Getting started with the six-step reference design firmware for STEVAL-ESC002V1

Introduction

The STSW-ESC002V1 package is the reference firmware for the STEVAL-ESC002V1 hardware based on the STSPIN32F0A advanced BLDC controller with embedded STM32 MCU.

It implements a six-step sensorless algorithm for electronic speed controller (ESC) designs.

1 Acronyms and abbreviations

Table 1. List of acronyms and abbreviations

Acronym	Description
ESC	Electronic speed controller
FCU	Flight control unit
PWM	Pulse width modulation
API	Application programming interface
BEMF	Back electromagnetic force
CMSIS	Cortex® microcontroller software interface standard
IDE	Integrated development environment
PMSM	Permanent magnet synchronous motor
SIP	System-in-package

2 STSW-ESC002V1 firmware package

2.1 Overview

The STSW-ESC002V1 firmware package main features:

- Six-step driving
- Back EMF sensing through comparators
- Designed for high speed operation
- Speed regulation through PWM input
- Optional UART interface
- Based on HAL libraries for STM32

The package includes:

- Drivers for the STEVAL-ESC002V1 evaluation board, the STPIN32F0A SIP and the embedded STM32.
- Middleware for the six-step driving library and the serial communication user interface.
- Project sample implementing the ESC solution.

The firmware sample is written in C programming language and uses either the STM32Cube HAL embedded abstraction-layer software or optimized access to STM32F031 resources. A prerequisite for using this library is basic knowledge of C programming, 3-phase motor drivers and power inverter hardware. In-depth know-how of STM32 functions is required only to customize existing modules and to add new ones for a complete application development.

The IDE tool supported is IAR Workbench (revision 8.22).

2.2 Package content

The STSW-ESC002V1 package is provided as a zip archive file. Once unzipped, under the main stm32_cube folder, four subfolders are available: Binary, Drivers, Middlewares and Projects.

2.2.1 Binary

This folder contains the pre-compiled binary file that can be used to program the board without any code customization.

2.2.2 Drivers

This folder contains the board source code, the STSPIN32F0A, the STM32Cube HAL for the STM32F0 family and CMSIS drivers.

The interface file for the resource mapping between the MCU and the SIP embedded 3-phase controller circuit is in the stspin32f0.h file.

Note: This file should not be changed in normal circumstances as the internal connections are fixed.

The interface file for the STEVAL-ESC002V1 resource mapping is the STEVAL-ESC002V1.h file. It must be updated according to the modification that could be done on the STEVAL-ESC002V1 board or according to a brand new board based on the STSPIN32F0A.

2.2.3 Middlewares

This folder contains the six-step library source code (the motor control algorithm core) and the serial communication user interface.

The six-step library is composed of “6Step_Lib.c” and “6Step_Lib.h”.

The serial communication user interface based on UART is composed of “UART_UI.c” and “UART_UI.h”.

For further details on the six-step library functions and API, refer to Section 7 STSPIN32F0A six-step firmware library overview.

2.2.4 Projects

This folder contains the project specific source code demonstrating the board functionalities and the IDE project files.

The STSW-ESC002V1 firmware package contains a demonstration layer (located in the “Projects\Multi\Examples \MotionControl\STEVAL-ESC002V1” folder) for further code development.

The **Src** sub-folder contains:

- **main_32F0.c**: main file for initialization and infinite loop. It contains MCU peripheral initialization and the six-step library entry point. Including the header file “6Step_Lib.h” and the “MC_SixStep_INIT()” call, the user level is linked to the motor library and all API functions are available.
- **stspin32f0_hal_msp.c**: standard ST Cube HAL file for MCU configuration which contains also the HAL callbacks (i.e. the ADC callback).
  - One key feature implemented in this file is the PWM interface adjusting the voltage on the motor.
- **stspin32f0_it.c**: STM32CubeHAL file for MCU interrupt request and handling function. The file contains the starting point for UART communication and defines all interrupt handlers.

