TB3330

Basic Kick E-Scooter BLDC Motor Driver Firmware

Author: *Sergiu Oprea*
Microchip Technology Inc.

INTRODUCTION

This firmware offers the basic functionalities needed to drive a BLDC/PMSM motor, which is part of a kick e-scooter. The firmware is intended to be used with the ***High-Performance E-Scooter BLDC Motor Driver Reference Design*** board. This provides a fast way to evaluate the performance of the motor driver reference design. The following key features are available in this current revision of the firmware:

- ***Field Oriented Control*** for high-efficiency and low operating noise of the motor
- Sensored (Hall sensors) or sensorless operating mode
- ***Constant Torque*** or ***Constant Speed*** operating modes
- ***Constant Power Mode*** to limit the maximum battery current
- Potentiometer or throttle key support to set the torque or the speed of the motor
- Regenerative brake with current profiler
- Protections: overcurrent on phase currents, **VBUS** under/over voltage, MOSFETs drivers bias under/over voltage, DC offset of the current amplifiers and overtemperature of the inverter
- ***X2C Scope*** support for debugging or internal signals view

DETAILED DESCRIPTION

The Motor Control Algorithm

This firmware uses the "*Field Oriented Control* (***FOC***)" algorithm for motor control. This algorithm provides excellent performance but it is computationally intensive. It generally needs microcontrollers of medium to high computational power to be implemented. The dsPIC33CK line of Digital Signal Controllers (DSC) is ideally suited for implementing the ***FOC*** algorithm.

The ***FOC*** requires a precise measurement of the rotor position. This position can be measured by either using encoders, resolvers and other high-resolution angle sensors or it can be estimated using certain mathematical models of the motor (sensorless operation). However, high-resolution angle sensors are expensive and the sensorless estimation of the rotor position is not effective at low speeds. Most of the motors used in low-cost e-mobility applications, like e-scooters and e-bikes, are equipped with low-resolution angle sensors, like the Hall position encoders. These motors are in general, controlled using simple six-steps control algorithms (trapezoidal currents) where the resolution provided by the Hall sensors is appropriate. By combining the information provided by the Hall sensors at low speeds with the information gathered from the ***Angle Observers*** at high speeds, it is still possible to use the low-cost position encoders to implement the ***FOC*** and maintain the high performance of the algorithm across a wide speed range. This firmware uses the Hall sensors to provide the rotor angle at low speeds and the ***Angle Observer*** to correct the rotor angle at high speeds.

To provide a high-resolution angle at low speeds, the angle is interpolated using the technique described in **AN4064**. Consult this application note for more details about the angle interpolation using Hall sensors. This technique is effective at low speeds, but provides poor results at high operating speeds due to the limited resolution of the Hall sensors. That is, only six positions of the rotor are provided and in many cases, the Hall sensors are not accurately aligned with the real rotor position. The ***Angle Observers*** are more effective if the speed of the motor is over a certain threshold. The angle provided is free of glitches and is generally more accurate compared with the angle provided by the Hall sensors interpolators. The firmware uses a non-linear ***Angle Observer*** described in **AN1078**. Please consult this application note for more details.

The driver can spin the motor starting from zero speed in sensored mode. However, to do this, the algorithm applies a six-step trapezoidal commutation if the measured speed is below a certain level where the angle interpolation is not possible. After this minimal speed, the control algorithm will switch to fully interpolation angle mode. The firmware will switch automatically to a six-step commutation if the speed is below the minimum threshold, or the motor is stalled and the angle interpolation is not possible anymore. There is a timer implemented in the firmware that monitors the transitions of the Hall sensors outputs. The value of this timeout timer is adjustable. An additional PLL-like algorithm is used to smooth the angle between Hall sensor transitions. This will further reduce the acoustic noise during the start-up from zero speed. The typical start-up in sensored mode is presented in **Figure 1**.

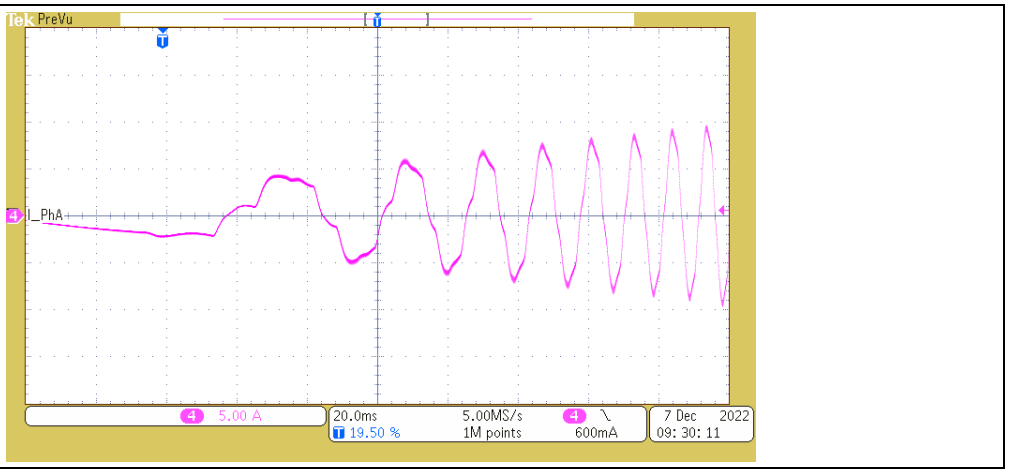

FIGURE 1: *Phase Current During Start-Up in Sensored Mode.*

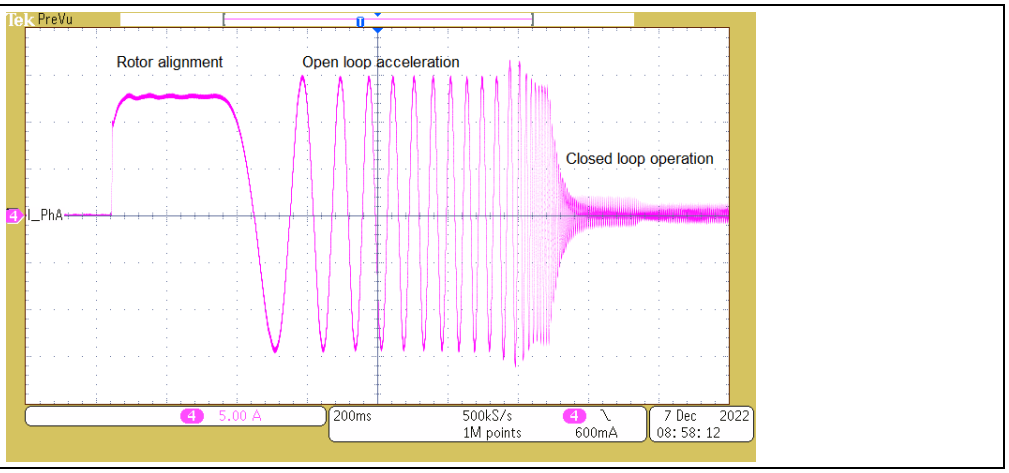

FIGURE 2: *Phase Current During Start-Up in Sensorless Mode.*

The start-up of the motor in sensorless mode is presented in **Figure 2**. There are three phases: the rotor alignment phase where the phase current is forced to a stationary value and the rotor will be aligned to the closest equilibrium position; the open-loop acceleration phase where the motor is accelerated to a minimum speed; and the closed-loop phase where the rotor angle is provided by the ***Angle Observer***. This start-up procedure is presented in detail in **AN1078**.

The Potentiometer/Throttle Key Interface

Most of the kick e-scooters use only a throttle key as the interface between the user and the motor driver. The potentiometer /throttle key provides a linear voltage, and this voltage is used by the driver to adjust the speed or the torque current depending on the operating mode. The firmware provides support for the potentiometer /throttle key, and this is the only user interface. No additional buttons or communication interfaces are implemented in this revision. To use the potentiometer /throttle key, the user must define the low and high voltage levels. The definition used in `User-Parms.h` file to set specific voltages are shown in the following list.

