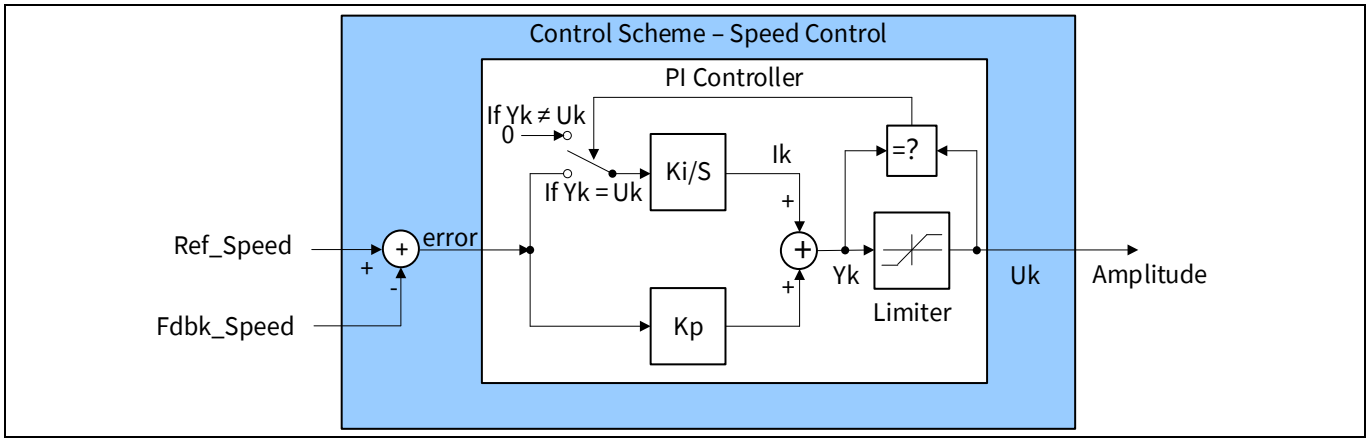


Figure 8 Control scheme – speed control

Ramp.output is the variable used for the Ref_Speed; *Motor0_BLDC_SCALAR.motor_speed* is the variable used for the Fdbk_Speed.

2.7.4 Current control

The current control scheme requires current measurement feedback and adjusts the voltage according to the required torque. With dynamic loads, the speed will vary, but the torque will remain constant. Current setpoint input will connect to ramp output (if ramp is enabled) or user-set value/analog input based on configuration.

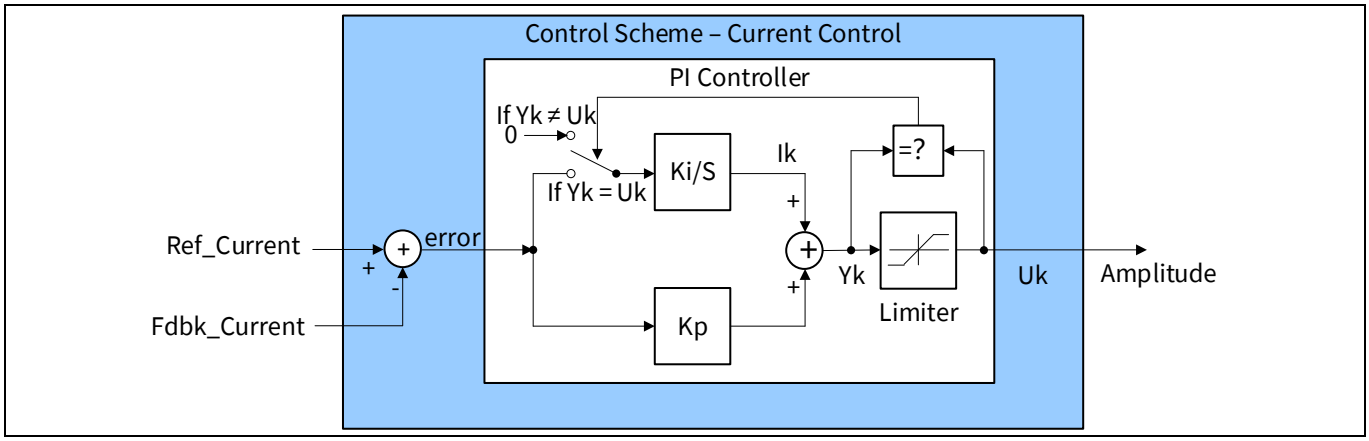


Figure 9 Control scheme – current control

Ramp.output is the variable used for the Ref_Current; *MotorVar.MotorCurrent* is the variable used for the Fdbk_Current.

2.7.5 Speed with current control

The speed with torque control scheme provides a cascaded control scheme, where the inner control loop adjusts the current (torque) by changing the voltage at the motor, and the outer control loop provides the current reference value in order to control the speed. Speed setpoint input will connect to ramp output (if ramp is enabled) or user-set value/analog input based on configuration.

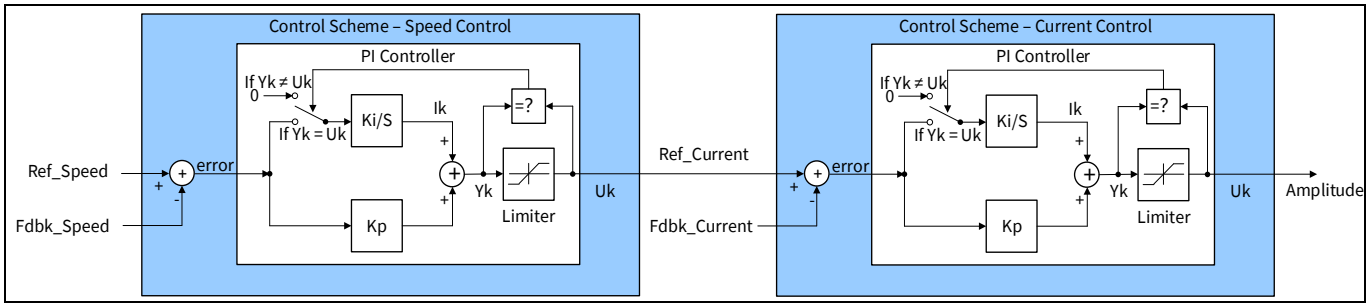


Figure 10 Control scheme – speed with current control

2.8 Ramp generator

The firmware provides a ramp function, which ramps input parameters from initial value to end value in a linear curve. Start and end values are configurable. Ramp generator input will be connected to user-set value or analog input based on configuration.

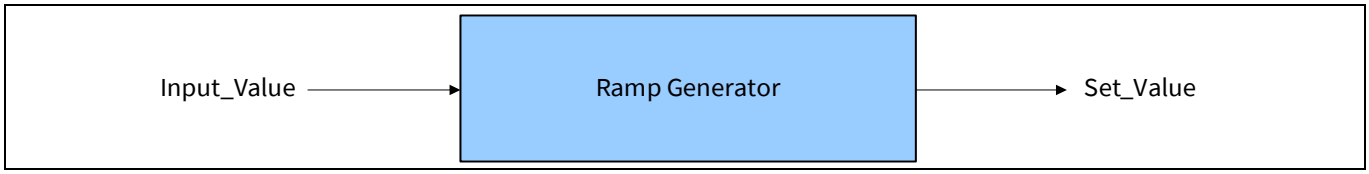


Figure 11 Ramp generator

2.8.1 Voltage clamping during fast braking

The example firmware is provided with a feature called “voltage clamping during fast braking/ramp-down”.

Basically, during ramp-down the DC-link voltage is monitored. The ramp-down happens only if the DC-link voltage is found below the user-configured voltage limit. The voltage limit can be configured by modifying *RAMP_DOWN_MAX_VOLTAGE*, which has a default value of 28 V.

2.9 PWM schemes

Table 6 explains three bits’ setting that can combine to make a PWM scheme. A total of eight PWM schemes can be derived which can be applied in BLDC motor control.