The **Inc** sub-folder contains:

- **main_32F0.h**: includes the header file “6Step_Lib.h”.
- **stm32f0xx_hal_conf.h**: STM32CubeHAL configuration containing in particular the list of modules to be used in the HAL driver.
- **stspin32f0_it.h**: contains the headers of the interrupt handlers.
- **MC_SixStep_param_32F0.h**: includes the motor control parameter file which provides all the parameters to drive a motor with the six-step library.
- **MC_SixStep_param_7PP_3S_propeller.h**: the motor control parameters files. The complete list of all parameters to drive a motor is shown and explained in Section 7 STSPIN32F0A six-step firmware library overview.

3 Architecture

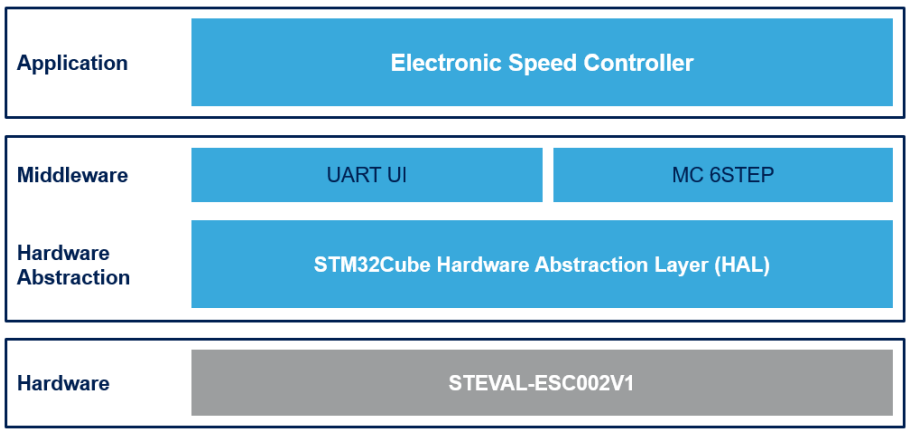
The software layers used by the Motor Control application sample are:

- **Demonstration level 2 layer**: contains the software demonstrating the capability of the evaluation board based on the middleware service layer, the low level abstraction layer (drivers) and the basic peripheral usage applications for board-based functions.
- **Middleware level 1 layer**: contains the user interface and the six-step library which interacts with each other by calling their respective APIs.
- **Driver level 0 layer**: contains the STM32CubeHAL sub-layer and the board support package (BSP) sub-layer.

  The STM32Cube HAL sub-layer provides the low level drivers and the hardware interfacing methods to interact with the upper layers (application, libraries and stacks). It provides generic, multi-instance and function-oriented APIs to help offload user application development time by providing ready to use processes.

  The BSP sub-layer offers a set of APIs related to the hardware components and it is composed of:
  - a component driver which provides specific APIs to the STSPIN32F0A that can be ported to any other board
  - a BSP driver which links the component driver to a specific board and provides a set of easy-to-use APIs

Figure 1. STSW-ESC002V1 architecture



4 STSW-ESC002V1 six-step library implementation

The six-step library source code is available inside the STSW-ESC002V1 package on www.st.com.
The specific implementation for the STEVAL-ESC002V1 supports the following functions:

- Six-step algorithm voltage mode
- Sensorless synchronization using BEMF sensing through comparators
- Complementary driving with dead-time generated though TIM1
- Fast demagnetization
- Open loop or speed loop control

Note: Other features are available, but not supported by the STEVAL-ESC002V1 hardware.

5 System setup guide

5.1 Hardware setup

For the hardware setup, refer to UM2518 freely available at www.st.com.

5.2 Loading the pre-compiled firmware

The easiest way to start working with the board is using the pre-compiled binary available in the dedicated folder. Any SWD programming tool compatible with the STM32F0 microcontroller family can be used for this purpose.

5.3 Building and loading a customized firmware

To customize the firmware you have to use the supported IDE: IAR embedded workbench for the ARM (IAR-EWARM) toolchain (V8.22 or above) which is provided by IAR Systems®.

The package includes ready-to-use project file in the **Projects** folder.

6 Six-step library customization

The library features are enabled and disabled through symbol definition in the project.

