Speed Control of Brushless DC Motors-Block Commutation With Hall Sensors

User’s Guide

Table of Contents

Introduction

Microsemi offers a simple, low cost way to try the SmartFusion products for the development of motor control application. SmartFusion® customizable system-on-chip (cSoC) FPGA devices contain a hard embedded microcontroller subsystem (MSS), programmable analog circuitry (ACE), and FPGA fabric consisting of logic tiles, static random access memory (SRAM), and phase-locked loops (PLLs). The MSS consists of a 100 MHz ARM® Cortex™-M3 processor, communications matrix, system registers, Ethernet MAC, DMA engine, real-time counter (RTC), embedded nonvolatile memory (eNVM), embedded SRAM (eSRAM), and fabric interface controller (FIC).

The SmartFusion cSoC devices have major advantages in terms of Fabric, MSS and ACE in the development of Motor drives and control, Power supply regulators, solar inverters etc. With a fabric-based motor controller, the designers have the advantage of flexibility in terms of design and having reliable and deterministic performance.

The SmartFusion Evaluation and Development Kit Boards are developed in a generic way that can be used with the custom inverter board for the development of majority of the motor control applications. This manual explains in detail the design of “Closed loop speed control of Brushless DC motor with Block commutation using Hall Sensors” that is developed based on the following hardware platform:

- The SmartFusion Development Kit Board **(A2F-DEV-KIT)** or the SmartFusion Evaluation Kit Board **(A2F-EVAL-KIT)** with an A2F200 device. Any new version of the board/with A2F500 the project has to be recompiled again.
- The Trinamic TMCM-AC-840-Motor Control Daughter Board Kit **(TMCM-AC-840)**

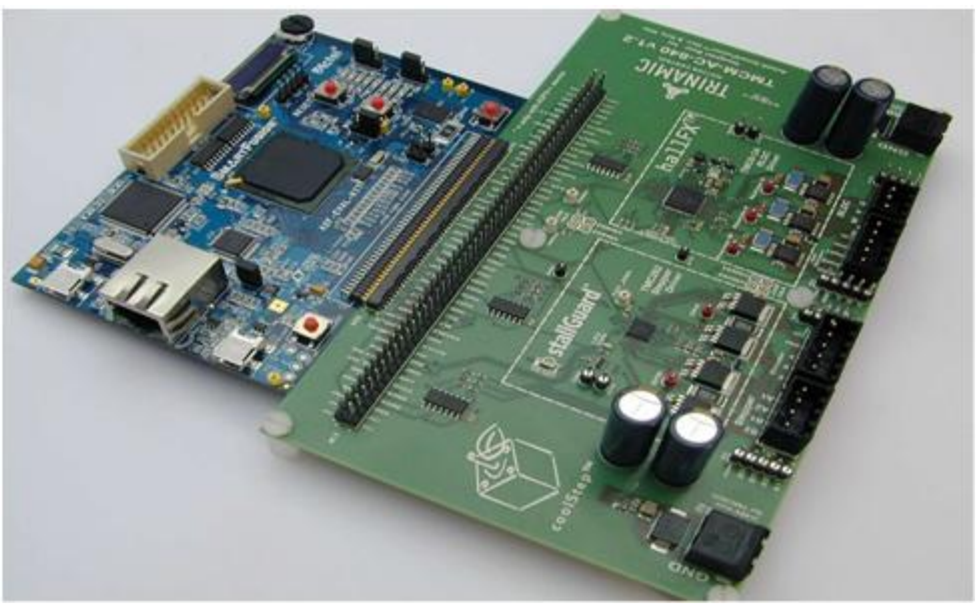

Figure 1 · Trinamic TMCM-AC-840-Motor Control Daughter Board Kit (TMCM-AC-840)

Reference documents:

1. SmartFusion cSoC User Guides & Manuals (www.microsemi.com/soc/products/smartfusion/docs.aspx)
2. SmartFusion Development Kit Board User’s Guide (www.microsemi.com/soc/documents/A2F500_DEV_KIT_UG.pdf)
3. SmartFusion Evaluation Kit Board User’s Guide (www.microsemi.com/soc/documents/A2F_EVAL_KIT_UG.pdf)

4. Trinamic Kit User Manual (www.trinamic.com/tmctechlibcd/integrated_circuits/TMCM-AC-840/TMCM-AC-840_manual.pdf)
5. Trinamic 603A chip User Manual (www.trinamic.com/tmctechlibcd/integrated_circuits/TMC603/tmc603_datasheet.pdf)
6. BLDC motor datasheet (www.trinamic.com/tmc/media/Downloads/QMot_motors/QBL4208/QBL4208_manual.pdf)

Brushless DC Motor Control Theory

Brushless DC Motor or the BLDC Motor is a rotating electric motor consisting of stator armature windings and rotor permanent magnets whereas in a conventional brushed DC motor the stator is made up of permanent magnets and rotor consists of armature windings. The conventional DC motor commutes itself with the use of a mechanical commutator whereas brushless DC motor needs electronic commutation for the direction control of current through the windings. Typically BLDC motors have three phase windings that are wound in star or delta fashion and need a three phase inverter bridge for the electronic commutation.

In BLDC motors the phase windings are distributed in trapezoidal fashion in order to generate the trapezoidal BEMF waveform. The commutation technique generally used is trapezoidal or called block commutation where only two phases will be conducting at any given point of time. An alternative way of commutating the motor is called sinusoidal commutation in which all the three phases will be conducting at any given point of time. PMSM motors also interchangeably called as BLDC motors which have the windings distributed in sinusoidal fashion suited for this sinusoidal type of commutation. The torque generated by PMSM motors is smooth as compared to BLDC motors in which torque will have more ripples. But the peak torque developed by PMSM motors is less as compared to BLDC motors.

The trapezoidal commutation method is the simplest way to control BLDC motors and easy to implement the control aspects of it. For proper commutation and for motor rotation, the rotor position information is very crucial. Only with the help of rotor position information, the electronic switches in the inverter bridge will be switched ON and OFF to ensure proper direction of current flow in respective coils. Hall effect sensors [Hall2, Hall1, Hall0] are used in general as position sensors for trapezoidal commutation. Each hall sensor is typically placed 120 degrees apart and produces '1' whenever it faces the North Pole of the rotor. The hall sensor patterns for a single pole pair BLDC motor during its 360 degree of rotation is shown in the following figures below. The hall sensor shown in red indicates that it outputs '1'.