Table 6 PWM scheme bit setting

Bit setting	**= 0**	**= 1**
DiodeRect (bit 0)	Complementary switching	Non-complementary switching
LowSideSw (bit 1)	High-side switching	Low-side switching
SixtyDegPwm (bit 2)	120-degree PWM	60-degree PWM

In the firmware, three of these eight PWM schemes are actually provided as an example:

- PwmType = 0 (high-side modulation with synchronous rectification)
- PwmType = 2 (low-side modulation with synchronous rectification)
- PwmType = 4 (alternative modulation with synchronous rectification)

These three PWM schemes are further illustrated in **Figure 12**.

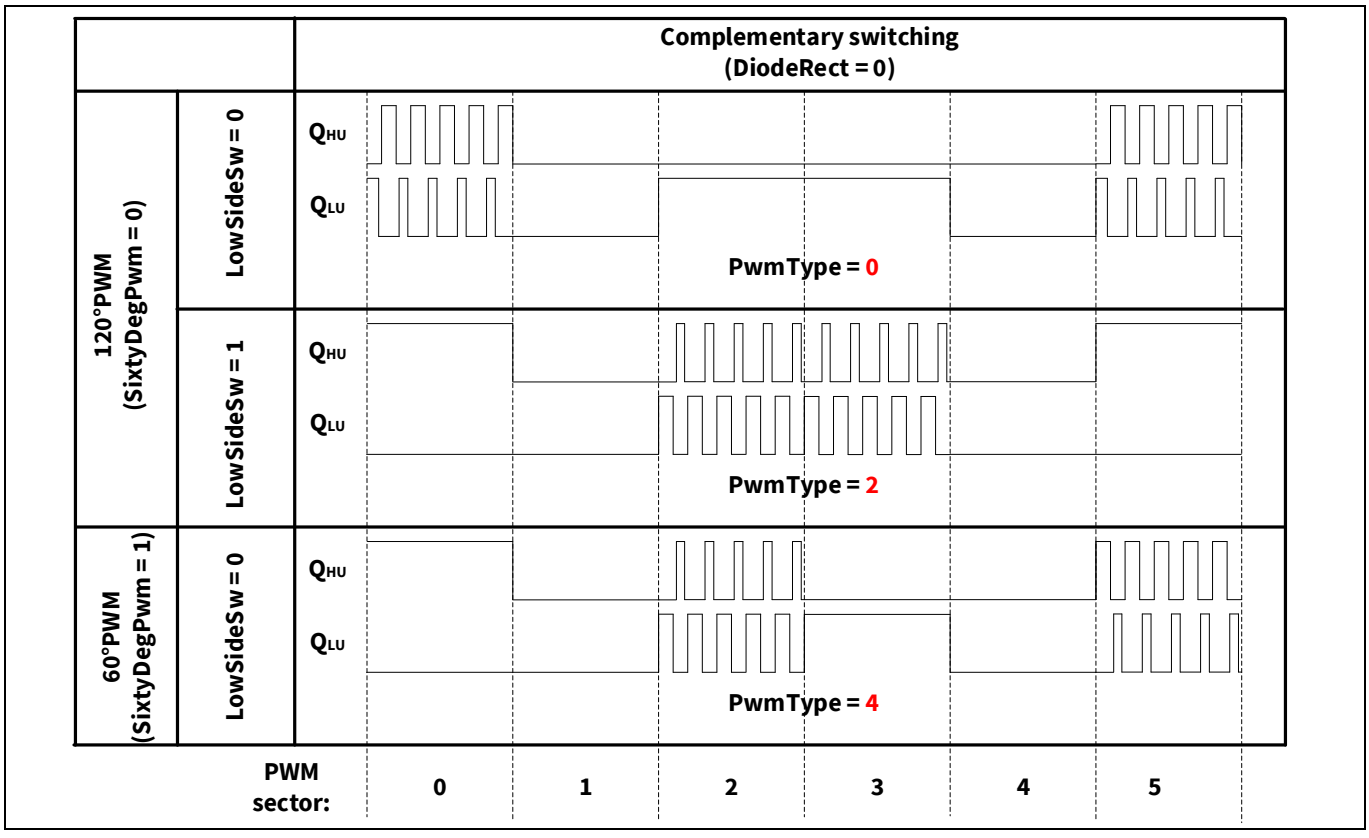


Figure 12 PWM scheme

Table 7 shows the actual firmware variable settings corresponding to PwmType in **Figure 12**.

Table 7 Firmware variables corresponding to the PWM scheme

MotorParam.PwmModulationType	MotorVar.PwmScheme	PwmType	Scheme explanation
0 (default)	0 (default)	0 (default)	PWM_HIGHSIDE_SYNCREC
1	2	2	PWM_LOWSIDE_SYNCREC
2	4	4	PWM_ALTERNATE_SYNCREC

2.10 Phase advance angle

Phase advance can be adopted to achieve higher efficiency and an extended speed range.

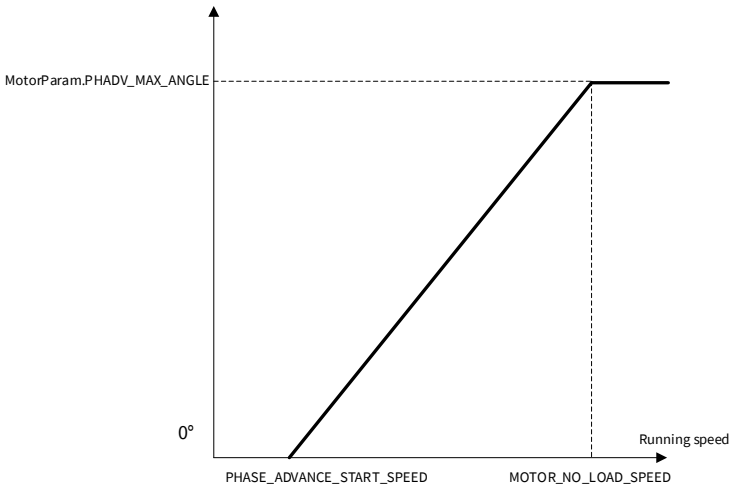


Figure 13 Phase advance angle

- Set *PHASE_ADVANCE_MAX_ANGLE* to a non-zero value to enable phase advance.
- When motor speed is above *PHASE_ADVANCE_START_SPEED,* phase advance will become active, the phase advance angle is linearly adjusted according to running speed.
- Commutation delay is adjusted according to actual speed and *PHASE_ADVANCE_MAX_ANGLE*, then this commutation delay is applied to the period value of CCU41 slice 1. This changes the moment that the new multi-channel pattern is applied. Refer to **2.5.1**.
- PWM sector will also be advance by one if phase advance is active with direction being considered, and not if it is inactive.

PHASE_ADVANCE_START_SPEED – Q14, defines phase advance starting speed, phase advance angle increases linearly above this speed.

PHASE_ADVANCE_MAX_ANGLE – defines maximum phase advance angle at maximum speed. Set this to 0 to disable phase advance.

Note: Always set PHASE_ADVANCE_START_SPEED < MOTOR_NO_LOAD_SPEED.

PHASE_ADVANCE_MAX_ANGLE should be < 60 °

2.11 Position interface

The POSIF module of the XMC™ device is used as a position interface. The POSIF module is responsible for acquiring the position and speed of the motor. It is also responsible for updating the commutation pattern.

In a BLDC motor three Hall sensors separated by 120 degrees are used to get the motor position and speed. Hall sensor inputs are interfaced to the POSIF module of the XMC™ device.

Profile 1 of the Hall sensor mode in the XMC™ device is used here (for more detail about profile 1 refer to the XMC™ reference manual), where two CCU4 slices are used.

1. One slice is used to acquire position and generate a signal for the commutation pattern update (MCM update). In the firmware it is CCU41 slice 1.

2. A second slice is used for speed measurement. In the firmware it is CCU41 slice 2.

The rising-edge ST signal of CCU41 slice 1 is used for Hall resampling. The time delay for the rising-edge ST signal is the blanking time. This helps to avoid any noise in the Hall signal.