Table 2. Pre-defined list of symbols

Symbol	Description	Default
STEVAL_ESC002V1	Sets the STEVAL-ESC002V1 as target hardware	Enabled
USE_HAL_DRIVER	Enables the HAL drivers	Enabled (mandatory)
STM32F031x6	Enables the specific configuration for the STM32F031x6 microcontroller	Enabled (mandatory)
UART_COMM	Enables the UART interface	Disabled(1)
ST_PWM_INTERFACE	Sets the PWM input as speed/voltage setting interface	Enabled
ARR_FILTER	Enables the averaging of the ZC position measurement	Enabled
SENSE_COMPARATORS	Sets the BEMF sensing method through comparators	Enabled (mandatory)
OPEN_LOOP_RAMP	Performs start-up procedure when open-loop is selected	Disabled (1)
VARIABLE_ADVANCE	Makes phase advance variable with speed	Disabled (1)
FAST_DEMAG	Sets the fast-demagnetization strategy	Enabled
PID_V2	Selects the PID control instead of PI for speed loop	Enabled, but ignored as speed loop is not selected
OPEN_LOOP	Uses open-loop approach (no speed loop)	Enabled
MC_7PP_3S_PROPELLER	Selects target motor configuration file	Enabled (mandatory)
DELTA_6STEP_TABLE	Enables differential strategy in six-step sequence generation	Enabled
COMPLEMENTARY_DRIVE	Complementary driving strategy	Enabled (mandatory)
VOLTAGE_MODE	Enables the voltage mode driving instead of the current mode	Enabled (mandatory)
SPEED_SENDING	Enables the monitoring of the speed through UART	Enabled, but ignored because UART is not selected
SPEED_RAMP	Sets the target speed of the motor limiting the acceleration	Enabled, but ignored as speed loop is not selected

1. *Symbols are disabled by adding the “NO_” prefix.*

Each symbol prefixed by “NO_” means the feature is disabled, otherwise it is enabled.

6.1 USE_HAL_DRIVER (mandatory)

The symbol must be defined as it allows the application, middleware and BSP code to be based on the HAL driver API. Otherwise, only a direct access to the embedded STM32F031 peripheral registers is possible.

6.2 STM32F031x6 (mandatory)

The symbol must be defined as it allows to use the data structures, the address mapping, the bit definitions and the macros specific to the STM32F031 embedded in the STSPIN32F0A chip.

6.3 UART_COMM

The symbol enables the usage of the serial port as a user interface. This service is based on the UART_serial_com middleware. When enabled, the ST_PWM_INTERFACE should be disabled by the user in the symbol list to avoid unexpected conflicts.

6.4 ST_PWM_INTERFACE

The symbol enables the PWM input as main driving signal for the motor control algorithm.

According to the driving method selected (see Section 6.11 OPEN_LOOP), the PWM input sets the target speed or the phase voltage of the motor. When the PWM interface is enabled, the UART_COMM should be disabled by the user in the symbol list to avoid unexpected conflicts.

The parameters in the configuration file defines the behavior of this part of the algorithm as described in Section 7.2 PWM interface configuration.

6.5 ARR_FILTER

The symbol enables the speed filtering based on the average computed from the period of the timer in charge of the step commutation. This allows less microcontroller computation and better average speed estimation.

6.6 SENSE_COMPARATORS (mandatory)

The symbol sets the BEMF sensing method through comparators.

Note: This is the only control method supported by the STEVAL-ESC002V1 hardware.

6.7 OPEN_LOOP_RAMP

The symbol enables the ramp-up phase at motor start. By default no ramp-up is applied.

6.8 VARIABLE_ADVANCE

The symbol enables an automatic compensation of the six-step sequence phase advance according to the motor speed.

The parameters in the configuration file defines the behavior of this part of the algorithm as described in Section 7.3 Motor control parameters.

6.9 FAST_DEMAG

The symbol enables the fast demagnetization.

6.10 PID_V2

The symbol enables the usage of a PID regulator for the speed and torque control instead of a PI regulator.