The voltage levels are defined in volts. In the above example, the low-level voltage is set to 0.99V and the high-level voltage is set to 2.5V. A hysteresis of 0.05V is also provided to avoid chatter. These values are typical for a throttle key that uses a linear Hall sensor. The voltage range of the potentiometer /throttle key will expand to the full operating range for the torque current or speed using linear interpolation. The driver can stop the motor by applying the regenerative or the non-regenerative brake if the voltage provided by the potentiometer /throttle key is below:
THROTTLE_LOW – THROTTLE_HYSTERESIS value. The driver will start the motor if the voltage provided by the potentiometer /throttle key is over:
THROTTLE_LOW + THROTTLE_HYSTERESIS value.

```
#define  THROTTLE_LOW(int)((0.99/3.3)*32767)         //  (low level voltage / scale) *32767
#define  THROTTLE_HIGH(int)((2.5/3.3)*32767)         //  (high level voltage / scale) *32767
#define  THROTTLE_HYSTERESIS(int)((0.05/3.3)*32767)  //  (hysteresis voltage / scale) *32767
```

The Protections and Hardware-Dependent Parameters

The firmware provides protections against the following events:

- Motor phases overcurrent
- **VBUS** under/over voltage
- +12V bias voltage “out of specs”
- Power inverter overtemperature (temperature of the power MOSFETs)
- Power inverter high temperature (temperature of the power MOSFETs)
- Large DC offset voltage of the current sense amplifiers
- The Hall sensors read error

The thresholds for these protections are adjustable. The user should use caution when altering these parameters.

Table 1 lists the typical parameters for protection. These parameters are located under “Parameters for Protections” section.

TABLE 1:

Parameter	Units	Description
MIN_VBUS	V	This is the threshold for the undervoltage detection of **VBUS**. The driver will be disabled if **VBUS** is below this threshold.
MAX_VBUS	V	This is the threshold for the overvoltage detection of **VBUS**. The driver will be disabled if **VBUS** is over this threshold.
VBUS_HYST	V	This is the hysteresis for **VBUS** voltage measurement
TYP_VBIAS	V	This is the typical value of the +12V bias voltage. The firmware will measure this voltage and if the value differs from the typical by more than **VBIAS_HYST**, the driver will be disabled.
VBIAS_HYST	V	This is the hysteresis around the typical value of +12V bias.
MAX_CURRENT	A	This is the maximum peak current for the software overcurrent protection. If one of the phase currents is over this value, the driver will be disabled.
MAX_DC_OFFSET	V	This is the maximum absolute value of the current amplifier’s DC offset.
MAX_TEMP_OFF	Units	This is the threshold for the overtemperature protection. The precise adjustment must be done using the **X2C Scope** to visualize the measured temperature voltage across various work conditions. The driver will be disabled if the MOSFETs temperature is greater than this threshold. This is an unsigned quantity; the range is 0 – 65535.
MAX_TEMP_LIMP	Units	If the measured temperature of the MOSFETs is greater than this threshold, the driver enters the **LIMP Mode** where the maximum torque current is reduced. Precise adjustment must be done using the **X2C Scope** to visualize the measured temperature voltage across various work conditions. This is an unsigned quantity: the range is 0 – 65535.
TEMP_HYST	Units	This is the hysteresis for the temperature measurements. Unsigned quantity: the range is 0 – 65535.

The driver will be disabled, and the **FAULT** LED will be turned “ON” if one of the following protections is active:

- Overcurrent protection. This protection is of “latching type” and the flag will be cleared when StopMotor() function is called. No hysteresis or filtering is applied when the values of the phase current are checked. This is a fatal error: the PWM interface will be disabled, and the driver is inactive.
- **VBUS** under-voltage protection. This protection is of “non-latching type” and the flag will be cleared automatically if **VBUS** is over a certain limit. Hysteresis is applied when the value is checked. This is a fatal error: the PWM interface will be disabled, and the driver is inactive if **VBUS** is below the programmed value.
- **VBUS** over-voltage protection. This protection is of “non-latching type” and the flag will be cleared automatically if **VBUS** is lower than a certain limit. Hysteresis is applied when the value is checked. This is a fatal error: the PWM interface will be disabled, and the driver is inactive if **VBUS** is over the programmed value.
- +12V bias voltage “out of specs” protection. This protection is of “non-latching type” and the flag will be cleared automatically if +12V bias is between certain limits. Hysteresis is applied when the value is checked. This is a fatal error: the PWM interface will be disabled, and the driver is inactive.
- Overtemperature protection. This protection is of “non-latching type” and the flag will be cleared automatically if the MOSFETs temperature is below a certain limit. Hysteresis is applied when the value is checked. This is a fatal error: the PWM interface will be disabled, and the driver is inactive.
- DC offset voltage of the current sense amplifiers is large. This protection is of “latching type” and the flag will be cleared when RunMotor() function is called. This is a fatal error: the PWM interface will be disabled, and the driver is inactive.
- The Hall sensors error. This protection is activated when an erroneous code is read by the Hall sensor reading function. This error can be produced by a defective Hall sensor or by a missing connection with the sensors. This is a fatal error: the PWM interface will be disabled, and the driver is inactive if the operating mode is “sensored”.

The MOSFETs high-temperature protection is of “non-latching type” and will activate the **LIMP Mode**: the driver is still active, but the maximum torque current is set to a lower value (compared with the nominal value). The flag of this protection is cleared automatically if the temperature is below the programmed threshold.

The temperature is measured using an **NTC** thermistor placed as close as possible to the power MOSFETs. The type of this thermistor is **B57861S0103F040**. The typical resistance for a given temperature can be determined using the tables provided by the manufacturer. Fine-tuning of the threshold values (**LIMP Mode** and shutdown) can be done using the **X2C Scope**. The **B57861S0103F040** thermistor is an example of overtemperature thresholds computation:

- From the table provided by the manufacturer, find the R_T/R_{25} parameter for the two temperatures that correspond to the threshold for the overtemperature shutdown and **LIMP Mode**. Allow a 5°C difference between the real threshold and the measured one to compensate for the thermal resistance; for example, if the **LIMP Mode** is active over 110°C, set the measured temperature at 105°C. From the table, R_T/R_{25} is 0.05886 at 105°C and 0.04454 at 115°C.
- Calculate the typical resistance at threshold temperatures:
  - R_{105} = 10k * 0.05886 = 0.5886 kΩ
  - R_{115} = 10k * 0.04454 = 0.4454 kΩ
- Compute the thresholds in ADC units using equations:
  MAX_TEMP_LIMP[units] = 65536 * (R_{105}/(R_{105} + 10 kΩ) and
  MAX_TEMP_OFF[units] = 65536 * (R_{115}/(R_{115} + 10 kΩ).
- Plug these values in the `UserParm.h` file under “Parameters for protections” section. Set the **TEMP_HYST** parameter to about 5% of the **MAX_TEMP_OFF** value.

A fast overcurrent protection is implemented using three internal comparators. The threshold for this protection is fixed at approximately 3.1V. For the “***High-Performance E-Scooter BLDC Motor Driver Reference Design***” board the threshold of this protection is approximative 57A.

This is the section “***Parameters for protections***” located inside `UserParms.h` file. The user must check all these parameters before performing any tuning or testing of the BLDC driver to avoid malfunction/damage of the hardware.

```
//      *** Section: Parameters for protections ***********
#define MIN_VBUS       (unsigned int)(32.0 / TOP_VBUS * 65535)  // in Volts, currently 32V
#define MAX_VBUS       (unsigned int)(45.0 / TOP_VBUS * 65535)  // in Volts, currently 45V
#define VBUS_HYST      (unsigned int)(2.0 / TOP_VBUS  * 65535)  // in Volts, currently 2V
#define TYP_VBIAS      (unsigned int)(12.0 / TOP_VBIAS * 65535) // in Volts, currently 12V
#define VBIAS_HYST     (unsigned int)(1.5 / TOP_VBIAS * 65535)  // in Volts, currently 1.5V
#define MAX_CURRENT    (int)(55.0 / IPEAK * 32767)              // In Amp, peak current, currently 55 A
#define MAX_DC_OFFSET  (int)(0.1 / 1.65 * 32767)                // in Volts, currently 100 mV
#define MAX_TEMP_OFF   2800                                     // in units. Aprox 115˚C. This is an
                                                                   unsigned quantity; range is 0 - 65535
#define MAX_TEMP_LIMP  3640                                     // in units. Aprox 105˚C. This is an unsigned
                                                                   quantity; range is 0 - 65535
#define TEMP_HYST      400                                      // in units. This is an unsigned quantity;
                                                                   range is 0 - 65535
//      ****************************************
```

There are also some parameters related to the hardware. These parameters are critical for the proper functioning of the driver. The user must not alter these parameters. These parameters are marked with **DO NOT ALTER**! in `UserParms.h` file.

The Motor Parameters

To reach its maximum performance, the ***FOC*** algorithm requires the motor parameters to be known. **Table 2** lists all motor parameters that must be specified for this driver. Some parameters are specific to the operating mode: sensorless or sensored. These parameters are located under the “MOTOR Definitions” section in `UserParms.h` file.