The period value of the CCU41 slice 1 timer is used to control the phase delay of commutation. After the phase delay period, the commutation pattern is updated in sync with the PWM signal.

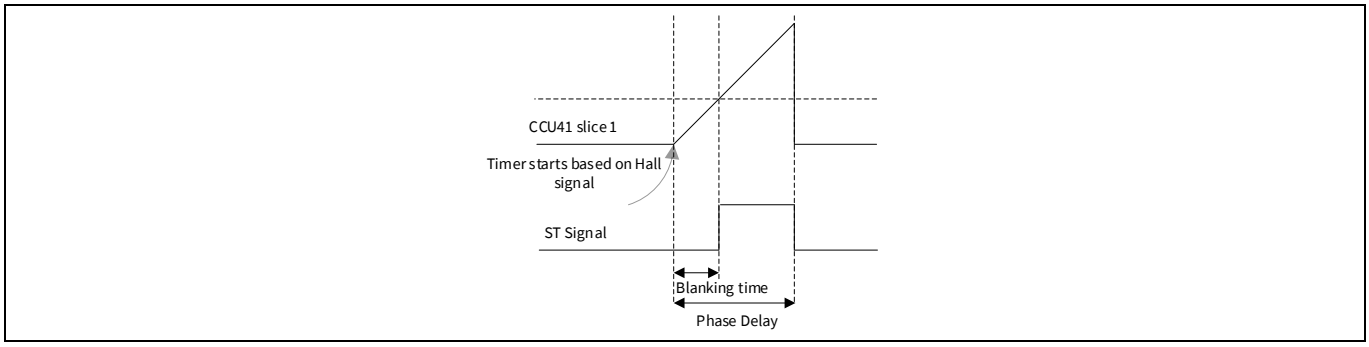


Figure 14 CCU4 timer for Hall edge detection

Hall input pin selection is shown in **Figure 15**.

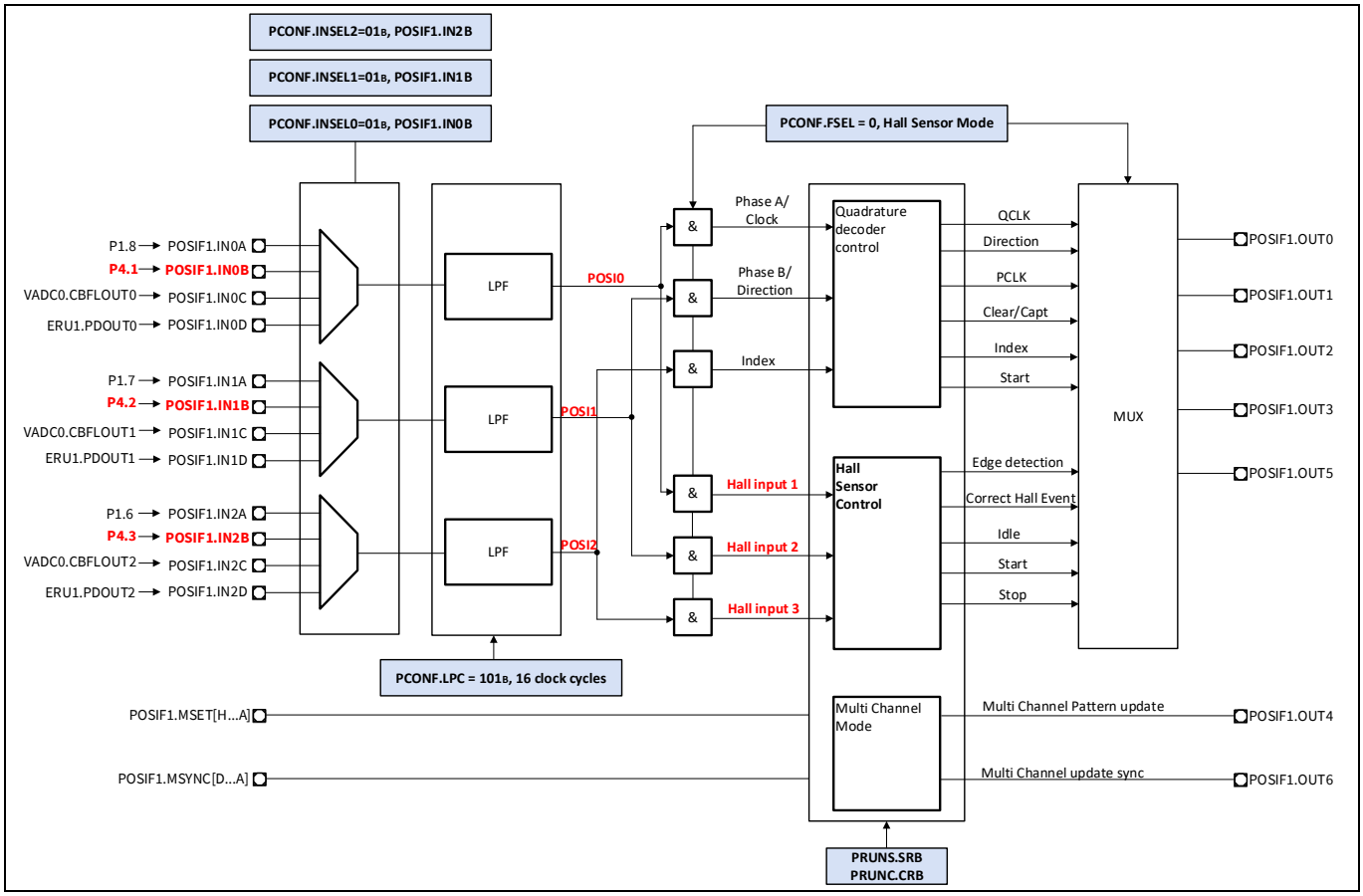


Figure 15 Hall input selector diagram

The whole Hall sensor timing diagram is shown in **Figure 16**. Please note:

- The firmware is realized by enabling CCU41 slice 1 period match interrupt or named commutation ISR , then updating the compare value and enabling Multi-Channnel pattern by setting the MCMS.MNPS bit in the ISR routine. The update is not done immediately due to the fact that the trigger that synchronizes the update with the PWM is still needed, it is done when one match event happens.
- POSIF1.OUT2/POSIF1.OUT3 is here to show how wrong Hall event (WHE) can be used to generate an IDLE signal and stop the PWM generation unit automatically. In the firmware, PCONF.HIDG bit is actually set to 1, the generation of IDLE signal is disabled and WHE is detected manually.