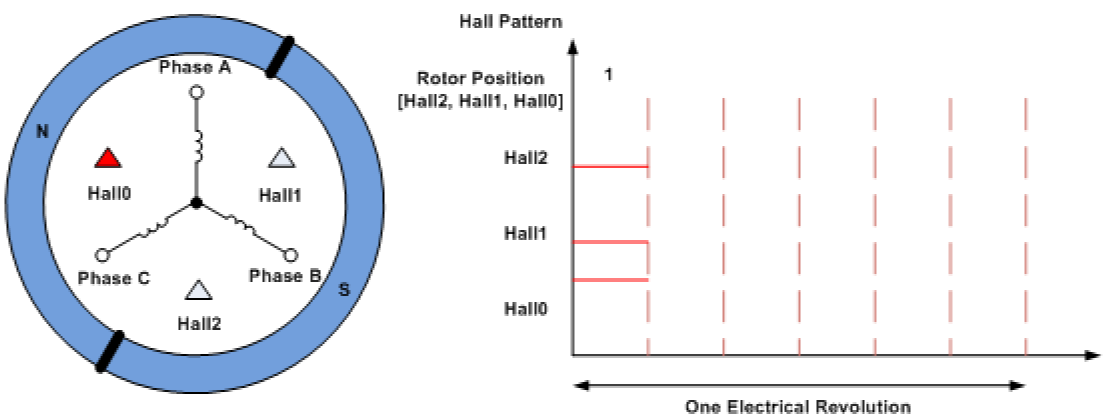

Figure 2 · Step 1: Hall0 Sensor Outputs '1'; Resultant Pattern is 001 [Hall2:Hall1:Hall0]

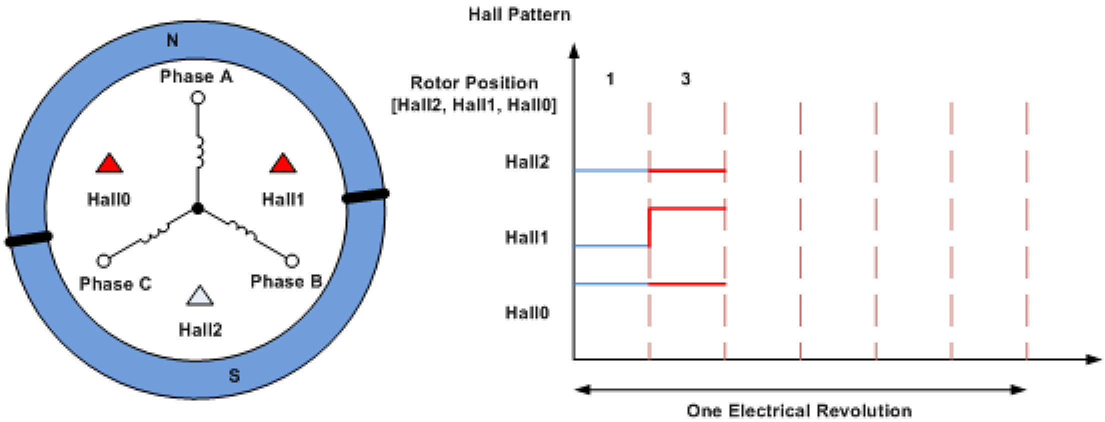

Figure 3 · Step 2: Hall0 and Hall1 Sensors Output ‘1’; Resultant Pattern is 011 [Hall2:Hall1:Hall0]

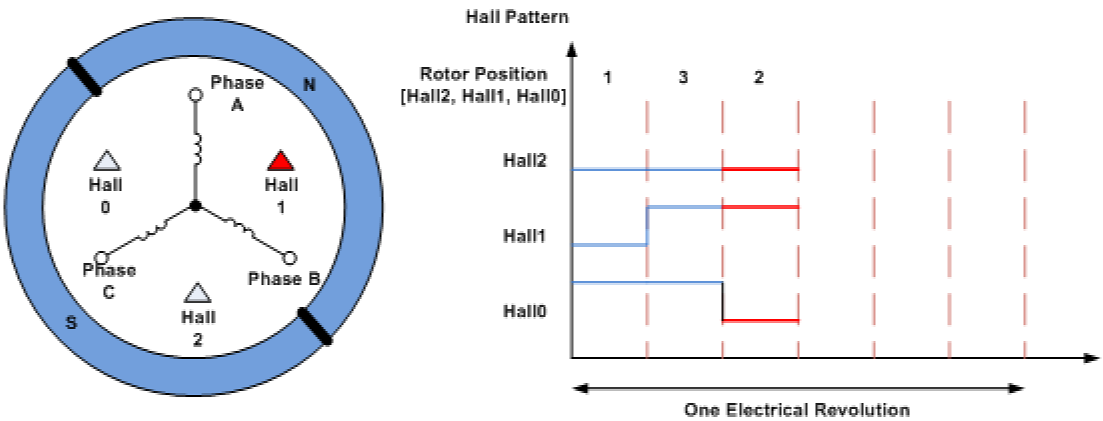

Figure 4 · Step 3: Hall1 Sensor Outputs ‘1’; Resultant Pattern is 010 [Hall2:Hall1:Hall0]

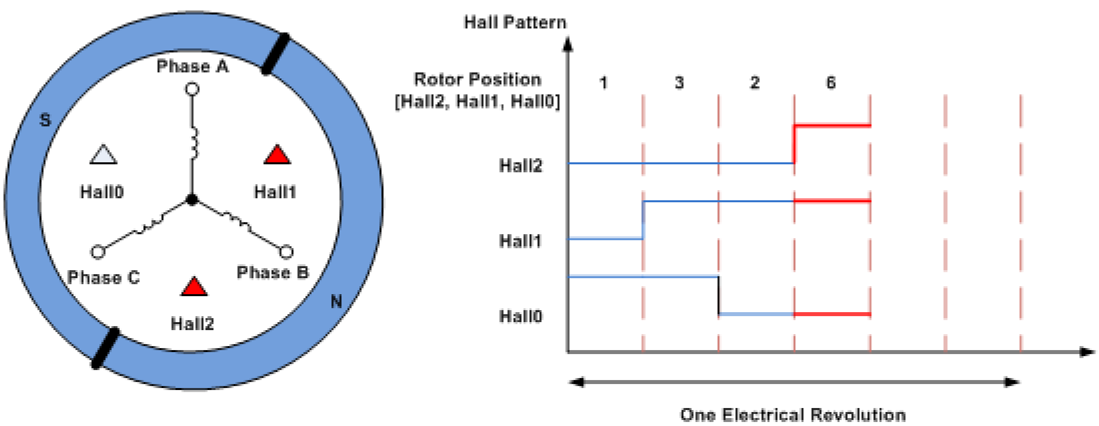

Figure 5 · Step 4: Hall1 and Hall2 Sensors Output ‘1’; Resultant Pattern is 110 [Hall2:Hall1:Hall0]

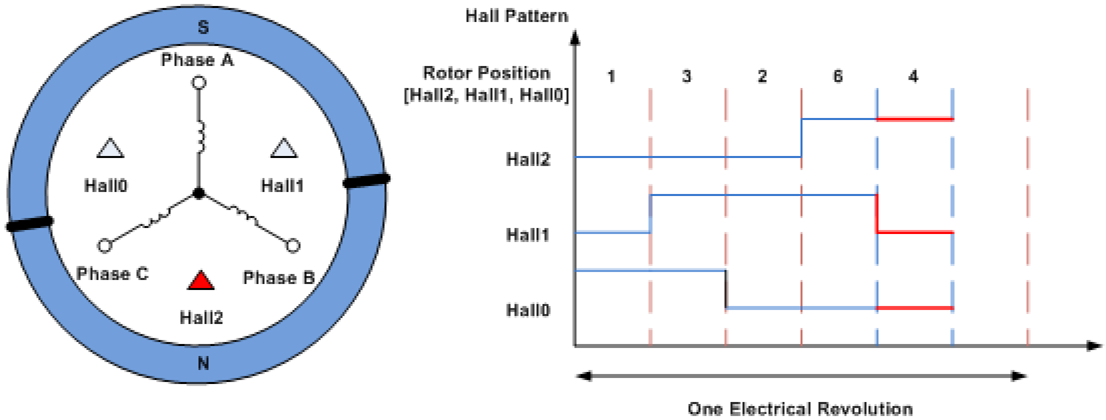

Figure 6 · Step 5: Hall2 Sensor Outputs ‘1’; Resultant Pattern is 100 [Hall2:Hall1:Hall0]

