SmartFusion Field Oriented Control of Permanent Magnet Synchronous Motors Using HALL and Encoder

User’s Guide
# Table of Contents

Introduction .......................................................... 5

1 Permanent Magnet Synchronous Motor ............................ 7
   Sinusoidal Voltage and SVPWM Generation .................. 12
   PWM Generation .................................................. 14
   Phase Current Measurement Using ADC ...................... 16
   Angle Estimation Using Hall Sensor State and Quadrature Encoder .................................................. 19
   Quadrature Encoder Overview .................................. 20

2 Design and Implementation ........................................ 23
   Design Description ............................................... 23
   Software Implementation ......................................... 26
   Hardware Implementation ........................................ 30
   SVPWM Generation ............................................... 31
   Design Customizations .......................................... 32
   Performance Details ............................................ 34

3 Hardware Configuration and Setup Details ..................... 35
   Programming the Kit ............................................ 35
   Connecting the SmartFusion Evaluation/Development Kit with the Trinamic Kit ..................................... 38
   Connections for Programming the Kit ......................... 38
   Connecting the BLDC Motor with the Trinamic Kit .......... 39
   Power Supply Connection ....................................... 40

A User Interface ..................................................... 39
   Conclusion ....................................................... 41

B Design and Programming Files ................................ 47

C Product Support .................................................... 49
   Customer Service ................................................ 49
   Customer Technical Support Center ............................ 49
   Technical Support ................................................ 49
   Website ................................................................... 49
   Contacting the Customer Technical Support Center ....... 49
   ITAR Technical Support ......................................... 50

Index ................................................................. 51
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) field programmable gate array (FPGA) devices contain a hard embedded microcontroller subsystem (MSS), 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).

SmartFusion cSoC devices have major advantages in terms of fabric, MSS, and analog compute engine (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.

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 Field Oriented Control of PMSM/BLDC motors using both HALL and Encoder sensors, and this design is developed based on the following hardware platforms:

- 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.
- Trinamic TMCM-AC-840-Motor Control Daughter Board Kit (TMCM-AC-840).

Reference documents:
1. SmartFusion cSoC user's guides & manuals
   (www.microsemi.com/soc/products/smartfusion/docs.aspx)
2. SmartFusion Development Kit Board User's Guide
   (www.microsemi.com/documents/A2F500_DEV_KIT_UG.pdf)
   (www.microsemi.com/documents/A2F_EVAL_KIT_UG.pdf)
5. Trinamic 603A chip User Manual
   (www.trinamic.com/tmctechlibcd/integrated_circuits/TMC603/tmc603_datasheet.pdf)
6. BLDC Motor Datasheet
7. Encoder Datasheet
   (www.usdigital.com/products/e2)
Permanent magnet synchronous motor (PMSM) is a rotating electrical machine that has the stator phase windings and rotor permanent magnets. The air gap magnetic field is provided by these permanent magnets and hence it remains constant. The conventional DC motor commutates itself with the use of a mechanical commutator whereas PMSM needs electronic commutation for the direction control of current through the windings. As the PMSM motors have the armature coils at the stator, they need to be commutated externally with the help of an external switching circuit and a three phase inverter topology is used for this purpose.

The PMSM and the driving inverter bridge topology are shown in Figure 1-1.

Figure 1-1 • PMSM Motor and Driving Inverter Topology

The torque is produced because the interaction of the two magnetic fields causes the motors to rotate. In permanent magnet motors, one of the magnetic fields is created by permanent magnets and the other is created by the stator coils. The maximum torque is produced when the magnetic vector of the rotor is at 90 degrees to the magnetic vector of the stator.

In the Block Commutation technique, there are only two coils conducting at any given point of time and the third coil remains unexcited. This causes a misalignment between the stator and rotor magnetic fields and hence ripples are introduced to the torque. The torque ripples produce more noise, vibration, and wear to the system.
In the Sinusoidal Commutation technique, all the three coils conduct at any given point of time. The currents through the coils are Sinusoidal and are phase displaced by 120 degrees. The torque produced is smooth and has no or less torque ripples. The torque production in Sinusoidal and Trapezoidal motors is shown in Figure 1-2.

Field Oriented Control Theory

In DC motors, the flux and torque producing currents are orthogonal and can be controlled independently. The magneto motive forces, developed by these currents are also held orthogonal. The torque developed is given by the equation:

\[ T_e = K_a \phi (I_a) I_a \]

EQ 1-1

where

\( \phi (I_a) \): flux

\( I_a \): armature current

Hence, the flux is only dependent on the field winding current. If the flux is held constant, then the torque can be controlled by the armature current. For this reason DC machines are said to have decoupled or have independent control of torque and flux.