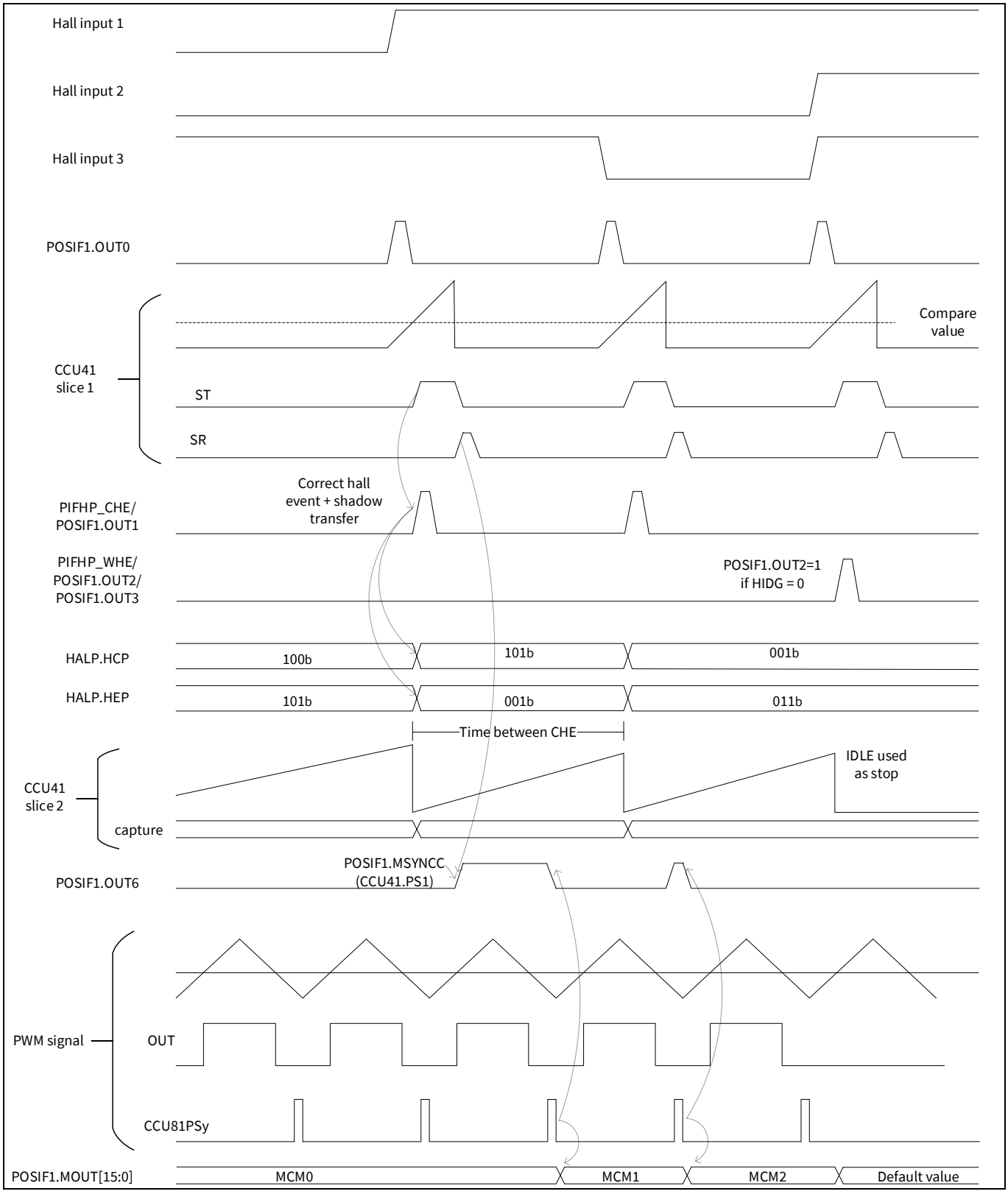


Figure 16 Hall sensor timing diagram

2.12 Measurements

Motor parameter measurements can be classified into analog input-based measurements and digital input-based measurements.

2.12.1 Analog input measurements

The current, voltage, and potentiometer measurements are under analog input measurements.

The queue request source of VADC is used for analog input measurements. The order of queue filling starts from current, voltage, and potentiometer based on priority.

The VADC resolution is fixed to 12 bits. The VADC conversion is based on an external trigger signal, which is phase V compare match event (triggered during down-count). The trigger delay can be set by macro *CURRENT_TRIGGER_DELAY* in µs.

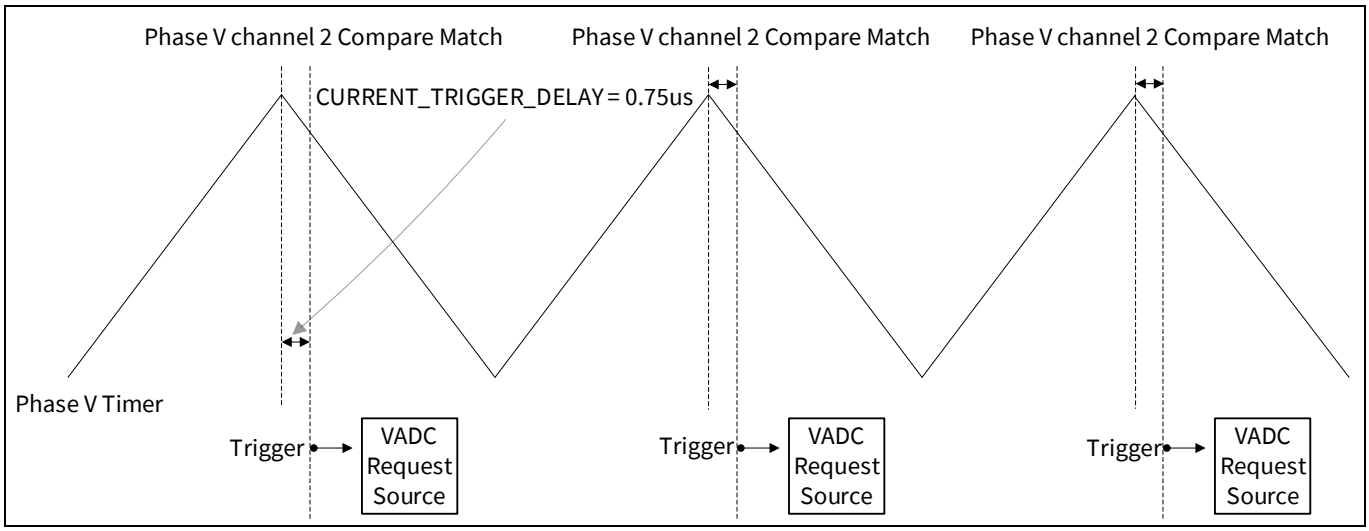


Figure 17 VADC trigger

2.12.2 DC-link current measurement

DC-link current is calculated by subtracting the DC amplifier offset from the ADC value, then scaled up to 2^14-bit format using a precalculated current scale.

IDC_FILTER_TIMECONST macro defines the DC-link current low-pass filter time constant in ms.

DC-link current measurement can be skipped during demagnetization time to avoid a wrong measurement. This duration is configurable by the user. *DEMAG_BLANKING_TIME* is the macro available for this configuration, in µs. It is converted and rounded to numbers of the PWM period.

2.12.3 DC-link voltage measurement

DC-link voltage is calculated by first getting the ADC value, then scaled to 2^14-bit format using a precalculated voltage scale based on the voltage divider value.

VDC_FILTER_TIMECONST macro defines the DC-link voltage low-pass filter time constant in ms.

2.12.4 Potentiometer measurement

The potentiometer value is used to determine the set value of the voltage/current/speed based on the selected control scheme.

The potentiometer usage is in unidirectional control mode: an additional direction switch is used to control the direction. The zero position of the potentiometer is considered as minimum duty cycle value; the maximum position is considered as full duty cycle.

Macro *ANALOG_INPUT_LOW_LIMIT* is provided to configure the analog input low limit, below which potentiometer analog input value is considered as 0. The default value is 5 percent of the full-scale input.

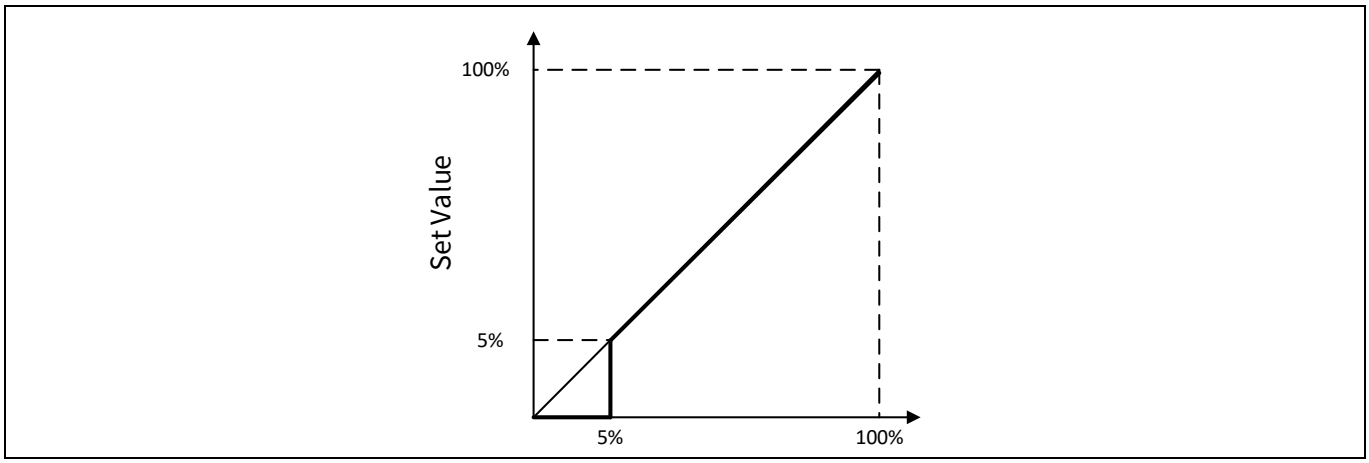


Figure 18 Potentiometer in unidirectional mode

2.12.5 Digital input measurement

The motor speed measurement comes under this section. Motor speed calculation is done using the time interval between two Hall events.