TABLE 2:

Parameter	Units	Description
POLEPAIRS	units	This is the number of motor pole pairs. It is necessary for measuring the speed of the motor.
PHASERES	Ω	This is the phase DC resistance of the motor. It is necessary for the ***Angle Observer***. The user can measure this parameter with an RLC bridge.
PHASEIND	H	This is the phase inductance of the motor. Necessary for the ***Angle Observer***. The user can measure this parameter with an RLC bridge.
PHASE_ZERO_D	units	This is the phase offset necessary to align the angle provided by the Hall sensors with the angle provided by the ***Angle Observer*** when the motor runs in “**Forward Mode**”.
PHASE_ZERO_R	units	This is the phase offset necessary to align the angle provided by the Hall sensors with the angle provided by the ***Angle Observer*** when the motor runs in “**Reverse Mode**”.
INITIALTORQUE_ALLIGN	A	This is the rotor alignment current. This value sets the peak current. Used only in sensorless mode.
INITIALTORQUE	A	This is the initial torque current. This value sets the peak current. Used only in sensorless mode.
PHASE_ADVANCE_6_STEP	units	This is the phase advance in six-step mode used during the start-up of the motor in sensored mode. There are 182 units for one electrical degree. Adjust this for best acceleration during start-up.
MAX_TORQUE_CURRENT	A	This is the maximum peak value of the torque current. The controller will limit the maximum torque current to this value.

TABLE 2: (CONTINUED)

Parameter	Units	Description
MAX_TORQUE_CURRENT_LIMP	A	This is the maximum peak value of the torque current in **LIMP Mode**. The controller will limit the maximum torque current to this value in **LIMP Mode**.
MIN_TORQUE_CURRENT	A	This is the minimum peak value of the torque current. The controller will limit the minimum torque current to this value.
BRAKE_CURRENT	A	This is the peak value of the current used during the regenerative braking.
MAX_INPUT_CURRENT	A	This is the maximum input current (battery current) during ***Constant Power Mode***.
MAX_SPEED_RPM	RPM	This is the maximum speed value in sensored, speed mode. Not used in sensorless mode.
MIN_SPEED_RPM	RPM	This is the minimum speed value in sensored, speed mode. Not used in sensorless mode.
CORRECTION_SPEED_THR	RPM	This is the threshold for the speed where the high-speed angle correction starts to operate. Used only in sensored mode.
CORRECTION_SPEED_HYST	RPM	This is the hysteresis for the high-speed angle correction. Used only in sensored mode.
ANGLE_OFFSET	units	This is an angle offset that can be added to the rotor angle for the fine-tuning of the maximum torque production. There are 182 units for one electrical degree. Tune this with a maximum of ±5° for best torque.
BRAKE_MIN_SPEED	RPM	If the speed is below this threshold, the regenerative brake is disabled. Used only in sensored mode.
LOCKTIMEINSEC	s	Initial rotor lock time in seconds. Make sure LOCKTIMEINSEC*(1.0/LOOPTIMEINSEC) is less than 65535.Used only in sensorless mode.
OPENLOOPTIMEINSEC	s	Open loop time in seconds. This is the time length from stand still to closed loop. Used only in sensorless mode.
NOMINALSPEEDINRPM	RPM	Make sure NOMINALSPEEDINRPM generates a MAXOMEGA < 1.0. Use this formula: MAXOMEGA = NOMINALSPEEDINRPM*SPEEDLOOPTIME*POLE PAIRS*2/60. If MAXOMEGA > 1.0, reduce NOMINALSPEEDINRPM or execute speed loop faster by reducing SpeedLoopTime. Maximum position of POT/Throttle will set a reference of NOMINALSPEEDINRPM. Used only in sensorless mode.
MINSPEEDINRPM	RPM	Minimum speed in RPM. Closed loop will operate at this speed. Open loop will transition to closed loop at this minimum speed. Minimum POT/Throttle position (CCW) will set a speed reference of MINSPEEDINRPM. Used only in sensorless mode.
MINIMUM_START_SPEED	RPM	The driver will start only if the speed of the motor is over this value. Used only if USE_MINIMUM_SPEED_TO_START option is active. This is a safety feature for kick e-scooter applications.

The Acceleration/Deceleration/Brake

To provide a good riding experience, the acceleration, deceleration and regenerative braking use programmable ramps and profiles. The parameters of the specific ramps are located under “Acceleration/deceleration/brake ramp parameters” section in `UserParms.h` file. **Table 3** lists these parameters.

TABLE 3:

Parameter	Description
ACCEL_RAMP_CONST	This is the acceleration ramp constant. A higher value provides a steeper ramp for acceleration. Limit the maximum value of this parameter to 3% of the full scale of the controlled parameter: torque current or speed. Minimum value is 1.
DECEL_RAMP_CONST	This is the deceleration ramp constant. A higher value provides a steeper ramp for the deceleration. Limit the maximum value of this parameter to 3% of the full scale of the controlled parameter: torque current or speed. Minimum value is 1.
BRAKE_RAMP_CONST	This is the regenerative brake ramp constant. A higher value provides a steeper ramp for the regenerative brake. Limit the maximum value of this parameter to 5% of the full scale of the brake current. Minimum value is 1.

A steeper acceleration provides a “sporty attitude” for the kick e-scooter but may significantly decrease the battery life. The user should be careful when adjusting this parameter. The acceleration/deceleration ramp is also a function of the operating mode, see Constant Speed and Constant Torque modes description.

The regenerative braking provides multiple advantages, improves the battery life, reduces the wear of the disc brake system, and may provide assistance during coasting. However, for a proper operation, the regenerative brake current must be correlated with the motor speed. The firmware uses linear interpolation with the help of tables to adjust the brake current as a function of speed. Two tables must be filled with the specific values for current and speed. The values for these tables are located under “Regenerative brake current profile” section in `UserParms.h` file. These values must be optimized for each application. The following is an example of regenerative brake current profile for a kick e-scooter using a 250W BLDC motor and a maximum load of 100 kg. In this case the speed varies from 80 RPM to 800 RPM and the brake current varies from 7A to 21.5A, peak value.

```
#define   B_CURRENT_7     21.5   // in Amps
#define   B_CURRENT_6     21
#define   B_CURRENT_5     20.5
#define   B_CURRENT_4     19.5
#define   B_CURRENT_3     19
#define   B_CURRENT_2     12
#define   B_CURRENT_1     10
#define   B_CURRENT_0     7

#define   B_CURRENT_7     800    // in RPM
#define   B_CURRENT_6     700
#define   B_CURRENT_5     600
#define   B_CURRENT_4     500
#define   B_CURRENT_3     400
#define   B_CURRENT_2     300
#define   B_CURRENT_1     200
#define   B_CURRENT_0     100
```

The current profile is non-linear but provides a comfortable braking response across the entire range of speeds. **Figure 3** presents the interpolated current versus the speed.