6.11 OPEN_LOOP

The symbol sets the open loop mode for the motor driving.

In this case the speed control is disabled and the algorithm directly applies a target voltage to the motor phase (default operation).

6.12 MC_7PP_3S_PROPELLER

The symbol selects the MC_7PP_3S_PROPELLER.h” configuration file which is the only available one in the firmware package and can be used as a template for custom configuration files.

6.13 DELTA_6STEP_TABLE

The symbol enables six-step high frequency timer PWM channel configuration differential change: only the differences between the previous step and the new step are programmed. This allows less microcontroller computation and higher motor speed.

6.14 COMPLEMENTARY_DRIVE (mandatory)

The symbol sets the complementary output of HF_TIMx PWM.

Note: It is the only driving method supported by the STEVAL-ESC002V1 hardware.

6.15 VOLTAGE_MODE (mandatory)

The symbol sets the voltage mode control.

6.16 SPEED_SENDING (optional)

The symbol enables the firmware speed sending: at every cycle of the speed loop, the filtered speed feedback is sent through the serial port. An external trace tool can then construct a speed versus time graph.

6.17 SPEED_RAMP (optional)

The symbol enables the firmware to control the acceleration or deceleration of the motor with a speed ramp. The speed target is ramped linearly at the mechanical rate ACC defined in the motor control parameter file. The speed target is the set point input of the PID regulator loop.

7 STSPIN32F0A six-step firmware library overview

7.1 Hardware resource mapping

The stspin32f0.h and STEVAL-ESC002V1.h files are the interface between the MCU and the six-step library.

They map the peripherals and the GPIOs according to internal (inside SIP) and external (on the board) connections.

For example, the high frequency timer (PWM) is the TIM1, the stpin32f0.h file maps the TIM1 MCU data structure with a generic name BSP_SIP_HF_TIMx used by the demonstration code in main_32F0.c HF timer init function to attach a generic timer handle HF_TIMx used by the six-step library to the TIM1 MCU data structure *HF_TIMx.Instance = BSP_SIP_HF_TIMx;*.

If the user changes the timer, the update of this file with the right peripheral used is mandatory.