In AC machines, the stator and rotor fields are not orthogonal to each other. The only current that can be controlled is the stator current. Field Oriented Control (FOC) is the technique used to achieve the decoupled control of torque and flux by transforming the stator current quantities (phase currents) from stationary reference frame to torque and flux producing current components in rotating reference frame.

Advantages of FOC:

- Transformation of a complex and coupled AC model into a simple linear system
- Independent control of torque and flux, similar to a DC motor
- Fast dynamic response and good transient and steady state performance
- High torque and low current at startup
- High Efficiency
- Wide speed range through field weakening

FOC technique involves three reference frames and needs transformations from one to the other.

1. Stator reference frame (a, b, c) in which a, b, and c are co-planar, at 120 degrees to each other.
2. An orthogonal reference frame (ab) in the same plane as the stator reference frame in which the angle between the two axes is 90 degrees instead of 120 degrees. The a-axis is aligned with a-axis in the second frame.
3. Rotor reference frame (dq), in which the d-axis is along the N and S poles or along the flux vector of the rotor and the q-axis is at 90 degrees to the d-axis.

Figure 1-3 shows the transformations done for decoupling the stator currents into the torque producing (I_q) and flux producing (I_d) components.

---

**Figure 1-3 • Transformations and Reference Frames**

The combined representation of the quantities in the entire reference frames is shown in Figure 1-4.

---

**Figure 1-4 • Combined Vector Representation of All Transformed Quantities**
Once the torque and flux producing components are controlled with the PI controller, the controlled outputs (-the voltages), are then transformed back (inverse transformation) to the stator reference frame.

Transformations
In FOC, the components $I_q$ and $I_d$ are referenced to the rotating reference frame. Hence, the measured stator currents have to be transformed from the three phase time variant stator reference frame to the two axis rotating dq rotor reference frame. This can be done in two steps as shown in Figure 1-6.

The transformation from the 3-phase 120 degree reference frame to two axis orthogonal reference frame is known as Clarke transform. Similarly the transformation from two axis orthogonal reference frame to the two axis rotating reference frame is known as Park transform.
Clarke Transformation

The measured motor currents are first translated from the 3-phase reference frame to the two axis orthogonal reference frame. The transform is expressed by the following equations.

\[
I_\alpha = I_a \\
I_\beta = (I_a + 2I_b)/\sqrt{3}
\]

Where

\[
I_a + I_b + I_c = 0
\]

Park Transformation

The two axis orthogonal stationary reference frame quantities are then transformed to rotating reference frame quantities. The transform is expressed by the following equations:

\[
I_d = I_\alpha \cos \theta + I_\beta \sin \theta \\
I_q = I_\beta \cos \theta - I_\alpha \sin \theta
\]

PI Controller

The regulation of Speed, Torque, and Flux are done with the PI controller. The error difference between the actual and reference 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 GUI.

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

\[
Output = K_P \times \text{error} + K_I \times \int \text{error} \, dt
\]

where,

\[
K_P: \text{Proportional gain} \\
K_I: \text{Integral gain}
\]

error: Difference between reference and actual
In discrete time domain the same PI controller is represented by the following equations:

\[ y_n (k+1) = y_n (k) + K_I \times e(k) \]
\[ Y_n (k+1) = y_n (k+1) + K_P \times e(k) \]

where,

- \( K_P \): Proportional gain
- \( K_I \): Integral gain
- \( e(k) \): Difference between reference and actual
- \( Y_n (k+1) \): Currently computed output
- \( y_n (k+1) \): Currently integrated error term
- \( y_n (k) \): Previously integrated error term

**Inverse Park Transformation**

The outputs (\( V_d \) and \( V_q \)) of the PI controllers provide the voltage components in the rotating reference frame. Thus an inverse of the previous process has to be applied to get the reference voltage waveforms in stationary reference frame. At first, the quantities in rotating reference frame are transformed to two axis orthogonal stationary reference frame using Inverse Park transformation.