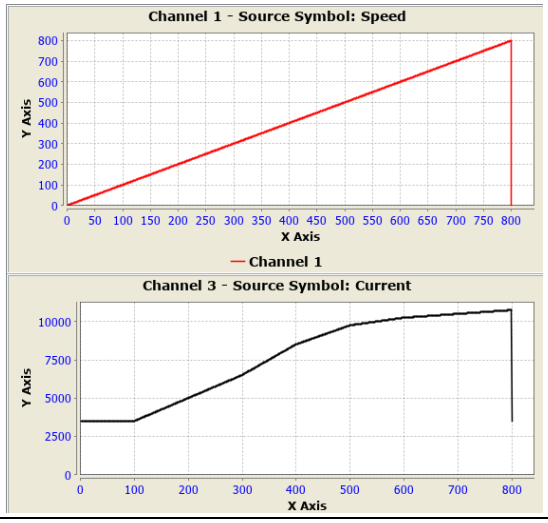

FIGURE 3: *Speed (Red) and the Interpolated Current (Black).*

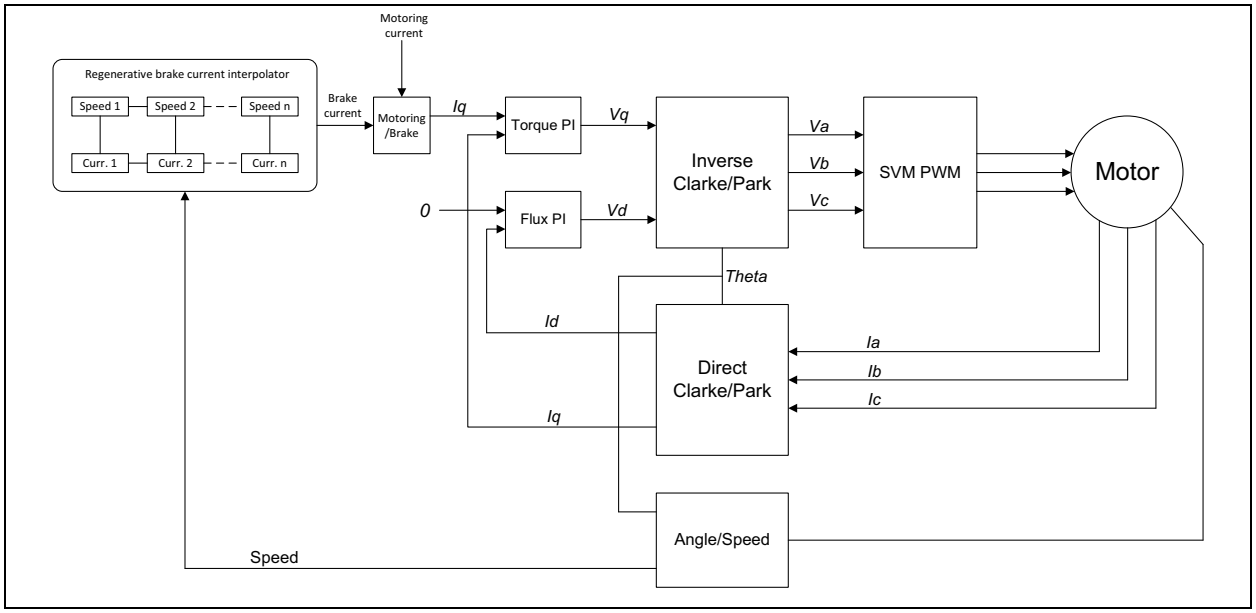

FIGURE 4: *The Simplified Control System Block Diagram.*

The current profiler for the regenerative brake can be disabled by commenting the "#define **REGEN_BRAKE_CURRENT_PROFILER**" option. In this case, the brake current will be ramped up to the maximum value set by "**BRAKE_CURRENT**" parameter. The deceleration ramp is steeper, the braking distance is shorter, however, braking at lower speeds becomes uncomfortable. The simplified block diagram of the control system is presented in **Figure 4**.

The input current during acceleration, constant speed run and regenerative brake phase is presented in **Figure 5**. The voltage is supplied by a 10S/7500 mA Li-Ion battery during this test. The area under the curve during acceleration and regenerative brake phases is a measure of the energy delivered to the motor and recovered from the motor, respectively. These areas must be equal for an efficient regenerative braking system. The motor phase current during the regenerative brake is presented in **Figure 6** and **Figure 7**. The current is sinusoidal ensuring an excellent power factor during the energy recovery process.

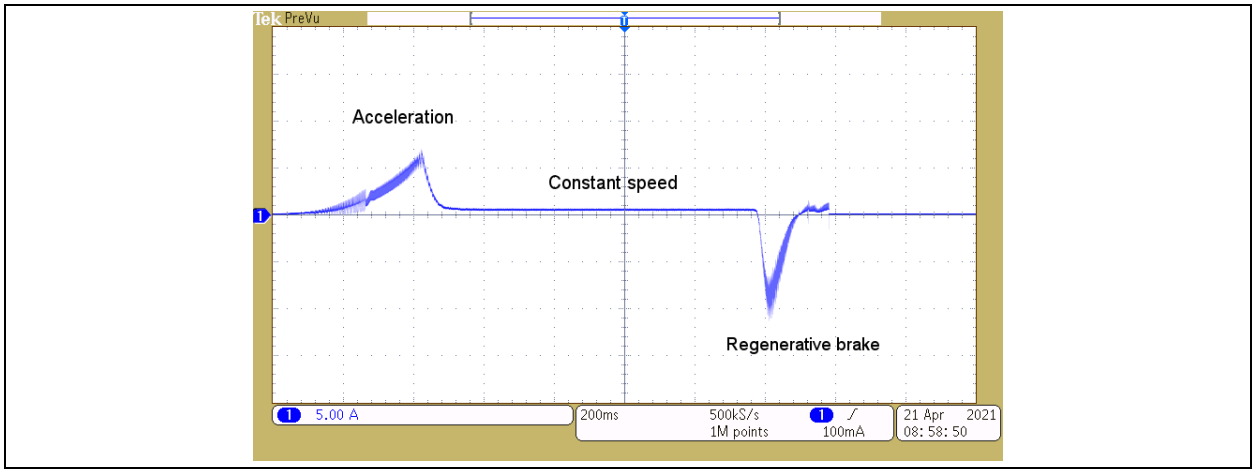

FIGURE 5: *The Input Current during Acceleration, Constant Speed Running and Regenerative Brake.*

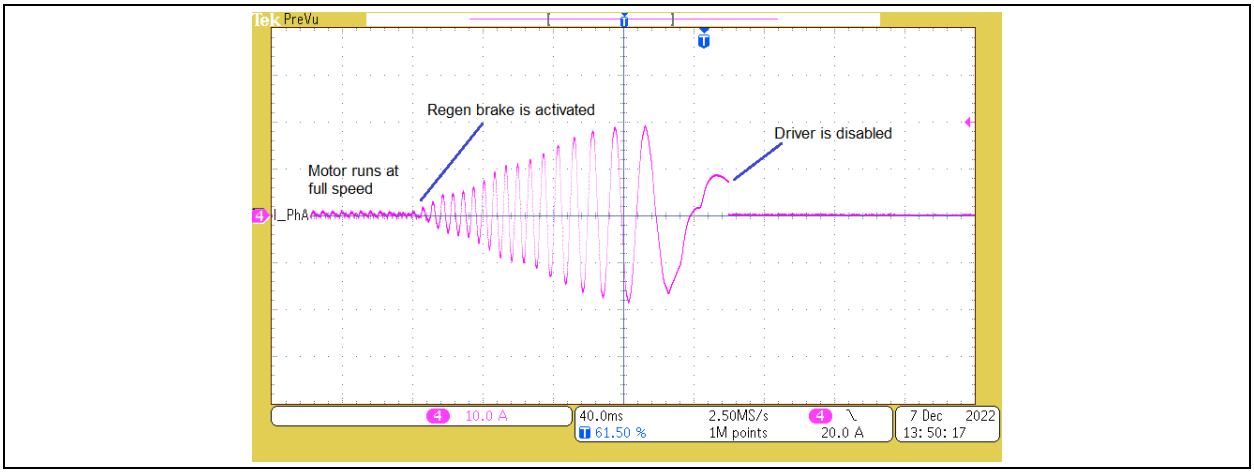

FIGURE 6: *The Phase Current during Regenerative Brake.*

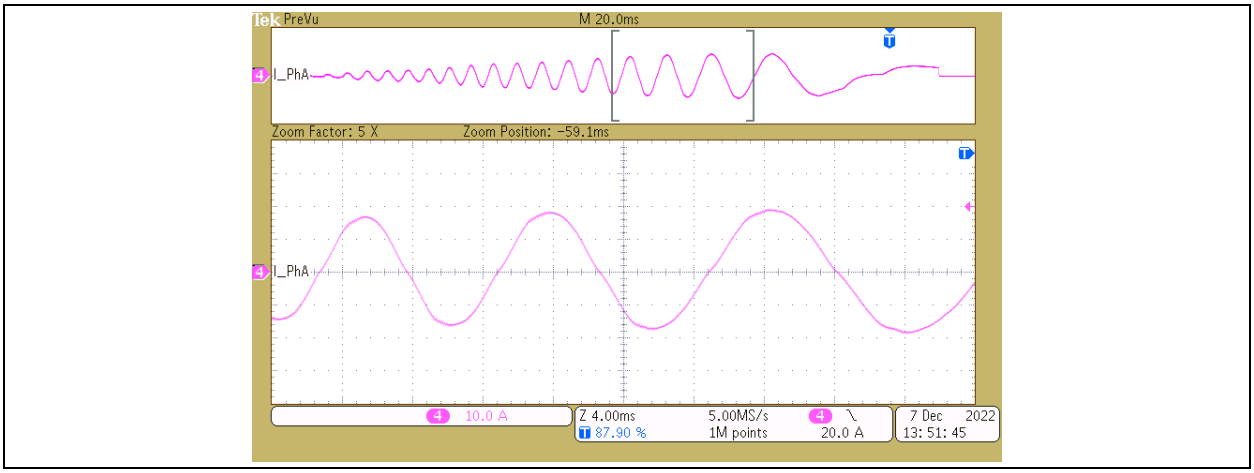

FIGURE 7: *The Phase Current during Regenerative Brake (Detail).*

An important safety measure implemented for kick e-scooter applications prevents the driver from starting when the speed of the motor is below a certain value. This feature prevents the accidental throttle keying when the scooter is in parking or resting position. The firmware supports this option and can be activated using **USE_MINIMUM_SPEED_TO_START** option. The minimum speed value is specified by the **MINIMUM_START_SPEED** parameter. The start condition for the driver is generated when the throttle key is in “neutral” position and the motor speed is over the specified start speed. This feature can be deactivated by commenting the **USE_MINIMUM_-SPEED_TO_START** option; in this case the driver will start at zero speed.