Table 3. Main peripherals mapped in the STEVAL-ESC002V1 firmware example

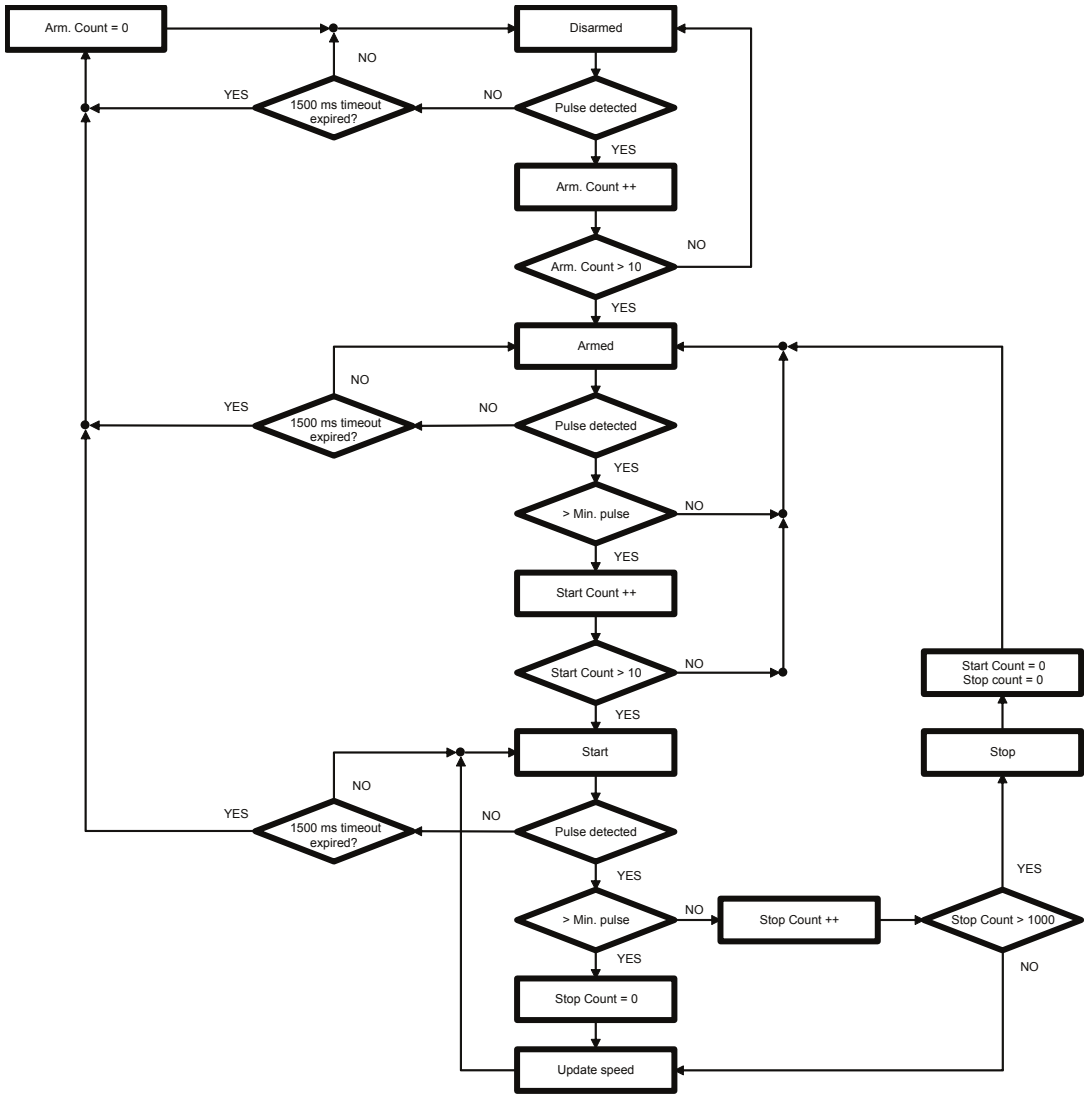
Six-step library generic handle	Drivers peripheral definition	MCU peripheral
HF_TIMx	BSP_SIP_HF_TIMx	TIM1
LF_TIMx	BSP_BOARD_LF_TIMx	TIM3
ZC_TIMx	BSP_BOARD_ZC_TIMx	TIM2
IF_TIMx	BSP_BOARD_IF_TIMx	TIM16
huart	BSP_SIP_UART	USART1

7.2 PWM interface configuration

The PWM interface monitors the PWM input pulse duration setting the voltage applied to the motor or the target speed according to the operation mode selected.

Note: The firmware does not monitor the PWM signal frequency. The algorithm considers only the duration of the positive pulses.

Figure 2. PWM interface flow chart



By default the behavior of the board is:

- At power-up the motor is stopped.
- The ESC waits for at least BSP_BOARD_IF_TIMx_ARMING_VALID_TON pulses before arming (that is allowing the motor driving) the board. Even if the board is armed, the motor is not driven yet.
- When at least BSP_BOARD_IF_TIMx_START_VALID_TON pulses longer than BSP_BOARD_IF_TIMx_MIN_SPEED_TON_US µs are detected, the motor is started.
- When more than BSP_BOARD_IF_TIMx_STOP_VALID_TON pulses shorter than BSP_BOARD_IF_TIMx_MIN_SPEED_TON_US µs are detected, the motor is stopped.
- When the motor is running, the speed is proportional to the PWM pulse duration. The maximum speed is achieved when the pulse duration is BSP_BOARD_IF_TIMx_MAX_SPEED_TON_US µs.
- If no pulses are detected for more than BSP_BOARD_IF_TIMx_STOP_MS ms, the motor is stopped and the board is disarmed, that is motor driving is not allowed.

The parameters for the configuration of the PWM interface are defined in the STEVAL-ESC002V1.h file.