The time interval is captured in XMC™ using the floating prescaler option of CCU4. As mentioned in the position interface section, one CCU4 slice is consumed for speed measurement. The capture event signal is provided by the POSIF module based on the correct Hall edge. The external capture diagram is shown in **Figure 19**.

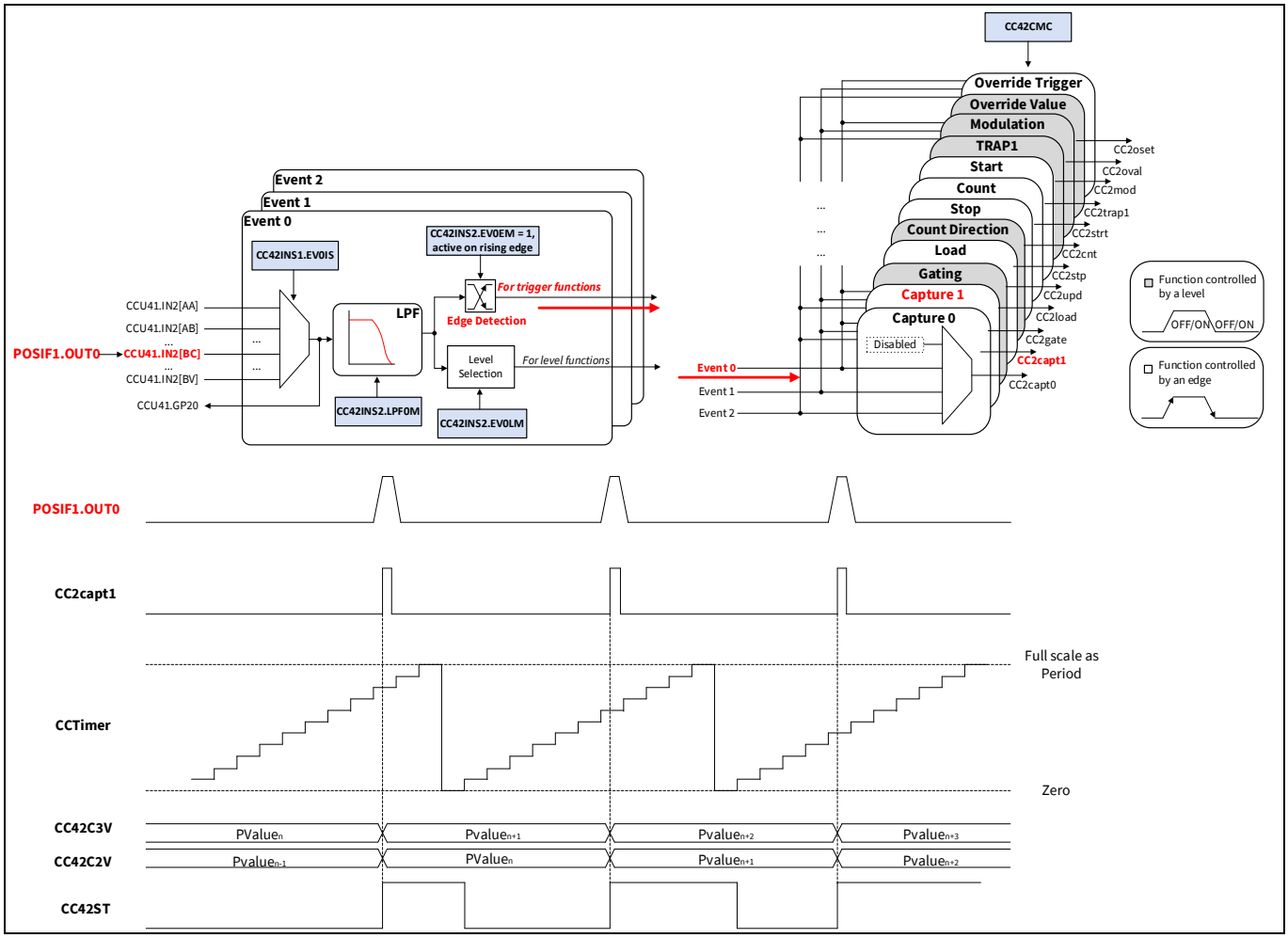


Figure 19 External capture diagram

2.12.6 Floating prescaler

The floating prescaler feature of XMC™ is used to measure a broad range of speeds. Whenever the counter overflows, the prescaler is incremented by 1. As the capture event occurs, the prescaler resets to its initial value.

In the firmware, prescaler initial value (PSIV) is set as 4; this is defined by macro *CCU4_PRESCALER*. An initial capture clock is derived as 96 MHz/2^4 = 6 MHz.

The speed calculation is done by reading the final prescaler and count in each capture event. The captured count in the PSIV unit can be calculated as:

$$\text{capval} = (2^{(final\ prescaler - PSIV)} - 1) * 2^{16} + CC42C3V.CAPTV$$

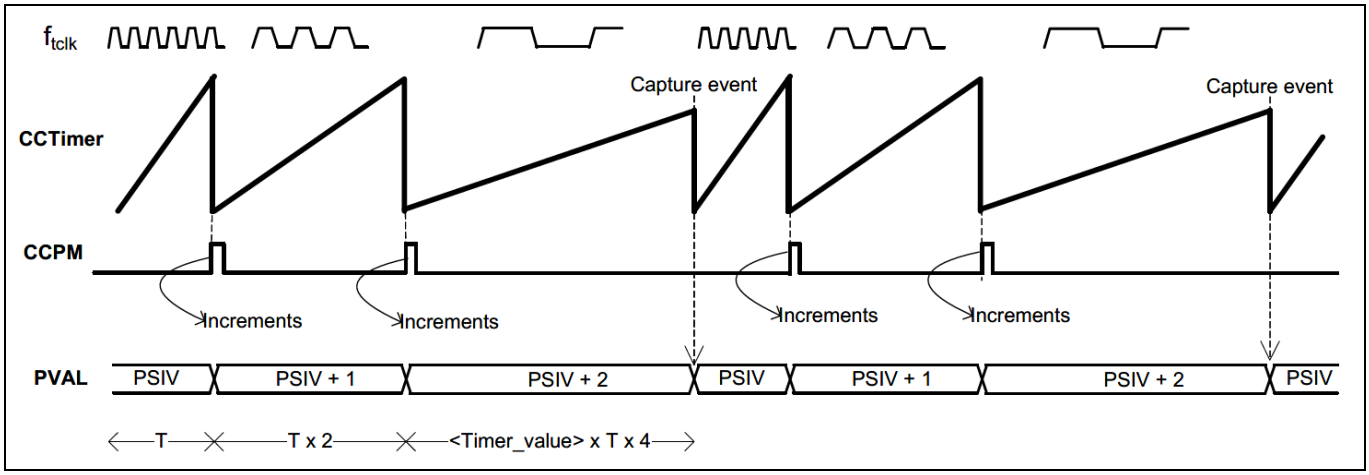


Figure 20 Floating prescaler capture mode

2.12.7 Speed measurement

A capture event is triggered by a correct Hall event, which occurs at every electrical 60 degrees. To avoid errors in speed measurements due to misplaced Hall sensors, a moving average of the last six readings is taken in the calculation, as shown in **Figure 21**.