The Constant Torque Mode

In this mode, the value of the torque current (Iq) is regulated to a value programmed by the potentiometer/throttle key. The torque provided by the motor is constant and the speed varies as a function of load. This operating mode provides a “natural” response on e-scooter applications: the response to throttle changes is generally smoother. A linear interpolation/scaling of the torque current provides a good, comfortable riding experience. Tuning in this mode implies only the adjustment of the acceleration and deceleration ramps. The speed varies and must be continuously adjusted using the throttle key if the riding conditions change: different terrain, different inclination, etc. Implementing the “***Auto Pilot***” function (speed is kept constant without operating the throttle key) in this mode is difficult. The “***Constant Torque Mode***” is activated by the ***TORQUE_MODE*** option. The torque current is regulated between **MIN_TORQUE_CURRENT** and **MAX_TORQUE_CURRENT** parameters.

The Constant Speed Mode

In this mode, the speed is regulated to a value programmed by the potentiometer/throttle key. The speed is kept constant regardless the riding conditions. Tuning in this mode is more difficult as the acceleration/deceleration response is a function of the speed reference ramps and the response of the speed PI

regulator. Implementing the "***Auto Pilot***" function in this mode is easy. This operating mode is also required for applications where the top speed of the scooter must be limited (safety regulations). The "***Constant Speed Mode***" is activated by commenting the **TORQUE_MODE** option. The speed is regulated between **MIN_SPEED_RPM** and **MAX_SPEED_RPM** parameters.

The Constant Power Mode

The typical power vs. speed graph for a BLDC motor operating in "***Constant Torque Mode***" is presented in Figure 8. The delivered power is the product of the applied torque and the running speed. The driver must be able to limit the delivered power to avoid excessive current consumption from the battery. The "***Constant Power Mode***" is activated by uncommenting the option **CONSTANT_POWER_MODE**. The driver will limit the input current (battery current) at a value specified by the **MAX_INPUT_CURRENT** constant. The typical power vs. speed graph for a BLDC motor operating in "***Constant Power Mode***" is presented in Figure 9. In this case, the delivered power is constant across a large range of speeds and the maximum input current is limited to a more convenient value.

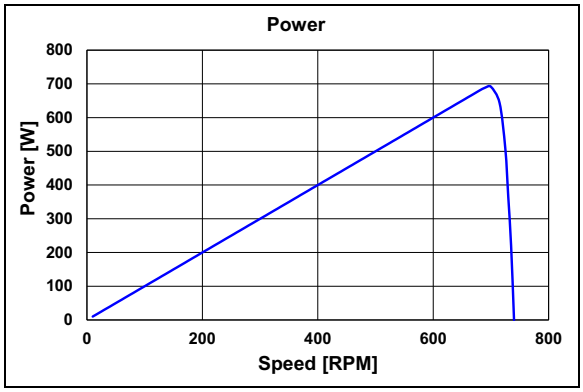

FIGURE 8: *Power vs. Speed in "**Constant Torque Mode**".*

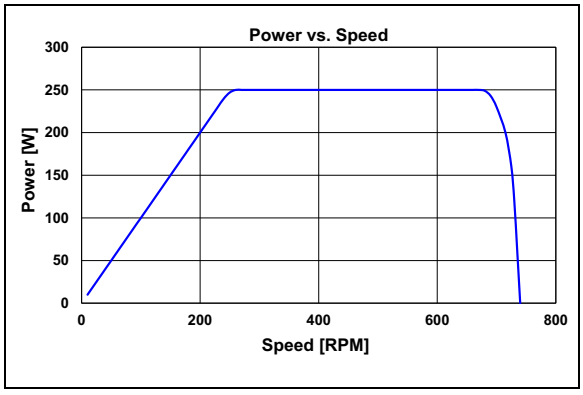

FIGURE 9: *Power vs. Speed in "**Constant Power Mode**".*

The Torque, Flux, and Speed PI Regulators

There are three PI regulators, one for regulating the torque current (Iq), one for regulating the flux current (Id) and one for regulating the speed. The parameters of these regulators are adjustable and are located under "PI Coefficients" section. The proportional, integral, and derivative gains as well as the maximum values for the outputs are available and must be tuned for a specific motor and load. The tuning of these PI regulators is beyond the purpose of this technical brief. Consult **AN1078** tuning guide for more details about tuning of PI regulators.

The PWM Frequency

The default PWM frequency is 16 kHz. This value provides a good compromise between the motor current ripple, input current ripple, the audible noise, and the switching losses on power inverter. This frequency can be modified by adjusting the **PWMFREQUENCY_HZ** parameter; this value represents the PWM frequency in Hz. Care must be used when this frequency is modified as many other parameters depend on this value: integral gain of the PI regulators, potentiometer/throttle key update rate, and acceleration/deceleration/brake ramps. The current measurement for phase currents may be also affected if this frequency is increased. The ADC sampling moment is synchronized with the PWM frequency/duty cycle to minimize the noise and to be able to reconstruct the phase currents. The ***FOC*** state machine also runs at the PWM frequency.

Development Tools

This firmware is developed using the **MPLABX 6.0 IDE** and the **XC16 V1.7** compiler. The optimization level O1 was used during the development. The **PICKIT 4** programming/debugging tool was used to program the board.

The X2C Scope

The **X2C Scope** is a **MPLABX** plug-in that allows runtime debugging or monitoring of the application. See the document "**Installing X2C Scope plugin in MPLAB® X IDE"** for details. The **X2C Scope** is used during the tuning of the firmware for a specific motor/application.

Project Structure

Table 4 lists the files included in this project and a short description of each file.

TABLE 4:

File Name	Description
`Main.c`	This is the main file and contains the main functions of the project.
`init_dspic.c`	This file contains the initialization functions for the processor.
`X2CScope.c`	This file contains the main **X2C Scope** functions.
`X2CScopeComm.c`	This file contains the communication functions for the **X2C Scope**.
`uart1.c`	This file contains the UART 1 initialization functions necessary for the **X2C Scope.**
`FOC.h`	This file includes all the definitions for the functioning of the ***Field Oriented Control*** algorithm.
`UserParms.h`	This file includes all the definitions of the parameters for the functionality of the motor driver.
`init_dspic.h`	This file includes the definitions of the processor initialization functions.
`X2CScope.h`	This file includes the definitions for the **X2C Scope** functions.
`X2CScopeComm.h`	This file includes the definitions for the **X2C Scope** communication functions.
`uart1.h`	This file includes the definitions for the UART 1 functionality.
`Kick_scooter.X.a`	This library file includes all the functions necessary for the motor control algorithm.
`libx2cscope-33ck-elf.a`	This library file includes the **X2C Scope** functions.

Project Resources

Table 5 lists all the microcontroller resources used by this project.

TABLE 5:

Peripheral used	Description
ADC	ADC Core 0 and ADC Shared Core are used for sensing of various signals.
ADC Channel 17 interrupt	This is the main interrupt. The ***FOC*** state machine function runs here.
Operational Amplifiers 1, 2 and 3	The internal operational amplifiers (1, 2 ad 3) are used to implement the current sense amplifiers.
PWM Module	The PWM module is used to generate the three-phase inverter's control signals.
CCP 1	Capture/Compare 1 is used for the motor speed measurement in sensored mode.
UART 1	UART 1 interface is used for **X2C Scope** communications.
CMP1, 2, 3	Interrupts of the DAC/CMP modules 1, 2 and 3 are used for hardware overcurrent protection.
WDT	The Watchdog Timer is used to prevent the accidental firmware stall.
DSP core registers	Used for all computations of the control algorithm.

The code size and speed are presented in Table 6.