Table 4. PWM interface basic parameters

Constant name	Description	Type and unit
BSP_BOARD_IF_TIMx_STOP_MS	No-PWM timeout in ms	Unsigned integer
BSP_BOARD_IF_TIMx_ARMING_VALID_TON	Number of valid PWM pulses (any duration) arming the board	Unsigned integer
BSP_BOARD_IF_TIMx_START_VALID_TON	Number of PWM pulses above minimum duration starting the motor	Unsigned integer
BSP_BOARD_IF_TIMx_STOP_VALID_TON	Number of PWM pulses below minimum duration starting the motor	Unsigned integer
BSP_BOARD_IF_TIMx_MIN_SPEED_TON_US	Minimum pulse duration (corresponding to minimum speed) in µs	Unsigned integer
BSP_BOARD_IF_TIMx_MAX_SPEED_TON_US(1)	Maximum pulse duration (corresponding to maximum speed) in µs	Unsigned integer
BSP_BOARD_IF_TIMx_MIN2MAX_BITS	Number of bit resolution for the PWM input on time.	Unsigned integer

1. *The value is calculated according the following formula: BSP_BOARD_IF_TIMx_MIN_SPEED_TON_US + $2^{BSP\_BOARD\_IF\_TIMx\_MIN2MAX\_BITS}$. It is not possible to impose an arbitrary value, but the range must be varied using the BSP_BOARD_IF_TIMx_MIN2MAX_BITS parameter.*

7.3 Motor control parameters

In the tables below, the constants in bold can be modified to tune the firmware to a specific motor and control board, while the other constants are computed by the compiler preprocessor using the constants in bold and shall not be modified.

Table 5. Motor control basic parameters

Constant name	Description	Type and unit
NUM_POLE_PAIRS	Number of pole pairs of the BLDC motor.	16bit unsigned
DIRECTION	Sets the motor direction: (0) for CW or (1) for CCW.	0 or 1
TARGET_SPEED_OPEN_LOOP	Target speed in ramp-up phase.	32bit unsigned, RPM
TARGET_SPEED	Target speed in closed loop. Not used when the PWM interface is enabled or in open loop mode.	32bit unsigned, RPM

Table 6. Motor control advanced voltage mode parameters

Constant name	Description	Type and unit
STARTUP_DUTY_CYCLE	Tenths of percentage of high frequency gate driving PWM on time at the start-up. It is also the minimum duty cycle applied to the motor in open loop mode.	16bit unsigned, between 1 and 1000
KP_GAIN	Kp parameter for PI(D) regulator.	16bit unsigned
KI_GAIN	Ki parameter for PI(D) regulator.	16bit unsigned
KD_GAIN	Kd parameter for PID regulator.	16bit unsigned
K_GAIN_SCALING	Kp, Ki, (Kd) scaling for PI(D) regulator.	8bit unsigned, typically 14, max 16
LOWER_OUT_LIMIT	Low Out value of PI regulator.	16bit signed
UPPER_OUT_LIMIT	High Out value of PI regulator.	16bit signed

Table 7. Motor control advanced common parameters: gate driving

Constant name	Description	Type and unit
GATE_DRIVING_PWM_FREQUENCY	PWM frequency for the gate drivers.	32bit unsigned, Hz
SYSCLOCK_FREQUENCY	System clock frequency of the embedded MCU.	32bit unsigned, Hz
HF_TIMX_PSC	High frequency timer prescaler.	32bit unsigned
HF_TIMX_ARR	High frequency timer period.	32bit unsigned
DEAD_TIME_NS	Time during both channels of the high frequency PWM are off.	32bit unsigned ns
DEAD_TIME	Dead time in clock pulses. Please refer to STM32F0x1 reference manual for more details.	32bit unsigned, between 0x00 and 0xFF
PULSE	Duration of the PWM on time at the initialization of the high frequency timer. Defines the maximum duty cycle achievable in Current Mode control.	16bit unsigned