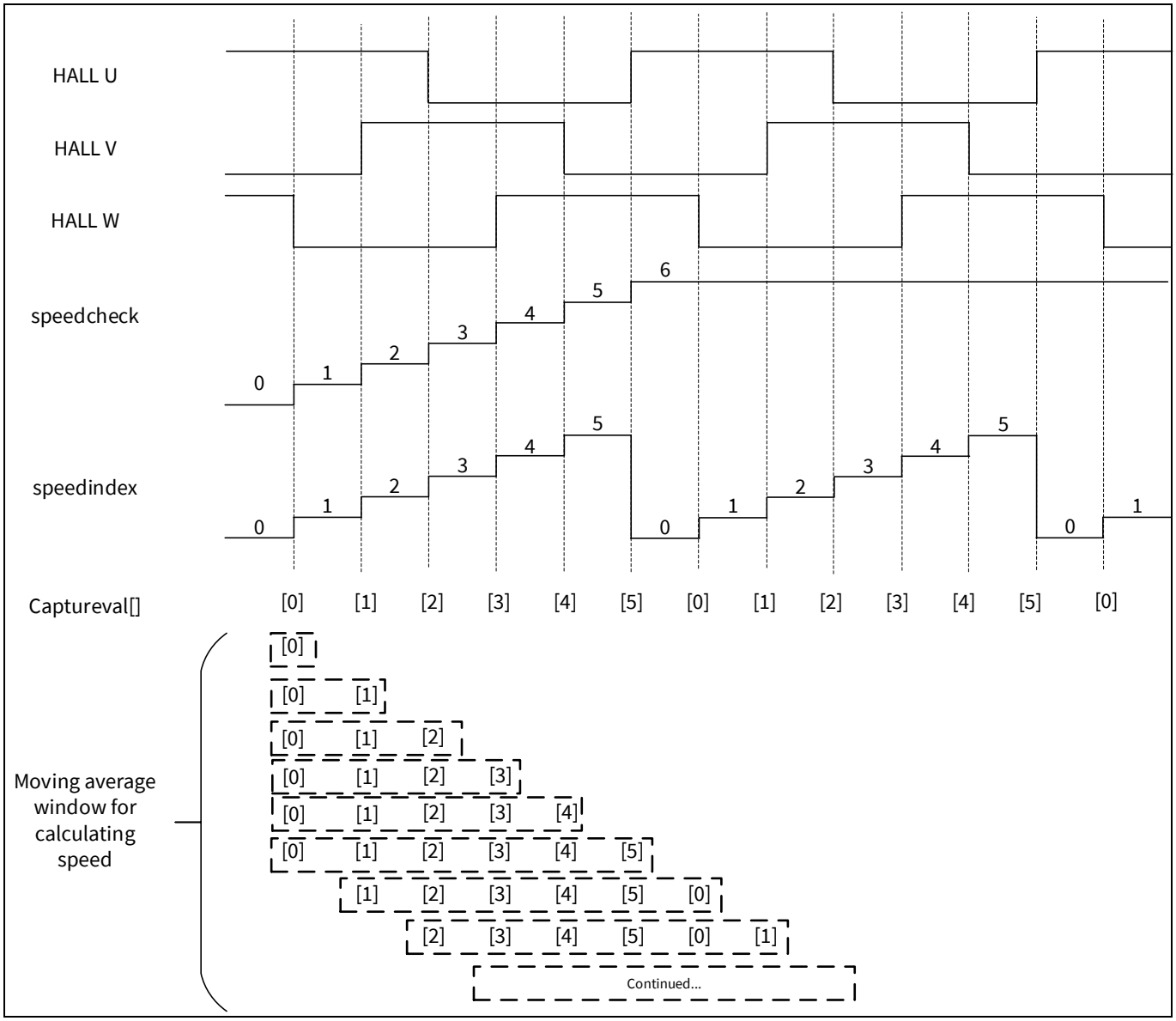


Figure 21 Speed measurement with moving average window

Actual electrical motor speed is calculated in RPM using the speed constant value.

$$\text{Electrical speed in RPM} = \frac{speed_constant}{speedaccum} RPM$$

$$= \frac{10^6 * \frac{96}{2^{PSIV}} * 10 * 6}{Sumoflast6values} \text{RPM}$$

$$= \frac{10^6 * 6 * 10 * 6}{Sumoflast6values} \text{RPM}$$

The mechanical speed is then calculated and converted into 2^14 format using pole-pair and speed scaling.

$$MechanicalSpeed = \frac{\text{ElectricalSpeedInRPM} * 2^{14}}{MOTOR_NO_LOAD_SPEED * Polepair}$$

2.13 Voltage compensation

DC-link voltage is measured every PWM period and the variation in the DC-bus voltage is compensated. The applied voltage based on actual DC-link voltage and configured DC-link voltage is increased or decreased, so effectively voltage applied to the motor will be maintained even if there is a variation in DC-link voltage.

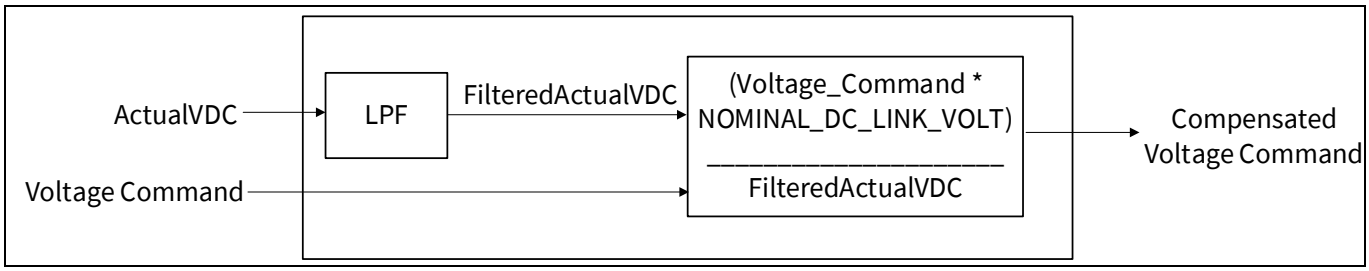


Figure 22 Voltage compensation

2.14 Hall pattern learning

Adaptive Hall learning provides a method to generate a correct six-step commutation pattern for three-phase BLDC motors. In the present version this feature is supported for motors in which three Hall sensors are placed at a relative angle of 120 electrical degrees from each other and their transitions are aligned to the zero crossing of phase-to-phase back EMFs.

To run the Hall-based BLDC motor, a proper sequence of excitation of motor phases is needed with respect to binary code generated from three Hall sensors. The Hall learning technique captures and defines these sequences automatically.

Adaptive Hall learning is achieved by exciting motor phase windings with a pre-defined excitation pattern, aligning the rotor to each commutation sequence and reading the Hall signal code. For successful tuning it is important that the rotor aligns every time to a new applied sequence. This operation is equivalent to forcing the motor to run in an open-loop step-by-step manner, without having continuous rotation. Each commutation pattern is applied to the motor for a defined period. The period is determined according to the open-loop speed and duty cycle of PWM defined by the required voltage.

Required speed and voltage are determined by the load on the motor, so the motor may get locked in a position. At this point, Hall sensor output is read to get the Hall pattern corresponding to the applied commutation pattern. The next commutation pattern is applied to move the motor forward and get the Hall pattern. Repeat the same procedure and capture the required sequences.

2.14.1 Configuration for Hall learning

HALL_LEARNING_VOLTAGE – open-loop voltage to be applied during Hall learning, in percentage of nominal voltage.

HALL_LEARNING_TPULSE – pulse time duration of each sector during Hall learning, in ms.

HALL_LEARNING_PATTERN – Hall learning result. This macro specifies an initial value, and Hall learning must be done to update the value if it does not correspond to the real hardware such as motor phase connection, Hall signal connections, etc.

2.14.2 Hall learning illustration

The whole Hall learning process is explained below and shown in **Figure 23**.