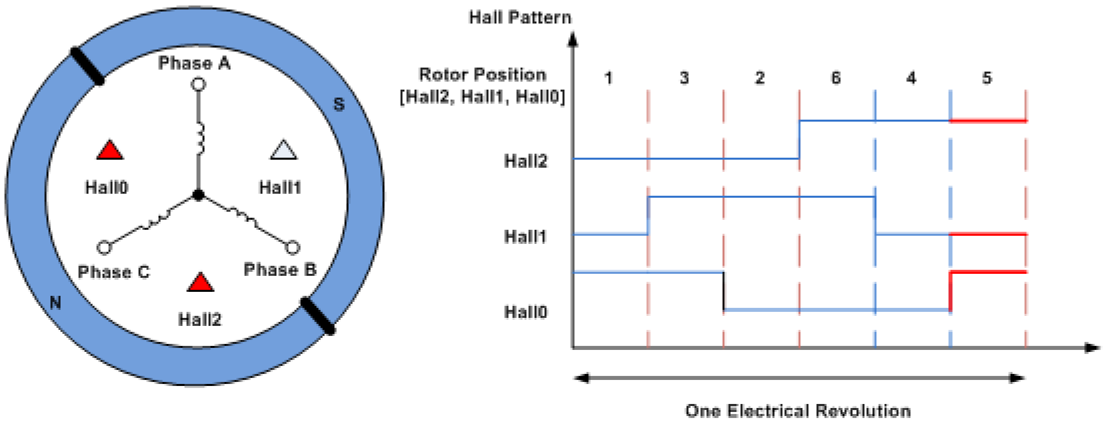

Figure 7 · Step 6: Hall2 and Hall0 Sensors Output ‘1’; Resultant Pattern is 101 [Hall2:Hall1:Hall0]

More on PolePairs

For motors with any number of pole pairs, for each electrical revolution we get six hall patterns as shown in Figure 8:

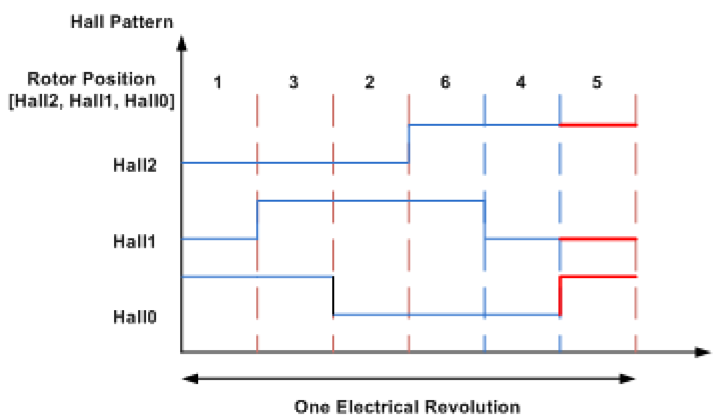

Figure 8 · 6 Hall Pattern for Each Electrical Revolution

But the number of mechanical revolution depends on the number of pole pairs of rotor. For a motor with a single pole pair, the number of mechanical and electrical revolution are equal and hence the electrical angle and mechanical angle. But for a motor with more than one pole pairs, the number of mechanical and electrical revolutions is not equal.

The relationship can be described with the following equation:

*Number of electrical revolutions = Polepairs * Number of mechanical revolution*

Equation 1

Therefore, Electrical angle = Polepairs * Mechanical angle

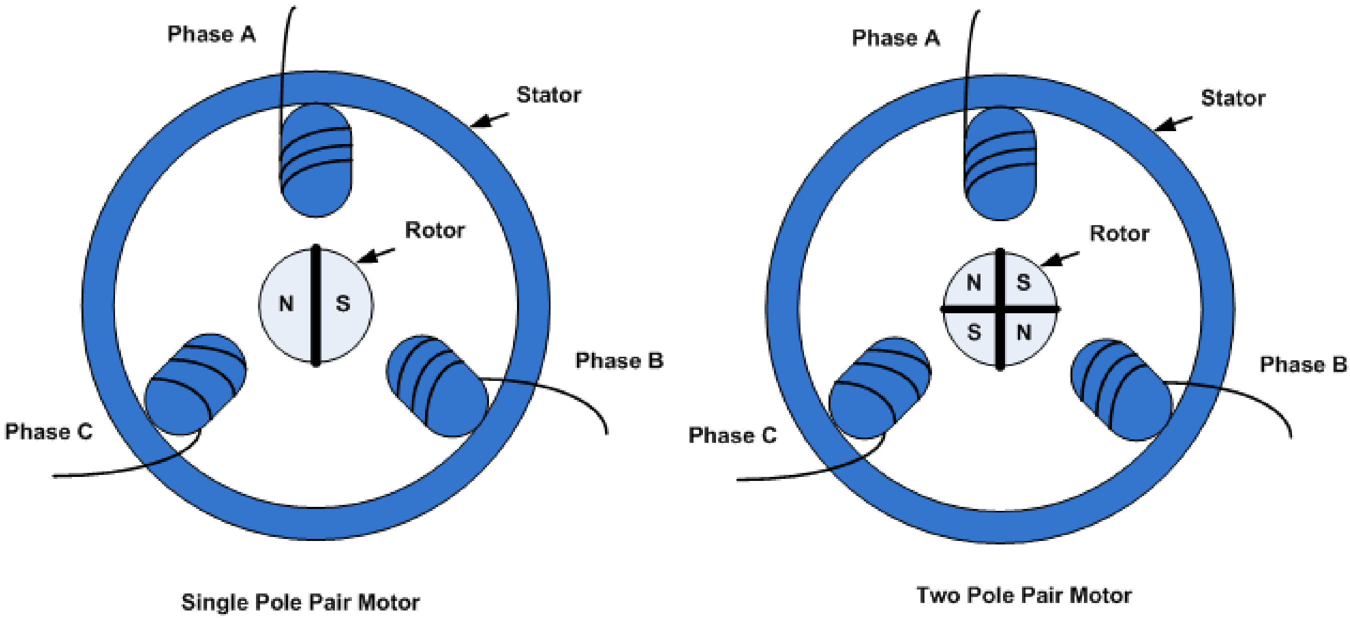

Figure 9 · Pole Pair Motors

As the number of pole pairs increase with the motor, more electrical revolutions occur, Hall pattern changes will be faster, and commutation changes will also be faster. For higher speed operations this necessitates for higher PWM Frequency to achieve good precision control.