Table 8. Motor control advanced common parameters: step timer

Constant name	Description	Type and unit
LF_TIMX_PSC	Low frequency timer prescaler.	32bit unsigned
LF_TIMX_ARR	Low frequency timer period. Always set to maximum.	32bit unsigned
LF_COUNTER_CYCLE_TIME_NS	Duration of a single timer tick in ns.	32bit unsigned ns
LF_TIMX_ARR_GUARD_TIME_NS	Unused	
LF_TIMX_ARR_GUARD_TIME_CYC	Unused	

Table 9. Motor control advanced common parameters: open loop control

Constant name	Description	Type and unit
ACC	Mechanical acceleration rate used during ramp-up and speed ramp generation.	32bit unsigned, RPM/s
MINIMUM_ACC	Minimum mechanical acceleration rate (for high inertia motor). Taking into account only if OPEN_LOOP_RAMP is set in the symbol list.	32bit unsigned, RPM/s
NUMBER_OF_STEPS	Maximum number of steps for acceleration during the ramp-up phase. If the open loop target speed is not reached when all the ramp-up steps have been done, an error is generated and the motor is stopped.	32bit unsigned
TIME_FOR_ALIGN	Time for alignment before the ramp-up.	16bit unsigned, ms
BUTTON_DELAY	Delay time to enable push button for new command.	32bit unsigned, ms

Table 10. Motor control advanced common parameters: closed loop control

Constant name	Description	Type and unit
ZCD_TO_COMM	Zero Crossing detection to commutation delay in 15/128 degrees.	16bit unsigned, 15/128 degrees
MIN_ZCD_TO_COMM	Minimum allowed value for the Zero Crossing detection to commutation delay.	16bit unsigned, 15/128 degrees
VARIABLE_ADVANCE_MUL	Variable phase advance multiplier. ZC to commutation delay applied to the motor is obtained through the following formula: $30 - \frac{VARIABLE\_ADVANCE\_MUL\ x\ speed}{2^{VARIABLE\_ADVANCE\_SHIFT}}$	16bit unsigned
VARIABLE_ADVANCE_SHIFT	Variable phase advance divider. ZC to commutation delay applied to the motor is obtained through the following formula:	16bit unsigned
ZC_TIMX_FREQUENCY_HZ	ZC timer frequency	16bit unsigned, Hz
ZC_TIMX_PSC	ZC timer prescaler.	32bit unsigned
ZC_COUNTER_CYCLE_TIME_NS	ZC timer tick duration in ns.	16bit unsigned, ns
PWM_EDGE_TO_ZC_READ_EXTRA_DELAY_NS	Unused	
PWM_EDGE_TO_ZC_READ_EXTRA_DELAY_CYC	Unused	
HF_COUNTER_CYCLE_TIME_NS	High frequency timer tick duration in ns.	16bit unsigned, ns
ZC_READ_TO_PWM_EDGE_PRE_GUARD_TIME_NS	Guard time where the ZC detection is disabled for this period before the power stage commutation.	16bit unsigned, ns
ZC_READ_TO_PWM_EDGE_PRE_GUARD_TIME_CYC	Guard time before power stage commutation in high frequency timer ticks.	16bit unsigned
ZC_READ_TO_PWM_EDGE_POST_GUARD_TIME_NS	Guard time where the ZC detection is disabled for this period after the power stage commutation.	16bit unsigned, ns
ZC_READ_TO_PWM_EDGE_POST_GUARD_TIME_CYC	Guard time after power stage commutation in high frequency timer ticks.	16bit unsigned

Table 11. Motor control advanced common parameters: speed

Constant name	Description	Type and unit
SPEED_LOOP_TIME	Speed loop time in ms	16bit unsigned, ms
FILTER_DEEP_SHIFT	Deep of digital filter	16bit unsigned
FILTER_DEEP	Number of bits for digital filter	16bit unsigned

Table 12. **Motor control advanced common parameters: debug**

Constant name	Description	Type and unit
`GPIO_ZERO_CROSS`	Enable (1) the GPIO toggling output for zero crossing detection.	0 or 1
`GPIO_COMM`	Enable (1) the GPIO toggling output for commutation detection.	0 or 1

Revision history

Table 13. **Document revision history**

Date	Version	Changes
19-Dec-2018	1	Initial release.

Contents

List of figures

List of tables