1. The commutation PWM output starts with PWM sector 0, then 1, 2, 3, 4, 5, 0, 1 … in a fixed sequence. According to the chosen PWM scheme, there will be block commutation output in related phases for each PWM sector. The motor will rotate step by step.

   The PWM sector is shown in **Figure 23**. A multi-channel pattern can be further derived according to the actual PWM scheme. For example, if the PWM sector is 1, output should be V -> W. Refer to chapter **2.9 (PWM schemes)** for the multi-channel pattern of each phase.

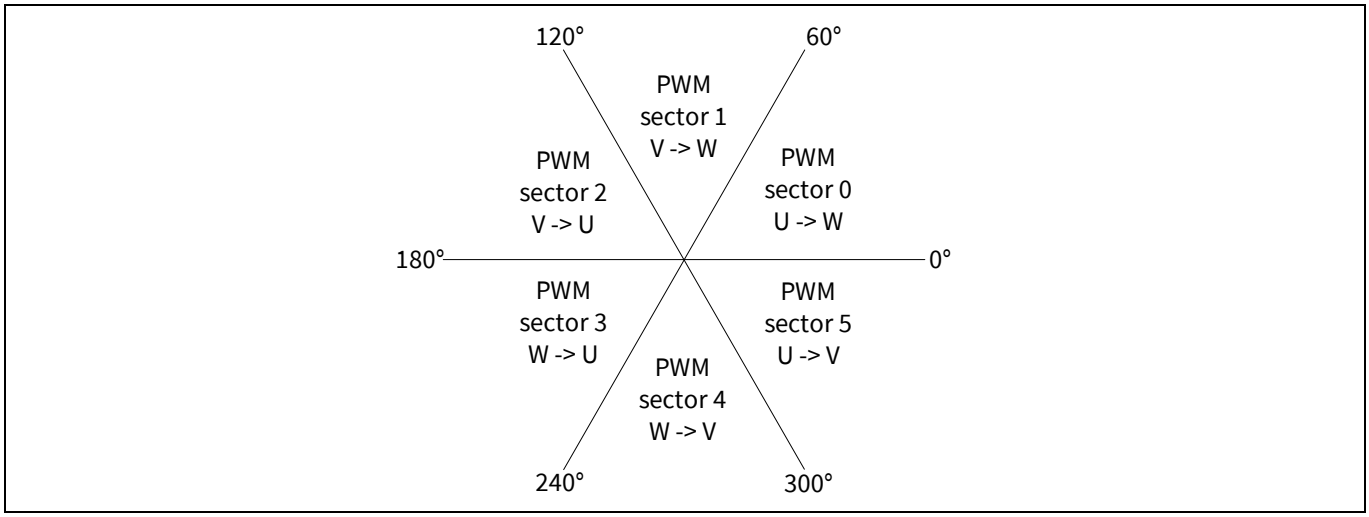


Figure 23 PWM sector illustration

2. The Hall input 1~ Hall input 3-pin transitions will be captured when the motor rotates. Three Hall inputs comprise a Hall pattern, and a total of 14 adjacent Hall patterns will be captured. This will cover at least two electrical periods of 12 Hall patterns for completeness. They are stored in the array of capt_hallpatt[0]~ capt_hallpatt[13].
3. The first Hall pattern is considered as invalid data and will be discarded for accuracy, so valid data starts from the second Hall pattern. This also means the first PWM sector, which is 0, will be discarded.

   Hall sector variables start counting from 0 simultaneously with the second Hall pattern input. This stands for the sequence.
4. The relationship between Hall pattern input and Hall sector during the first electrical period will be extracted and combined, then *MotorParam.HALL_SECTOR_MAP* is derived.
5. *MotorParam.HALL_SECTOR_MAP* is updated and shown in the GUI.

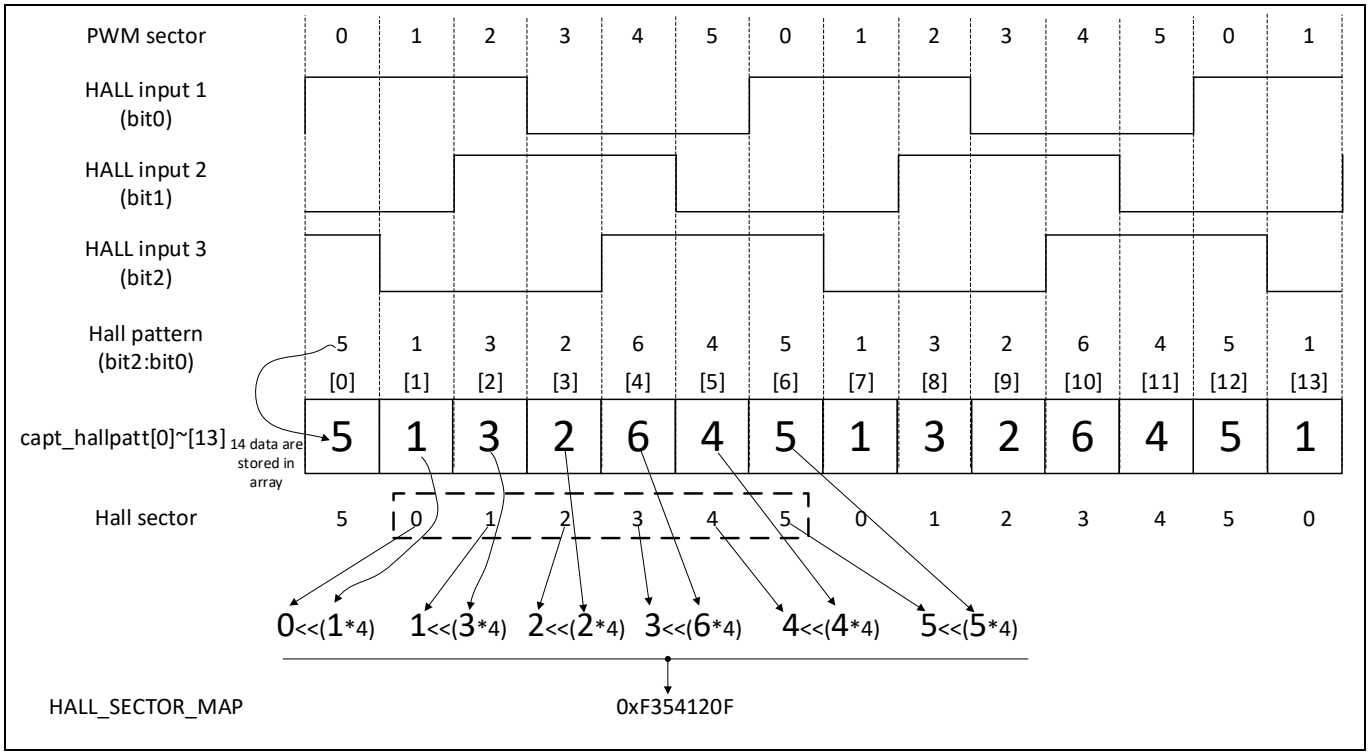


Figure 24 Hall learning illustration

The purpose of Hall learning is to store the relationship between Hall pattern and PWM sector, so that during normal operation, the firmware knows how to output PWM after getting a Hall pattern as input. The relationship between Hall pattern and Hall sector is stored as *MotorParam.HALL_SECTOR_MAP*, which is also the result of Hall learning.

During normal operation after Hall learning is completed successfully, the Hall sector can be derived by searching in *MotorParam.HALL_SECTOR_MAP* using Hall pattern input, then the PWM sector can be further derived. This is because the PWM sector has a fixed relationship with the Hall sector after taking running direction and phase advance into consideration. So normal operation can be considered an opposite process to Hall learning.

2.15 PI controller

The proportional – integral (PI) controller supports integral anti-windup. It stops the integrator if PI output value reaches maximum or minimum value and resumes the integrator once PI output is within the specified range. The PI controller sets the limit flag, if PI output reaches the specified limit.