The Inverse Park transformation is expressed by the following equations:

\[ V_\alpha = V_d \cos \theta - V_q \sin \theta \]
\[ V_\beta = V_q \cos \theta + V_d \sin \theta \]

**Inverse Clarke Transformation**

The transformation from two axis orthogonal stationary reference frame to the three phase stator stationary reference frame, is accomplished using the Inverse Clarke transformation. The Inverse Clarke transformation is expressed by the following equations:

\[ V_a = V_\alpha \]
\[ V_b = [-V_\alpha + \sqrt{3}V_\beta]/2 \]
\[ V_c = [-V_\alpha - \sqrt{3}V_\beta]/2 \]

**Sinusoidal Voltage and SVPWM Generation**

The output of the Inverse Clarke transformation provides the duty cycles of the Pulse-width modulation (PWM) channels that corresponds to the three Phase voltages. For sinusoidal excitation of the phase voltages, these duty cycle values are used directly as shown in Figure 1-8 on page 13. There are many conventional ways of implementing the SVPWM algorithm available. A simplified approach which is equivalent to the conventional modulation strategy is used in this current implementation.
In this approach the instantaneous average of the minimum and maximum of all the three phase voltages is calculated as the Voltage offset. Then this instantaneous Voltage offset is subtracted from the instantaneous three phase voltages.

\[
V_{\text{off}} = \frac{\min(V_a, V_b, V_c) + \max(V_a, V_b, V_c)}{2}
\]

\[
V_{a}^* = V_{a} - V_{\text{off}}
\]

\[
V_{b}^* = V_{b} - V_{\text{off}}
\]

\[
V_{c}^* = V_{c} - V_{\text{off}}
\]

EQ 1-8

The final voltage waveforms are shown in Figure 1-9.
PWM Generation

Generation of three phase edge-aligned PWM is supported in the demo design. Dead time insertion logic is included in order to avoid the short circuit of the inverter's high and low side switches. Total of 6 PWM signals generated, 3 for the high side, and 3 low for side switches. The PWM for high and low side switches are complementary for the same inverter leg.

The principle of operation of edge-aligned PWM is shown in Figure 1-10.

The PWM generation mainly depends on the following:

- PWM counter
- Compare value
- Active state value

The PWM counter is loaded with the value calculated based on the PWM frequency required.

For example, with the Fabric frequency of 75 MHz and if the required PWM frequency is 20 kHz, then the PWM counter value is given by:

\[
PW\text{M count value} = \frac{\text{Fabric Frequency}}{\text{PWM Frequency}} = \frac{75 \times 10^6}{20 \times 10^3} = 3750
\]

EQ 1-9

The PWM counter counts from zero to the PWM count value. When it reaches the PWM count value it achieves the PWM period and it starts counting from zero again.

Compare value is the duty cycle calculated for each of the Phases. Each phase has different duty cycle values and hence three compare value variables. When the PWM counter value reaches the compare value the PWM output is toggled and remains in its state till the PWM period match occurs and toggle its state.

Typically on a period match the PWM output puts in its active state (ON state) until a compare match occurs. On a compare match the state puts in its passive state (OFF state) till the next period match occurs.

The active state value (either ‘1’ or ‘0’) is decided based on the driver IC and Switching device characteristics. However the states can be modified in the design for application specific needs.
Phase Current Measurement Using ADC

The three phase currents can be measured using external current shunt resistors available at the lower side of the metal-oxide-semiconductor field-effect transistor (MOSFET) switches of the inverter. Three 10 mΩ resistors are used for the measurement of the phase currents. One end of the shunt resistor is connected to the low-side MOSFET and the other end is connected to the GND of DC Bus Voltage. The voltages across the shunt resistors are seen at the connections Rsense1, Rsense2, and Rsense3.

Figure 1-11 • Three Phase Current Measurement with Shunt Resistors
The measured voltages across the shunt resistors are amplified to match the ADC input voltage ranges. The amplified currents CUR1, CUR2, and CUR3 from the TMCM 840 board are connected to the ABPS inputs of the SmartFusion device.

Table 1-1 • Current Measurement Pin Configuration

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>TMC603 Pin</th>
<th>SmartFusion Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUR1</td>
<td>61</td>
<td>W9</td>
<td>ABPS2 input and voltage range [±2.56]V</td>
</tr>
<tr>
<td>CUR2</td>
<td>62</td>
<td>AB7</td>
<td>ABPS3 input and voltage range [±2.56]V</td>
</tr>
<tr>
<td>CUR3</td>
<td>65</td>
<td>W12</td>
<td>ABPS6 input and voltage range [±2.56]V</td>
</tr>
</tbody>
</table>

The measured currents are amplified using the built-in op-amp circuitry available with the TMC603 device. Figure 1-12 shows the internal op-amp circuit available with TMC603 device.