Commutation Logic with Hall Sensor Inputs

Note that on every Hall event, only one bit changes. The motor datasheet from the manufacturer specifies the Hall pattern for the CW and CCW direction of rotation and the corresponding coils that need to be energized.

Energizing the appropriate phase coils based on the Hall sensor inputs is known as commutation logic. Whenever a new Hall signal change is detected, the new drive switching pattern is applied. Figure 10 shows the motor current direction through the windings when they are energized, based on the Hall pattern, and is represented as numbers from 1, 2, 3, 4, 5, and 6.

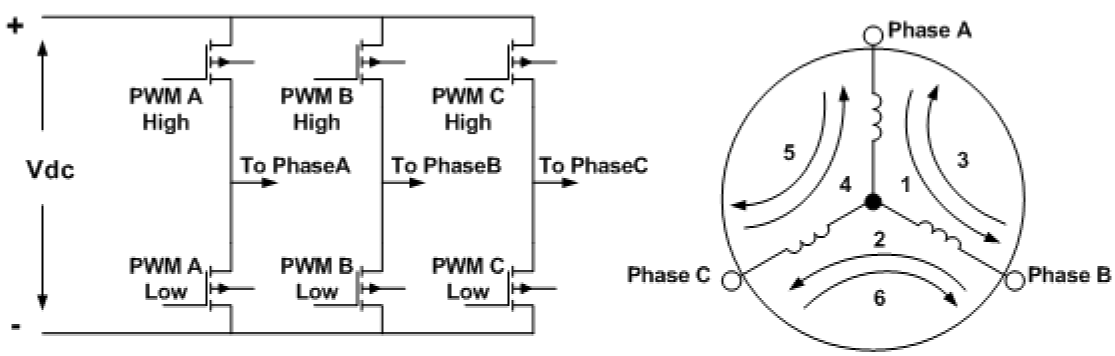

Figure 10 · Commutation Logic with Hall Sensor Inputs

The following table lists the details on the power device switching sequence for a given Hall signal pattern to rotate in clockwise (CW) direction. This table is applicable for the default motor which is shipped with the kit. The switching states '1' and '0' represent the ON and OFF conditions of the switches.

Table 1 · Drive Pattern for CW Direction

Hall3 (H3)	Hall2 (H2)	Hall1 (H1)	PWM C High	PWM C Low	PWM B High	PWM B Low	PWM A High	PWM A Low
1	0	1	0	1	0	0	1	0
1	0	0	1	0	0	0	0	1
1	1	0	1	0	0	1	0	0
0	1	0	0	1	1	0	0	0
0	1	1	0	0	1	0	0	1
0	0	1	0	0	0	1	1	0

The following table gives the drive switching pattern for running the motor in counter clockwise direction. Please note that for CCW direction, the bits of the High-side and Low-side bits which drive each phase winding are just swapped to arrive at the drive switching pattern.

Table 2 · Drive Pattern for CCW Direction

Hall3 (H3)	Hall2 (H2)	Hall1 (H1)	PWM C High	PWM C Low	PWM B High	PWM B Low	PWM A High	PWM A Low
1	0	1	1	0	0	0	0	1
1	0	0	0	1	0	0	1	0
1	1	0	0	1	1	0	0	0
0	1	0	1	0	0	1	0	0
0	1	1	0	0	0	1	1	0
0	0	1	0	0	1	0	0	1

If the Hall sensor inputs are interchanged, refer the table and update the design accordingly.

Speed Control of Brushless DC Motor

The speed of the BLDC motor is directly proportional to the applied voltage. The commutation logic specifies the coils that need to be energized for every 60 degree of electrical revolution based on Hall inputs. The Pulse Width Modulation logic specifies the time intervals during which the switches should be ON and OFF to average the DC Bus voltage applied thereby controlling the Speed. If the switches are ON for the complete duration of the commutation period, then the DC bus rated voltage is directly fed to the phase windings of the motor. Hence the motor will run at the rated speed as specified in the motor datasheet. To operate at any speed below this level, the commutation pattern applied at either the High-side or Low-side switch should be pulse-width modulated with the PWM Pulses at a specified frequency, called the PWM Frequency.

Open Loop Speed Control

In open loop speed control, the duty cycle is directly calculated from the set reference speed and there is no actual speed feedback for control purpose.

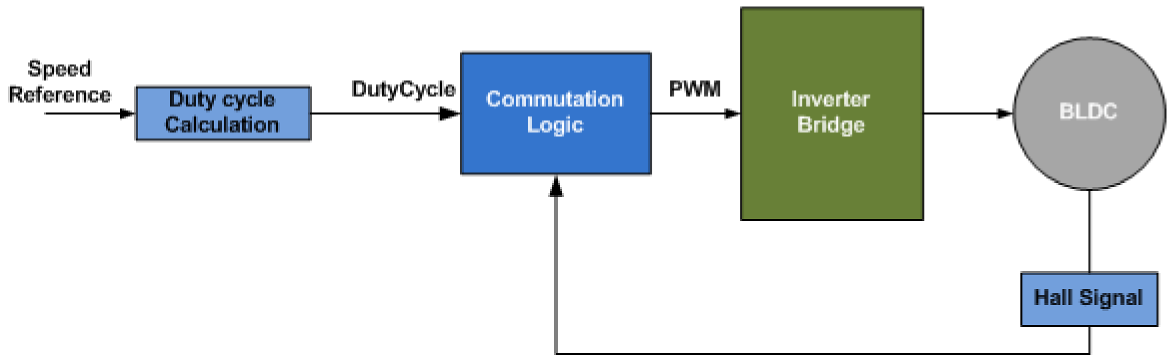

Figure 11 · Open Loop Speed Control

Closed Loop Speed Control

In closed loop speed control, the set speed and the actual speed are compared and the error is fed to the PI controller, which finally outputs the required duty cycle in order to achieve the required speed operation of the motor.

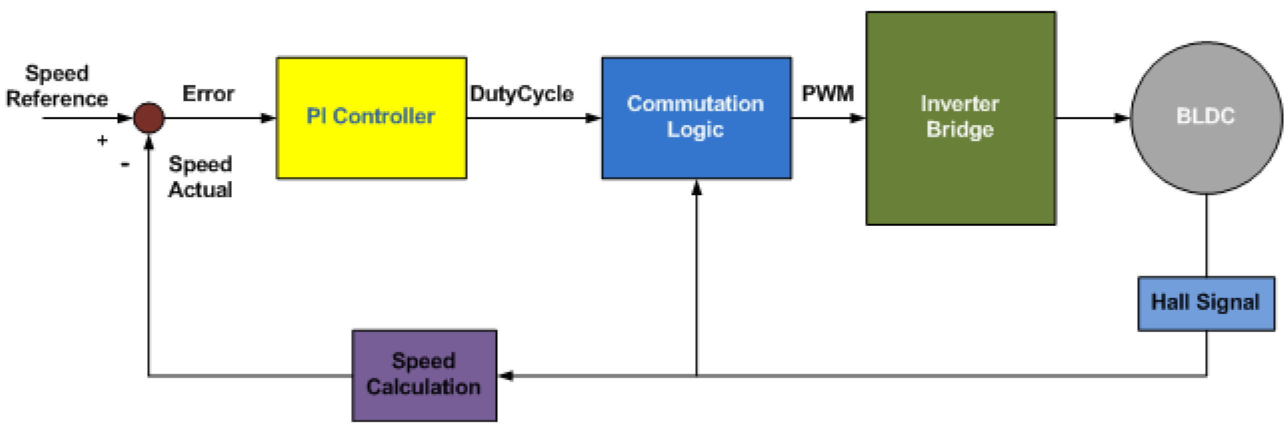

Figure 12 · Closed Loop Speed Control

Proportional-Integral Controller (PI Controller)

The regulation of speed is done with the PI controller. The error difference between the actual speed and reference speed is calculated at every PWM cycle and is given as an input to the PI controller. The proportional and integral gains of the controller are configurable using UART commands.