TABLE 6:

Program memory	3606 words	**X2C_SCOPE** option is disabled
Data memory	712 bytes	**X2C_SCOPE** option is disabled
Program memory	7392 words	**X2C_SCOPE** option is enabled
Data memory	6198 bytes	**X2C_SCOPE** option is enabled
FOC state machine speed	15 µs	**X2C_SCOPE** option is enabled
FOC state machine speed	14 µs	**X2C_SCOPE** option is disabled

Firmware Options

Various features of the firmware are activated using specific definitions. These specific definitions are presented in **Table 7**.

TABLE 7:

Name	Description
`#define` **SENSORLESS**	This option activates the sensorless operation mode. The rotor angle and the speed of the motor will be provided by the ***Angle Observer***. The Hall sensors are not used in this mode. If this option is disabled, the sensored mode becomes active and the rotor angle and the speed will be provided by the Hall sensors. There are limitations related to the minimum motor speed in this mode. Consult **AN1078** for more details. This operating mode may not be suitable for e-scooter applications where the minimum speed is low.
`#define` **OPENLOOPONLY**	This option will set the functionality of the driver in open-loop only. This option must be used only when the SENSORLESS option is active. Used only for debugging purposes.
`#define` **REVERSE_DIRECTION**	This option is used to reverse the spinning direction of the motor. Works in both modes: sensorless and sensored.
`#define` **TORQUE_MODE**	This option activates the ***Torque Mode*** where the torque component of the stator currents (Iq) is regulated. Works in both modes: sensorless and sensored. If this option is disabled, the driver will operate in the ***Speed Mode*** and the motor speed will be regulated.
`#define` **HIGH_SPEED_ANGLE_CORRECTION**	This option activates the high-speed angle correction and is valid only in sensored mode. If the option is activated the rotor angle will be corrected after a certain speed using the information provided by the ***Angle Observer***.
`#define` **REGEN_BRAKE**	This option activates the regenerative braking. A braking current will be used to reduce the speed of the motor. The resulting energy must be stored in a battery or dissipated using a dissipative device (resistor) otherwise the VBUS voltage can reach dangerous levels. **WARNING: Make sure that this option is activated only when a battery is used as the power source. Otherwise, severe damage to the board may occur.** If this option is disabled, a non-regenerative braking option is activated; this will set the torque current to the minimum defined value.
`#define` **REGEN_BRAKE_CURRENT_PROFILER**	This option activates the regenerative brake current profiler: the brake current will be interpolated as a function of speed. If this option is disabled, the brake current will not be adjusted as a function of speed.
`#define` **USE_MINIMUM_SPEED_TO_START**	If this option is active the driver will not start until the speed of the motor is over a predefined value. This is a safety feature for kick e-scooters to prevent accidental throttle keying at zero speed.
`#define` **CONSTANT_POWER_MODE**	If this option is active the driver will limit the maximum input current (battery current) to a value defined by the **MAX_INPUT_CURRENT** constant.
`#define` **X2C_SCOPE**	This option activates the **X2C Scope**. When this option is disabled, all functions related to the **X2C Scope** are not compiled during the building of the main project; this is to save the program/data memory.

Tuning Procedure for a Specific Motor

To achieve the maximum performance and functionality, initial tuning is necessary for each motor. The following steps are necessary to tune the application:

1. Plug all the motor parameters described in section "*The motor parameters*" and set the options "**SENSORLESS**", "**OPENLOOPONLY**", "**TORQUE_MODE**" and "**X2C_SCOPE**". *Make sure that the option "**REGEN_BRAKE**" is commented*. Consult **AN1292** for details about how the parameters of the motor can be measured. For this step, when using the **X2C Scope**, a serial connection is necessary. See the document "*Installing X2C Scope plugin in MPLAB® X IDE*" and the driver board user guide for details. Set the potentiometer/throttle key to neutral position; the provided voltage must be 0V if a potentiometer is used or the minimum value if a throttle key is used. Power the board from the power supply or battery, insert a current probe in series with one of the motor phases and slowly actuate the potentiometer/ throttle key to the active position. The motor should start spinning when the voltage provided by the potentiometer/ throttle key is over the *"THROTTLE_LOW"* threshold and will accelerate to the final speed which is defined by parameter "**MINSPEEDINRPM**". The motor must be unloaded to avoid the stall condition. Using the potentiometer/ throttle key, verify if the phase current can be controlled between "**MIN_TORQUE_CURRENT**" value and the "**MAX_TORQUE_CURRENT**" value. The parameter "**MIN_TORQUE_CURRENT**" must be set to a minimum value higher than 0A during this test.
   A recommended start value is minimum 10% of the "**MAX_TORQUE_CURRENT**" value. The phase current must be free of high frequency components across the entire range and the motor must run smoothly. The Id and Iq PI controller's parameters can be adjusted during this step. Follow the recommendations from "**AN1292 Tuning Guide**" document. **Figure 10** presents the phase current during this test for an BLDC motor with 15 pole pairs, a phase inductance of 395 µH and a phase resistance of 260 mΩ. This motor has a sinusoidal BEMF waveform. During this step, the functionality of the current regulators is verified, which is mandatory in achieving a good performance of the ***FOC algorithm***. In this case, use the ***X2C Scope*** to plot the Ia, Ib and Ic parameters. The resulting measured currents must be free of both noise and high-frequency components as is presented in **Figure 11**. If the phase currents are not quasi-sinusoidal or are affected by high-frequency components even after the tuning of the Id/Iq PIs parameters, the user must check the output of the current sense amplifiers. **Figure 12**, **Figure 13** and **Figure 14** present the waveforms at the output of the current sense amplifiers. The signals must be well-defined, with sharp transitions and minimum overshoots. The temperature of the board and the motor should be carefully monitored during this test as the losses are high.