K_p and K_i values of the PI controller are represented in unsigned value ranges from 0 to 65535. PI controller output is represented as signed value ranges from 0xC000 to 0x4000 (Q14).

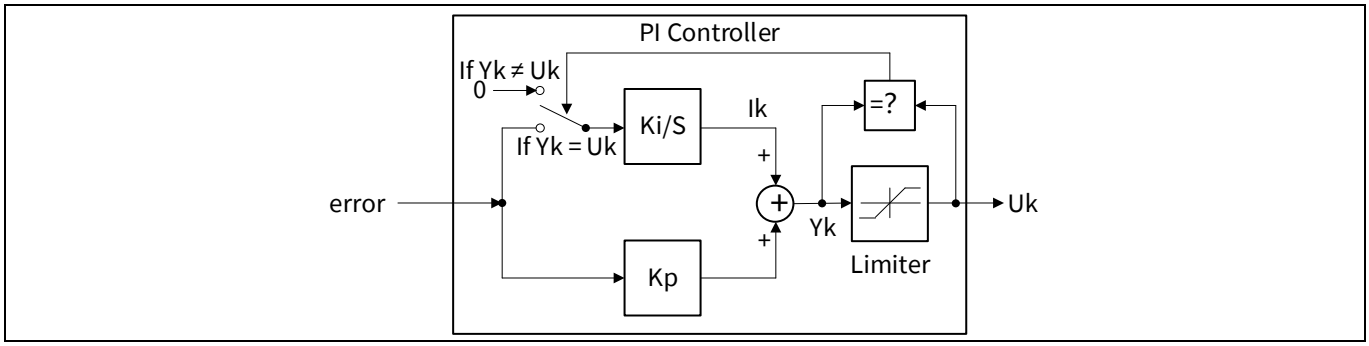


Figure 25 Block diagram of the PI controller

2.16 Protection

The firmware implements the following protections. In the firmware, the fault bit in *MotorVar.error_status* will not cause a motor fault unless the corresponding bit in *MotorParam.EnableFault* is set. In other words, *MotorVar.error_status* is masked by *MotorParam.EnableFault*; if a fault is not enabled by the *MotorParam.EnableFault* bit, it will not put motor control into ERROR state.

2.16.1 Trap protection

Trap protection detects the pin nFAULT signal of MOTIX™ 6EDL7141, which reports any malfunction occurring in it. The specific faults can be enabled or disabled by user configuration in the GUI. Please refer to the **MOTIX™ 6EDL7141 datasheet** for the complete faults and protections table.

2.16.2 Wrong Hall protection

Wrong Hall protection will be detected if the Hall pattern is not as expected or is invalid.

2.16.3 Hall learning protection

Hall learning protection will be detected if it has failed too many times during the Hall learning process.

2.16.4 Motor stall protection

Motor stall protection will be detected if there is no change in the Hall signal for the user-specified time. Configurations:

STALL_MIN_AMPLITUDE_DEFAULT – minimum amplitude in percentage. When amplitude is below this value, motor stall detection is disabled.

STALL_DETECTION_TIME_DEFAULT – detection time for stall protection in seconds; setting to 0 disables stall protection.

2.16.5 DC-link overvoltage/undervoltage protection

DC-link OV/UV protection can be enabled or disabled, and the threshold can be set in the GUI. In the firmware, the following are default configurations:

DC_LINK_OVER_VOLTAGE_LEVEL – DC-link OV threshold in volts

DC_LINK_UNDER_VOLTAGE_LEVEL – DC-link UV threshold in volts

NOMINAL_DC_LINK_VOLT – nominal DC-link voltage in volts

2.16.6 SPI fault protection

SPI fault protection will be detected if there is a fault in the SPI communication with MOTIX™ 6EDL7141.

2.16.7 Gate driver reset protection

This protection will be detected if there is MOTIX™ 6EDL7141 reset during states other than IDLE state.

This protection is realized by writing 0xF into the USER_ID bitfield of OTP_PROG register in MOTIX™ 6EDL7141 by the firmware, and checking this bitfield continuously when the MCU is running in states other than IDLE. If the value in this bitfield is found to have changed, it means there is an unwanted reset of MOTIX™ 6EDL7141 and the gate driver reset protection is detected.

2.17 Controlling the MOTIX™ 6EDL7141 smart gate driver

Here are some characteristics of controlling the MOTIX™ 6EDL7141 smart gate driver.

- EN_DRV pin output level (1 or 0) from XMC1400 is controlled by *EdlIo.en_drv_level*. Please note that the motor state machine controls the EN_DRV pin in case of a certain state change; the EN_DRV pin control is only valid if there is no motor control state change.
- nBrake pin output level (1 or 0) from XMC1400 is controlled by *EdlIo.nbrake_level*. Please note that nBrake pin control is independent of motor control.

Note: *EN_DRV and nBrake control are for test purposes only. Although it is not blocked by firmware or GUI, we strongly suggest not setting either EN_DRV or nBrake to a low level while the motor is running.*

- DC calibration control can be turned on or off, by writing 1 or 0 to MOTIX™ 6EDL7141 to configure the register bitfield CSAMP_CFG[11].
- ADC input selection ($I_{digital}$/D_{VDD}/V_{DDB}) is written to the MOTIX™ 6EDL7141 configure register bitfield ADC_CFG[2:1].
- ADC request is initiated by writing 1 to the MOTIX™ 6EDL7141 configure register bitfield ADC_CFG[0].

2.18 Supporting GUI oscilloscope

To make it possible for the GUI to select the scope signal in “Test Bench” view, there is a register map *RegisterAddrMap[]* defined in register_map.c which organize the data in order according to TraceId. Change *ProbeScope_Chx.TraceId* (x = 0, 1, 2, 3…) to select a different trace signal for each scope channel.
Follow the process as below to add a new variable as trace signal in GUI oscilloscope:

1. Open register_map.c, and go to the definition of array *RegisterAddrMap[]*. Find one of the array elements that gets address from variable “*Reserved*”, then replace “*Reserved*” with a new variable name that is to be added as trace.
2. Add the new variable’s data type accordingly to the array *RegisterTypeMap[]* to provide the variable’s information such as 32bit or 16bit, signed or unsigned. Datatype should be correct and corresponding to the variable in the same index of array *RegisterAddrMap[]*!

Attention: ***Variable’s data type in RegisterTypeMap[] should be correct and correspond to RegisterAddrMap[] one by one. Or else there is possibility for memory fault to happen and damage the motor!***

3 Graphical user interface

The GUI has integrated a default .hex and .elf that are built from a formally released version of the XMC™ firmware described in this document. Hence the following operations can be achieved:

- Change variables' value in XMC™ RAM and apply on the fly
- Flash XMC™ firmware
  - Flash the default .hex or choose another .hex
- Write parameters
  - Flash the motor parameters and MOTIX™ 6EDL7141 parameters into a specified area in the XMC™ Flash
- Burn MOTIX™ 6EDL7141 OTP
- Show trace signal for each scope channel

For more information about the GUI, refer to BPA Motor Control GUI "Help".

References

[1] Infineon Technologies AG: *EVAL_6EDL7141_TRAP_1SH 18 V brushless DC motor drive board;* **Available online**

[2] Infineon Technologies AG: *MOTIX™ 6EDL7141 datasheet*; **Available online**

[3] Infineon Technologies AG: *XMC1400 Reference Manual*; **Available online**

[4] Infineon Technologies AG: *ModusToolbox™ Software*; **Available online**

Revision history

Document version	Date of release	Description of changes
V 1.0	2021-11-08	First release
V 1.1	2022-01-24	Trademarks updated
V 1.2	2022-04-25	Changed DAVE to ModbusToobox
V 1.3	2022-09-13	Added attention to GUI oscilloscope
V 1.4	2023-03-22	Removed fast sync, added phase advance angle description

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

Edition 2023-03-22
Published by
Infineon Technologies AG
81726 Munich, Germany

© 2023 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?
Email: erratum@infineon.com

Document reference
AN_2110_PL88_2111_120446

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

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

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

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

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