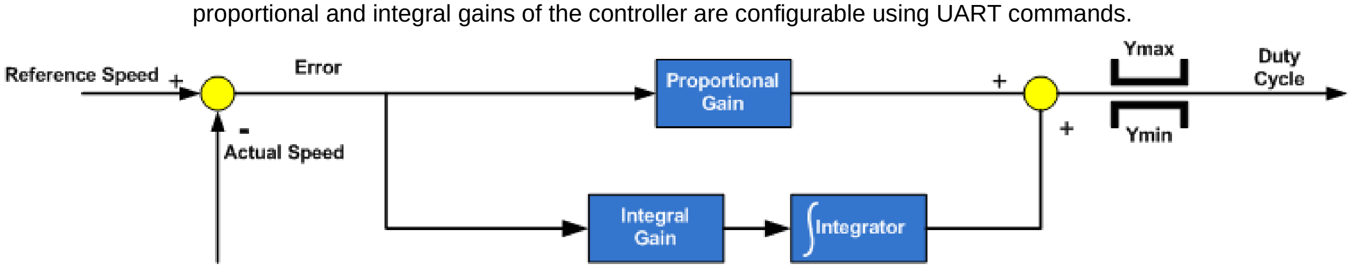

Figure 13 · Proportional-Integral Controller

The duty cycle output from the PI controller is given in continuous time domain as:

$$DutyCycle = K_P * error + K_I * \int error\, dt$$

Equation 2

where,

K_P: Proportional Gain

K_I : Integral Gain

$error$: Difference in Reference speed with Actual speed

$DutyCycle$: Controller Duty cycle Output

In discrete time domain the same PI controller is represented by the following equations.

$$y_n(k+1) = y_n(k) + K_I * e(k)$$

Equation 3

$$Y_n(k+1) = y_n(k+1) + K_P * e(k)$$

Equation 4

where,

K_P: Proportional Gain

K_I: Integral Gain

$e(k)$: Difference in Reference in speed with Actual speed

$Y_n(k+1)$: Current computed dutycycle

$y_n(k+1)$: Current integrated error term

$y_n(k)$: Previously integrated error term

Hardware Configuration and Setup Details

This demonstration design is developed for using the SmartFusion Evaluation Kit Board with an A2F200 SmartFusion cSoC device. The project needs to be recompiled for any new version of the kit/with A2F500 device accordingly.

Connection of SmartFusion Evaluation/Development Kit With Trinamic Kit

When using the SmartFusion Development Kit Board, connect the TMCM-AC-840 Daughter Board to J21 (Mixed Signal Header) via the H3 board-to-board connector.

Switch off all power supplies while connecting/disconnecting the SmartFusion Development/Evaluation Kit Board from the TMCM-AC-840 Daughter Board. Make sure there is no air gap remaining between the SmartFusion board and Trinamic's daughter board as shown in the picture below.

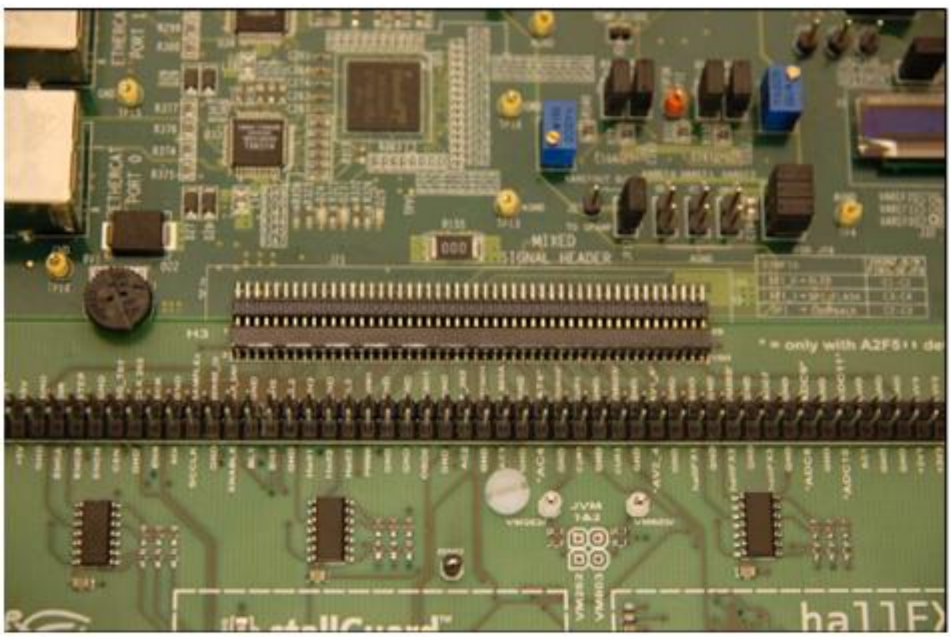

Figure 14 · H3 Board Connector Interface to SmartFusion Kit

Connections for Programming the Kit

Programming with SmartFusion Development Kit

1. Connect the USB cable supplied with the kit to the J15 LCPS interface via the LC programmer board.
2. Connect 5V power supply to J1.

Programming with SmartFusion Evaluation Kit

Connect the USB cable supplied with the kit to the USB program and debug interface.

UART Communication

Connect the USB cable supplied with the kit to the USB/UART interface.

Connection of BLDC Motor with Trinamic Kit

Switch off all power supplies when connecting or disconnecting any motor to/from the TMCM-AC-840 Daughter Board. Connect a BLDC Motor, for example, Qmot QBL4208-41-04-006, to the 3 pole motor connector H1 (UVW). Additionally, Hall Sensor signals can be connected to the 5 pole Hall signals connector H4 (+5V, GND, H1, H2, H3). See the picture below for an example.