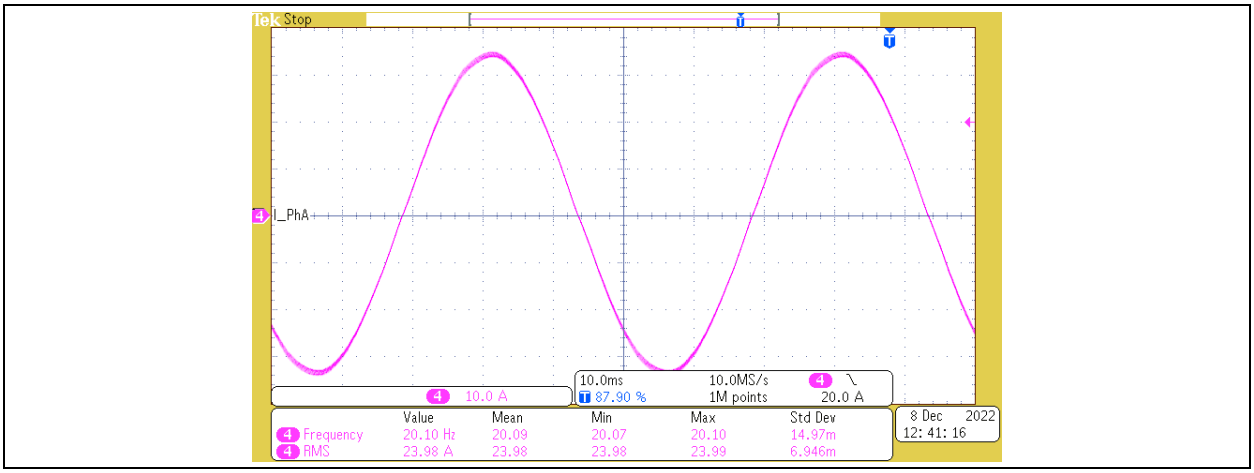

FIGURE 10: *Phase Current in "Open-Loop Mode".*

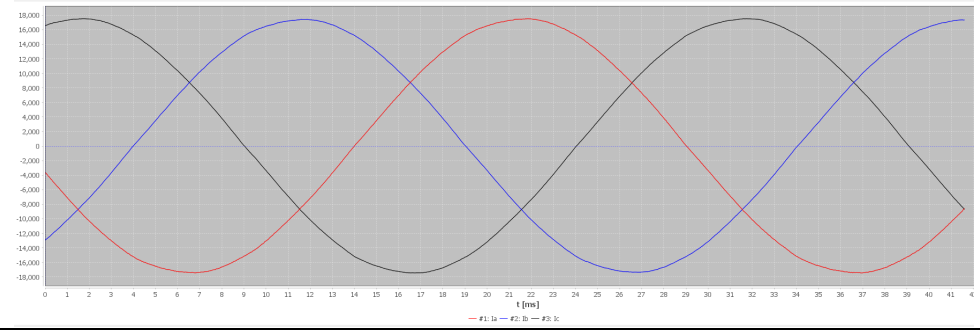

FIGURE 11: *Phase Currents in "Open-Loop Mode" Plotted Using the **X2C Scope**.*

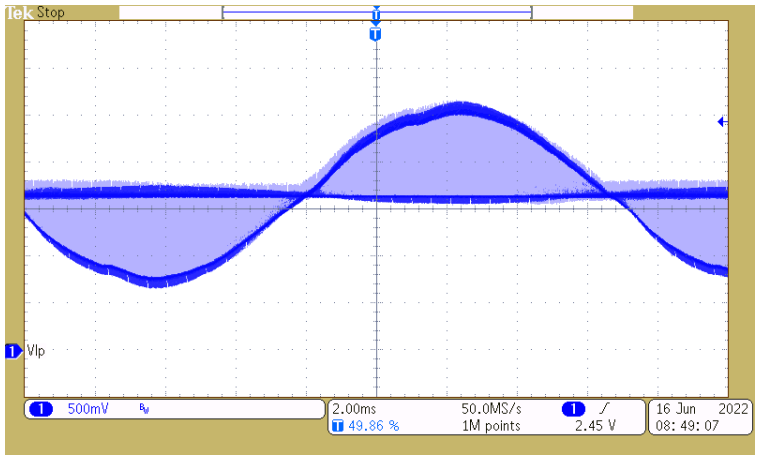

FIGURE 12: *The Output of the Current Sense Amplifier.*

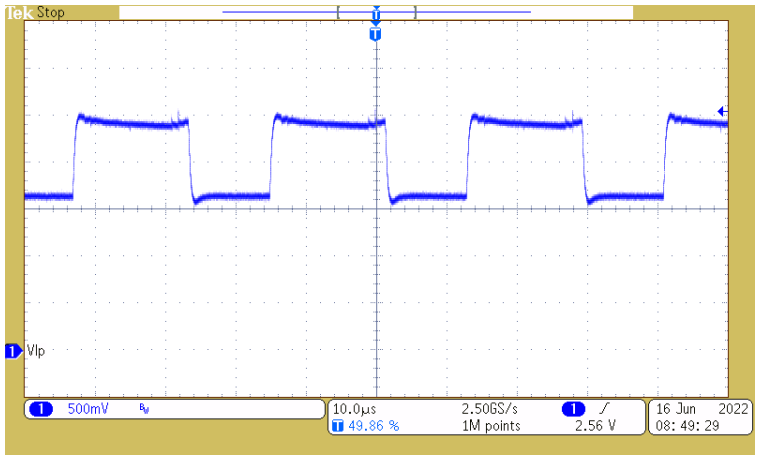

FIGURE 13: *The Output of the Current Sense Amplifier During Positive Phase Current.*

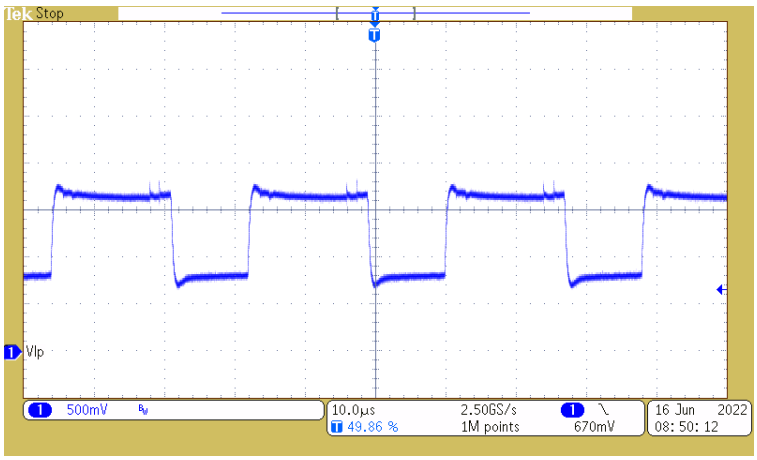

FIGURE 14: *The Output of the Current Sense Amplifier During Negative Phase Current.*

2. For the next step, set the options "**SENSORLESS**", "**TORQUE_MODE**", "**TUNE_PHASE_DIFFERENCE**" and "**X2C_SCOPE**". Set the parameters "**PHASE_ZERO_D**" and "**PHASE_ZERO_R**" to 0. The Motor's Hall sensors must be connected to the driver board; check the motor datasheet for proper Hall sensors connection. Set the potentiometer/throttle key to neutral position and power the board. Slowly actuate the potentiometer/ throttle key to the active position and check if the motor starts to spin and reach the maximum speed; the speed is not controlled in "**TORQUE_MODE**" operating mode and the motor runs at the maximum speed for the programmed phase current and a specific load. Set the potentiometer/throttle key to neutral position and connect the **X2C Scope** to the current firmware project. Make sure that the option "**REVERSE_DIRECTION**" is commented and the motor is spinning in the forward direction. After the **X2C Scope** connection is established, pick from the list of available variables the variables named "*ParkParm.qAngle*" and "*Hall_Angle*" and enable the real-time plot of these variables. These are the angles produced by the ***Angle Observer*** and by the Hall interpolator, respectively. Both angles must evolve in the same direction if the phases and Hall sensors connections are correct. **Figure 15** shows the angle evolution when the phase connections are wrong; in this case, swap two phases and re-check the angles. **Figure 16** shows the angles when the connections are OK. Angles evolve in the same direction but there is a significant phase difference between them. This phase difference must be compensated.

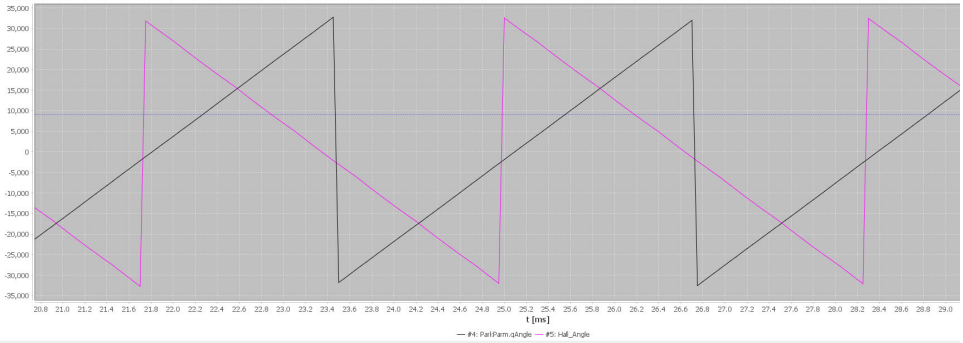

FIGURE 15: *Wrong Phases/Hall Sensors Connections.*

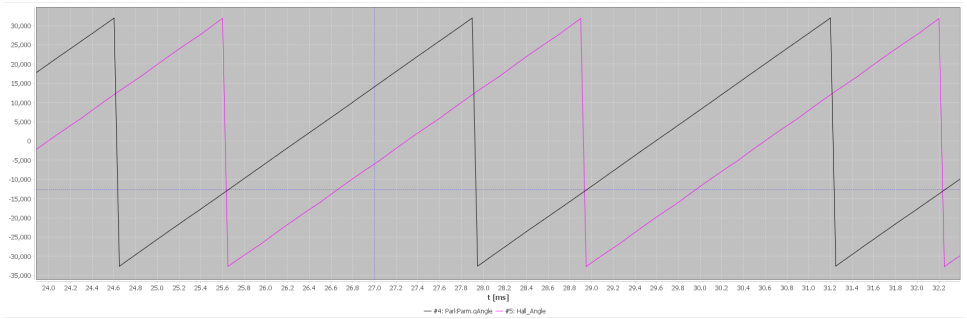

FIGURE 16: *Good Phases/Hall Sensors Connections.*

3. Using **X2C Scope** plot the variable named "*Phase_difference_filtered*". Run the motor and apply a load that sets the phase current to approximately 20% of its maximum phase current. Plug the value of the "*Phase_difference_filtered*" into the "**PHASE_ZERO_D**" parameter. Uncomment the option "**REVERSE_DIRECTION**", run the motor and plug the value of the resulted "*Phase_difference_filtered*" into the "**PHASE_ZERO_R**" parameter. **Figure 17** shows the "*Phase_difference_filtered*" parameter as is plotted by the **X2C Scope**. The ripple is produced by the imperfect alignment of the Hall sensors with the rotor position and must be ignored. In this step, the angle provided by the Hall interpolator is aligned with the angle provided by the ***Angle Observer*** (considered the reference angle). This step is critical and must be performed before running the motor in "sensored" mode. **Please note that in some cases, the phase difference is as high as ±180° and the plotted "*Phase_difference_filtered*" parameter has a very large ripple**. In this case plug 32767 or –32768 for the "**PHASE_ZERO_D**" or "**PHASE_ZERO_R**" parameters and recheck the alignment between the angle of the Hall interpolator and the angle of the **Angle Observer**. **Figure 18** presents the angles when the phase error is compensated. In this case, the phase error between angles is negligible.