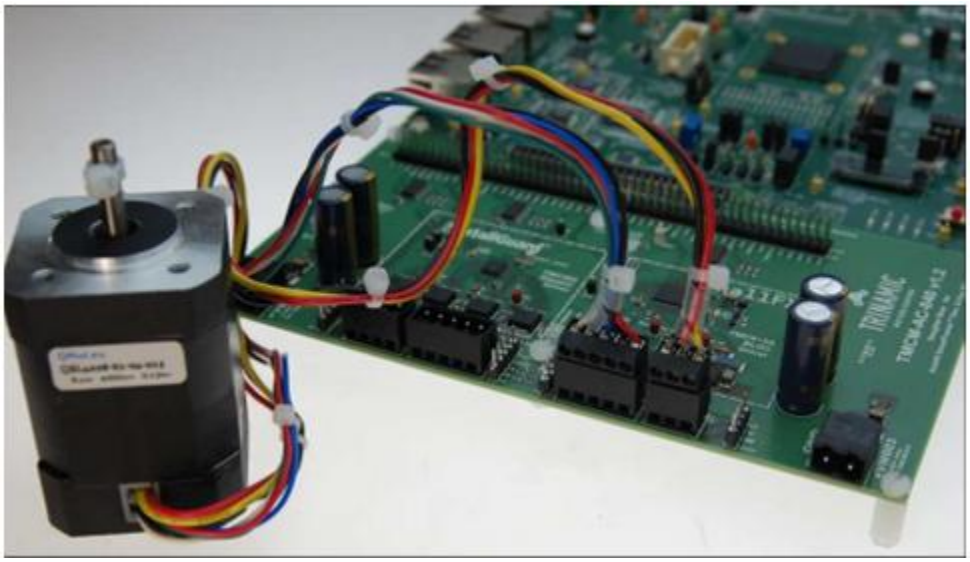

Figure 15 · Connecting BLDC Motor with Trinamic Kit

Motor Wiring Details

Watch the printed information next to the motor connectors on the board. Connect the motor wires accordingly.

Cable type 1		Gauge	Function
Red		UL1007 AWG26	Vcc Hall Sensor +5VDC to +24VDC
Blue		UL1007 AWG26	Hall A
Green		UL1007 AWG26	Hall B
White		UL1007 AWG26	Hall C
Black		UL1007 AWG26	GND Hall Sensor Ground
Yellow		UL1007 AWG20	Phase U
Red		UL1007 AWG20	Phase V
Black		UL1007 AWG20	Phase W

Figure 16 · Motor Wiring Details

Power Supply Connection

Use the H2 connector in TMCM-AC-840 Daughter Board for the power supply and the driver supports up to 48V for VM603 (BLDC driver).

Figure 17 · BLDC Driver Power Supply Connector

This kit comes with the following power supply adaptor and is safe to operate the motor under no-load conditions only.

Note: Any testing carried out with load may cause damage to the power adaptor as it has 1A max current limitation.

Dehner Elektronik SYS 1357-2424 Supply, 24 V/DC/1000 mA

- Manufactured by Sunny Computer Technology Europe
- Input 100-240V, 1.0A max., 50Hz
- Output +24VDC, 1A, 24W max

For higher current ratings (loaded conditions) DC external regulated power supply can be used and the BLDC driver can support for the maximum voltage of 48V and 4A. The power supply connection example is shown in Figure 18.

Figure 18 · Power Supply Connection

Programming the Kit

Program the SmartFusion Evaluation Kit Board or the Development Kit Board with the generated or provided *.STP file using FlashPro. Start HyperTerminal with 57600 baud rate, 8 data bits, 1 stop bit, no parity, and no flow control. If your PC does not have HyperTerminal program, use any free serial terminal emulation program like PuTTY or Tera Term. Refer to the *Configuring Serial Terminal Emulation Programs* tutorial for configuring the HyperTerminal, Tera Term, and PuTTY and then power cycle the board.

User Interface

Below are the user interface options available for configuration and running the motor.

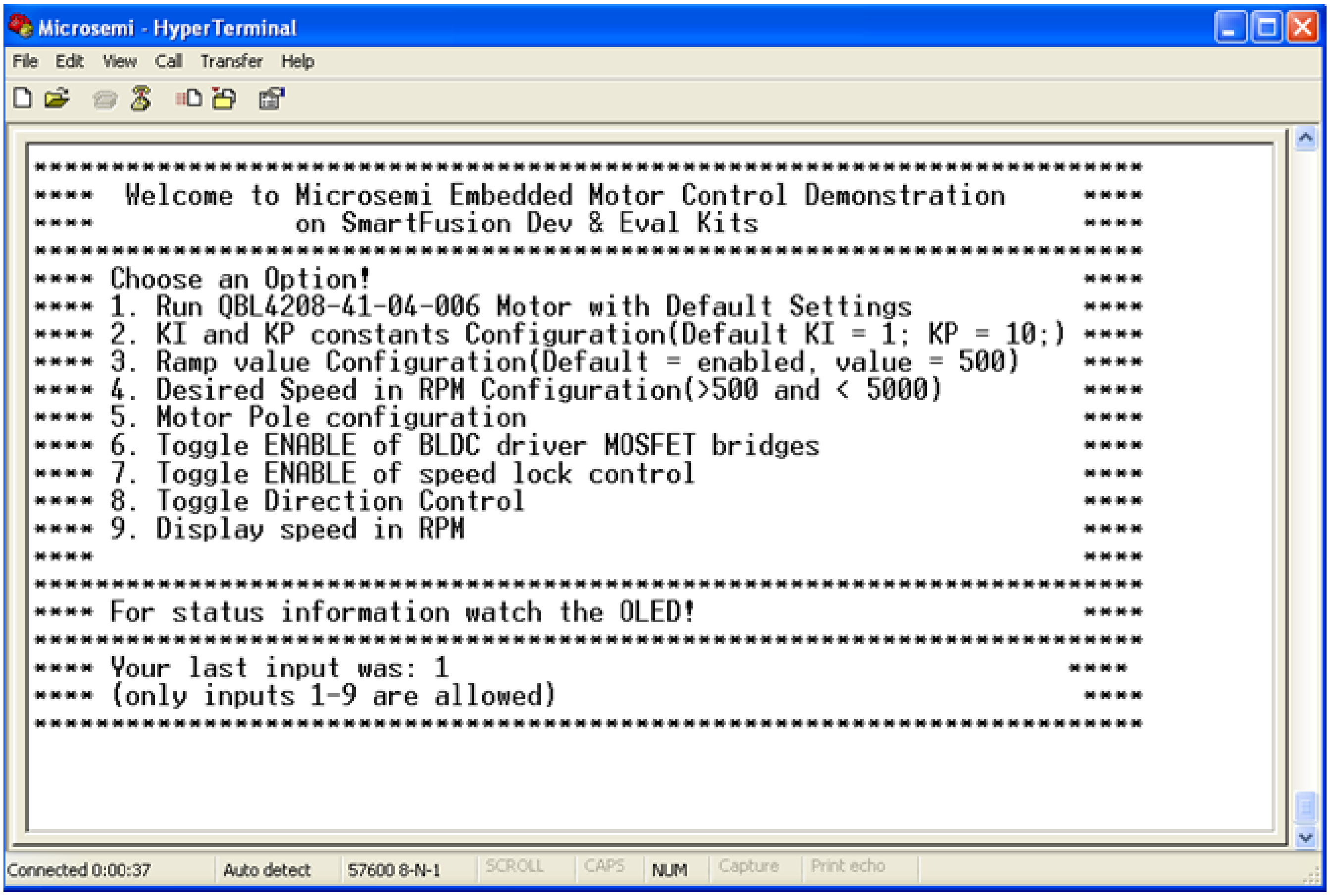

Figure 19 · UI Command Interface Using Hyperterminal

Description of options:

1. Following are the default configurations to run the default motor (QBL4208-41-04-006) that comes with the kit without any additional configurations.
   - KI = 1
   - KP = 10
   - RAMP value = 500 RPM/s
   - Number of motor poles = 8
   - Reference speed = 1000 RPM
   - Direction: Clock wise
2. KI and KP constants Configuration (Default KI = 1; KP = 10) : Press 2 to enter the KI, KP values for the PI controller in fabric, by default the KI is set to 1 and KP is set to 10.
3. Ramp value Configuration (Default = Disabled, if enabled the default value = 500): Press 3 to enable the Ramp profile for motor start, stop, and direction change. The default value of ramp up rate is 500 RPM/s. The maximum value should be less than desired speed. Minimum value should be greater than zero. Higher the RAMP value, lesser is the time to reach the desired speed.

4. Desired Speed in RPM Configuration (>500 and < 5000): Press 4 to enter desired RPM. Typically the value should be between 500 and 5000 RPM. For default motor, the desired speed is 3000 rpm
5. Motor Pole configuration: Press 5 to enter the number of poles of the motor. For Default motor: 8 poles. Refer the motor datasheet for configuring the number of poles. Any wrong configuration of Number of Poles will affect the actual speed calculation and the closed loop operation.
6. Toggle ENABLE of BLDC driver MOSFET bridges: Press 6 to toggle the Enable/Disable the motor.
7. Toggle ENABLE of speed lock control: Press 7 to toggle the speed lock of motor. The speed lock option enables the motor to achieve the desired speed, once the motor achieves the desired speed then the closed loop control will be disabled and the MSS will be free from the closed loop control.
8. Toggle Direction Control using: Press 8 to toggle the direction of motor.
9. Display speed in RPM: Press 9 to see the actual speed.

Design and Implementation

The design blocks of the closed loop speed control operation of BLDC motors is shown in Figure 19. The block commutation algorithm is completely implemented in Fabric like PWM Unit, Commutation logic, Hall pattern detection, Hall event counter, PI controller etc. Only control & monitoring functions are implemented in MSS. The dotted lines shown in the figure are optional design that you can develop if required. Here the ACE is configured for the measurement of the three phase currents and can be extended if overcurrent detection or speed inner current control is required.

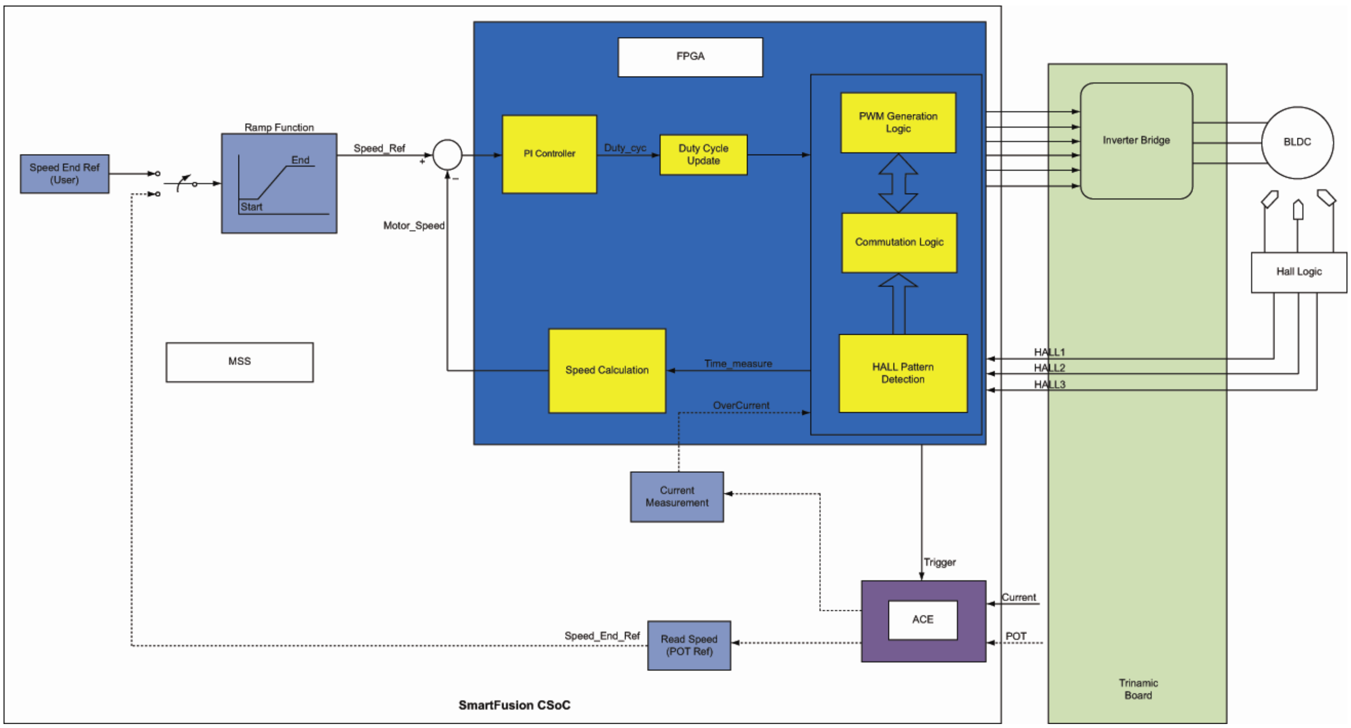

Figure 20 · Design Blocks of Closed Loop Speed Control Operation of BLDC Motors

Hardware Implementation

- tmc603_pwm.vhd: This module generates fixed 8 KHz PWM signal; the duty cycle is taken as input from the Cortex-M3. This module operates on 75 MHz FAB clock. The width of the PWM counter is 14 bit and it is an edge aligned signal.
- PI_controller.vhd: This module operates on 75MHz FAB clock and generates the ON time duration for the inverter bridge. The implementation of this module is the same as the functionality mentioned in the Proportional-Integral Controller (PI Controller) section. In this Mircosemi's standard macros 32-Bit adder, 16x16 signed multipliers are used. It takes about ~ 20 clock cycles to complete the operation. The final result (Duty cycle) is directly fed to PWM generator.
- dirswt.vhd: This module selects the High-and Low-side switches of the inverter bridge based on the direction.
- hall_chk.vhd: This decodes the Hall pattern, identifies the direction, and generates the enable signal for speed calculation.
- mux2to1x3.vhd: This module selects the Hall signals directly from the motor or emulated signals from the Trinamic daughter board.

- hall2uvw.vhd, gtcntl_uvw.vhd, and chopcntl_uvw.vhd: These modules select the ON/OFF states of the inverter bridge.
- tmc603_commutator_top.vhd: This module integrates all the sub-modules and it has the speed calculation logic. Speed calculation uses the moving average method which uses the counts between Hall events. The counts are based on the Fabric Clock, which is stored in a buffer that keeps the last six event counts. The values are passed to the Cortex on request basis.
- bridge_APB3_to_userlogic.vhd: This module decodes the all APB transaction from the Cortex-M3 processor.