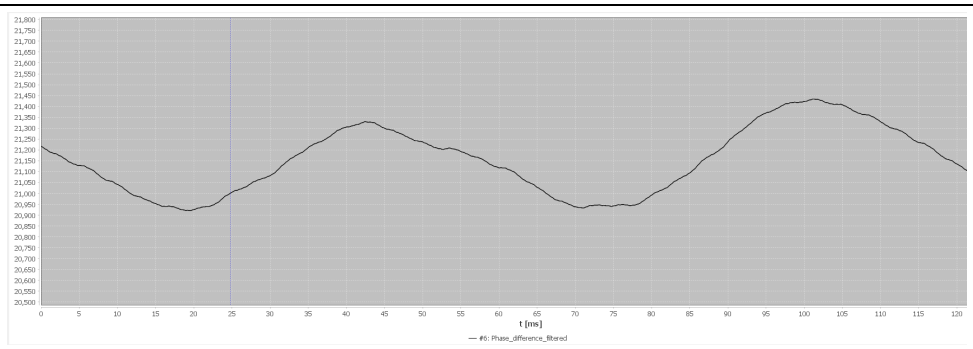

FIGURE 17: *"Phase_Difference_Filtered" Parameter Plotted Using **X2C Scope.***

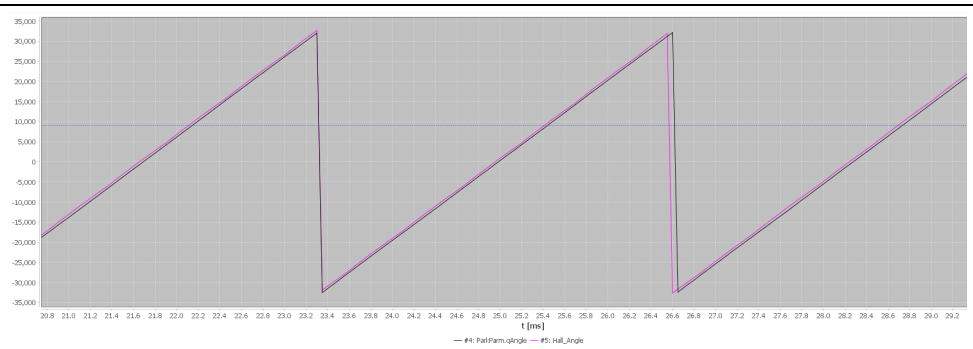

FIGURE 18: *The Angles Provided by the **Angle Observer** and the Hall Interpolator with Compensated Phase Error.*

4. Comment the options “**SENSORLESS**” and “**TORQUE_MODE**” and set the parameter “**MIN_TORQUE_CURRENT**” to 0A. Set the potentiometer/throttle key to neutral position and power the board. Slowly actuate the potentiometer/ throttle key to the active position and check if the motor starts to spin. The speed of the motor must be controlled by the potentiometer/ throttle key between the “**MIN_-SPEED_RPM**” and “**MAX_SPEED_RPM**”. Verify if the speed can be adjusted across the entire range and regardless of the load applied to the motor. Adjust the coefficients of the speed PI controller to ensure consistent operation for all load conditions. In this step, the functionality of the speed regulation loop is verified.
5. Uncomment the “**TORQUE_MODE**” option, slowly actuate the potentiometer/ throttle key to the active position and check if the motor starts to spin. The speed of the motor will reach the maximum value if the motor is unloaded. The operation of the motor must be smooth, without significant audible noise. Load the motor and check the phase current waveforms with an oscilloscope and a current probe. **Figure 19** shows the phase current in “***Torque Mode***” when the option “**HIGH_SPEED_AN-GLE_CORRECTION**” is not active.

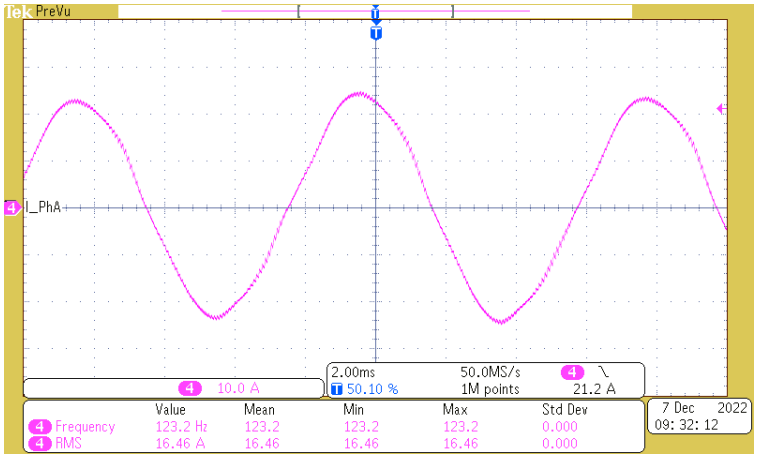

FIGURE 19: *Phase Current in “**Torque Mode**”; “**HIGH_SPEED_ANGLE_CORRECTION**” Option is Deactivated.*

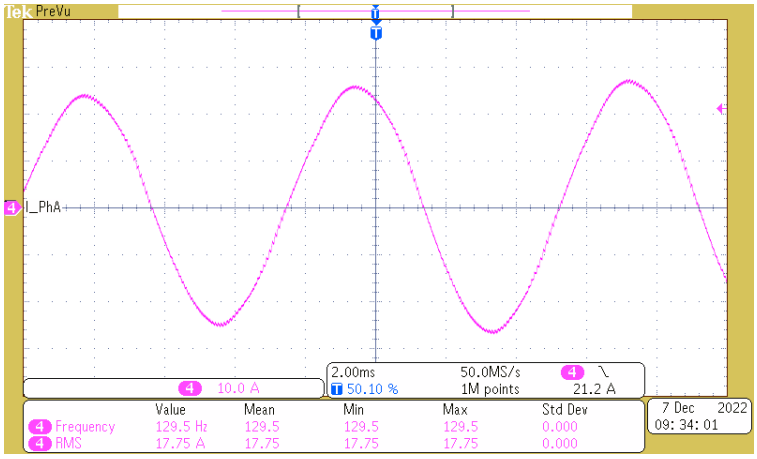

FIGURE 20: *Phase Current in “**Torque Mode**”; “**HIGH_SPEED_ANGLE_CORRECTION**” Option is Activated.*

6. Make sure that the option **CONSTANT_POWER_MODE** is active, run the motor and apply the maximum load. Verify if the maximum input current is equal to the value defined by the **MAX_INPUT_CURRENT** parameter.

> **Note:** It can be difficult and dangerous to apply the maximum load to the motor on the "bench conditions" without a special fixture or stand. In this case, it is recommended to perform this step on the real kick e-scooter while riding by inserting a current probe on one of the input terminals and measuring the absorbed current.

7. Uncomment the option "**REGEN_BRAKE**", connect the BLDC driver to a battery and run the motor. Verify the functionality of the regenerative brake. The motor must suddenly stop spinning when the throttle key returns to the neutral position.

> **Warning:** Severe damage to the driver may occur if is not powered from a battery during this test. Make sure that a battery is used as the power source during this test.

CONCLUSIONS

Together with the ***High-Performance E-Scooter BLDC/PMSM Motor Driver Reference Design*** board, this firmware provides a fast "out of the box" solution for modern kick e-scooter applications. The firmware provides all necessary features for a high-performance BLDC driver: ***FOC*** for high efficiency and low operating noise, constant speed or constant torque operating modes, constant power mode, high performance regenerative braking, and various protections and safety options.

REFERENCES

[1] AN4064 "Sensored (Hall Effect Sensor-Based) Field Oriented Control of Three-Phase BLDC Motor Using dsPIC33CK"

[2] AN1078 "Sensorless Field Oriented Control of a PMSM"

[3] AN1078 "Tuning guide"

[4] AN1292 "Sensorless Field Oriented Control (FOC) for a Permanent Magnet Synchronous Motor (PMSM) Using a PLL Estimator and Field Weakening (FW)"

[5] AN1292 "Tuning Guide"

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable" Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

Trademarks

ISBN: 978-1-6683-1900-0

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4485-5910
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820