Software Implementation

The MSS operates at 75 MHz frequency and does the job of controlling and monitoring parameter configurations of the block commutation control algorithm only. The complete motor control block commutation algorithm is implemented in fabric. Hence, the MSS consumes very less CPU time (< 7.5 µs) and allows the system level application to work independently. For a PWM Frequency of 20 kHz, the CPU load will be < 15% and for a PWM Frequency of 8 kHz, the CPU load will be < 6 %.

The following is the list of functions available in MSS:

1. init_system() function: This function initializes the different peripherals and some parameters used for control and monitoring purposes.
2. OLED_statrtup_message() function: This displays the startup message on the OLED.
3. show_main_menu() function: This shows the Menu options on the HyperTerminal through UART.
4. update_OLED() function: This updates the OLED with desired data.
5. process_uart_data() function: This function processes the option selected on Hyper terminal.
6. Timer1_IRQHandler ISR routine function: The Timer 1 is configured to operate at the PWM Frequency and generates interrupt on overflow. The Timer Interrupt hander reads the speed values from the fabric and calculates the actual speed in terms of revolutions per minute (RPM). This actual speed calculated is then passed to the PI controller. The speed calculation function can also be implemented in Fabric and the necessary scaling should be taken care for the actual and reference speeds.

```
__attribute__((__interrupt__)) void Timer1_IRQHandler(void)
{
    uint32_t TMC603_duty_cycle;
    float actual_speed_cal_temp;
    if(g_PI_Speed_en == 1)
    {
        g_actual_speed = HW_get_32bit_reg( TMC603_SPEED_VAL_REG_ADDR );
        actual_speed_cal_temp = (float)((g_actual_speed*g_pole_pair)>>1); /* For 4Pole Motor */
        actual_speed_cal_temp = (FAB_FEQ)/actual_speed_cal_temp;
        actual_speed_cal_temp = actual_speed_cal_temp*60;
        g_actual_speed = (unsigned int )actual_speed_cal_temp;
        TMC603_duty_cycle = PIControllerSpeed(&err_integration,(unsigned int)g_reference_ramp_speed,(unsigned int)g_actual_speed,(uint32_t)(g_KP<<9),(uint32_t)((g_KI<<15)/1000));
    }
    MSS_TIM1_clear_irq();

}
```

Figure 21 · Timer1 ISR for Speed Calculation and PI Controller Function Calls

7. PIControllerSpeed() function: This function is called in the Timer 1 interrupt handler. This function gets the KI, KP, actual speed, and reference speed as inputs. It calculates the error and passes the error to the PI controller in the fabric. This function also integrates the ramp up, ramp down profile for motor start, stop, and direction change based on the ramp profile enable selection. In some of the applications, a soft start/stop or acceleration of the motor is required. The reference speed is incremented or decremented until the required speed reference is reached at the rate of the ramp up speed specified or configured. However, the functionality can be disabled if required.

The Speed Ramp up/down functionality is shown in Figure 22. The speed end reference is incremented or decremented based on the ramp rate you specify until the set reference speed is reached and the flag increment_speed is cleared with the value 0. As this function is called at every PWM period, the variable g_ramp_counter is incremented by 1 and compared with the g_ramp_count. Whenever the g-ramp-counter value exceeds the g_ramp_count, the flag increment_speed is set to 1.

```
unsigned int PIControllerSpeed(long *err_integrator_buff_ptr,unsigned int reference,unsigned int act,uint32_t speed_kp, uint32_t speed_ki)
{
                int pierror;
                uint8_t increment_speed;
                g_ramp_counter++;
                if(g_ramp_counter >= g_ramp_count)
                {
                    g_ramp_counter = 0;
                    increment_speed = 1;
                }
                if(increment_speed == 1)
                {
                  increment_speed = 0;
                  if(g_stop == 1)
                  {
                      if(g_reference_ramp_speed == 0)
                      {
                        HW_set_32bit_reg( CTRL_REG_1_ENN_TMC603_ADDR, 0 );
                      }
                      else
                      {
                          g_reference_ramp_speed--;
                      }
                  }else if(g_dir_change == 1)
                  {
                      if(g_reference_ramp_speed == 10)
                      {
                        HW_set_32bit_reg( CTRL_REG_3_CTRL_SELEC_ADDR, CTRL_REG_3_CTRL_SELEC_VAL );  // write updated value
                        g_dir_change = 0;
                      }
                      else
                      {
                          g_reference_ramp_speed--;
                      }
                  }
                   else if(g_reference_ramp_speed > g_reference_speed)
                        {
                          g_reference_ramp_speed--;
                        }
                        else if(g_reference_ramp_speed < g_reference_speed)
                        {
                           g_reference_ramp_speed++;
                         }
                }
                pierror   =   ( g_reference_ramp_speed - act);
                HW_set_32bit_reg( TMC603_PI_ERR_REG_ADDR, pierror );
}
```

Figure 22 · PI Controller Function

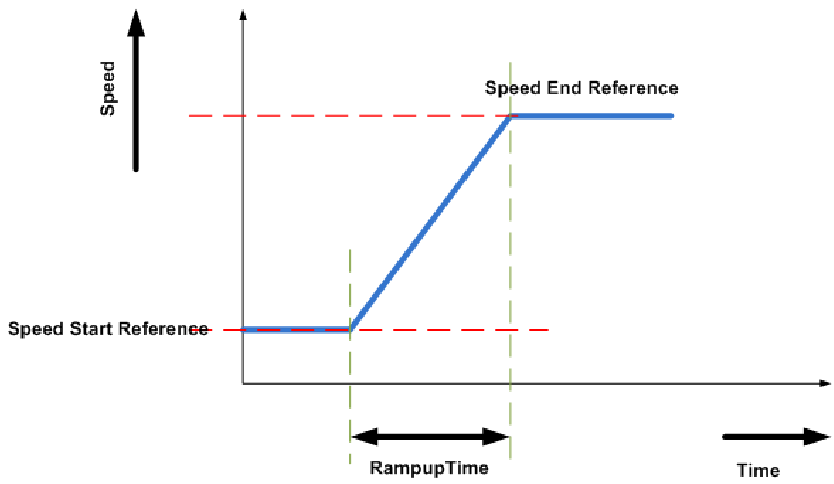

Figure 23 · The Speed Ramp Up/Down Functionality

Conclusion

This User’s Guide uses the features of SmartFusion cSoC FPGAs to develop an effective motor control demo by partitioning the algorithms and implementing it in MSS and Fabric effectively. Having the functional blocks in Fabric, the CPU is offloaded and the MSS can perform any other system level operations. This demo block commutation application has all the control functions implemented in Fabric and MSS made to perform only control and monitoring functions.

Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call **800.262.1060**
From the rest of the world, call **650.318.4460**
Fax, from anywhere in the world **650. 318.8044**

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Microsemi SoC Products Group Customer Support website for more information and support (http://www.microsemi.com/soc/support/search/default.aspx). Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on website.

Website

You can browse a variety of technical and non-technical information on the Microsemi SoC Products Group home page, at http://www.microsemi.com/soc/.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

50200301-0/02.12