![Figure 1-12 • Current Measurement Amplifier](image)

The TMC603 CURx outputs deliver a signal centered to 1/3 of the 5 V VCC supply. This allows the measurement of both the negative and positive signals. The current amplifier is an inverting type.

For the zero voltage input the amplifier outputs a value of 1.667 mV. An amplification factor of 4.5 or 18 is selectable and it is selected as 18 in the current design.

\[
\text{Voltage across shunt resistor} = R_{\text{shunt}} \times \text{Current through the resistor}
\]

\[EQ \ 1-10\]

For a current of 1A flowing through the resistor,

\[
\text{Voltage across shunt resistor} = 0.01 \times 1 = 0.01 \text{ V or 10 mV.}
\]

\[
\text{Voltage seen at the analog-to-digital converter (ADC) input} = \text{Zero Offset Voltage} + \text{Amplified voltage}
\]
\[
= \text{Zero Offset Voltage} + (\text{Amplification Gain} \times \text{Voltage across shunt resistor})
\]
\[
= 1.6667 + (18 \times 0.01)
\]
\[
= 1.6667 + 0.18
\]
\[
= 1.8467 \text{ Volts}
\]

\[EQ \ 1-11\]

The voltage variation at the ADC input is about 0.2 V per 1A current flowing through the shunt resistor. The active bipolar prescalers (ABPS) inputs are configured for the input voltage range of ±2.56 Volts. The maximum positive voltage range possible with the circuit is 2.56 - 1.6667 V = 0.9 V

The maximum current allowed is 0.9/0.18 = 5 A

Few motors under no-load operation may draw very low current of the order of 50 mA to 100 mA.
At 100 mA, the voltage difference seen at the ADC input from zero current offset is given below:
\[
\text{Voltage difference} = 0.1 \times 0.01 \times 18 = 0.018 \text{ V} = 18 \text{ mV}.
\]
No-load motor operation can be a problem not only at high speed operations but can be even challenging at very low speed operations. The motor QBL4208-41-04-006 draws very low current and hence it is recommended to run the demo with a small amount of load at the motor shaft.

The CUR1, CUR2, and CUR3 are connected to the ABPS channels of the SmartFusion by default. In case of ABPS the 12 bits are used to represent -2.56 V to 2.56 V. Each bit represents 0.3125 mV, whereas in case of ADC direct channels, 12 bits are used to represent 0 to 2.56 V. Each bit represents 0.625 mV.

The current design supports two phase current measurements, which are CUR1 and CUR2 measurements by connecting to the ADC direct channels, the resolution of the measured currents can be improved by double and thus improving the motor performance. This can be done with small wiring on the board as follows:
- Connect CUR1 Pin of the TMC daughter card Header K1 to the Pin ADC7 on the same header.
- Connect CUR2 Pin of the TMC daughter card Header K1 to the Pin AC1 on the same header.

### Table 1-2 • Current Measurement Pin Configuration

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>TMC603 Pin</th>
<th>SmartFusion Pin</th>
<th>Signal Name</th>
<th>TMC603 Pin</th>
<th>SmartFusion Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUR1</td>
<td>61</td>
<td>W9</td>
<td>ADC7</td>
<td>91</td>
<td>T12</td>
<td>ADC Direct input and voltage range [0 to 2.56] V</td>
</tr>
<tr>
<td>CUR2</td>
<td>62</td>
<td>AB7</td>
<td>AC1</td>
<td>82</td>
<td>U9</td>
<td>ADC Direct input and voltage range [0 to 2.56] V</td>
</tr>
</tbody>
</table>

### ADC Current Sampling Window

The low-side current measurements are done when the low-side switches are ON. There are practical limitations in terms of current measurement. If the ON time duty cycle of any of the phases is high, then the ON time of the corresponding low-side switch is low and current measurements are not possible.

The sample window available mainly depends on ADC sampling time, the ON time of low-side switches, dead time, and switching transient time. This limits the high speed operations of the motor.

The current design supports two direct ADC phase current measurements of Current A and Current B. The current sampling is triggered with the delay of about 6 µs after all low-side switches are ON. The 6 µs delay provides the time for dead time logic and current to reach for stable value due to switching transient.
The current trigger timing in edge-aligned PWM is shown in Figure 1-13.

For proper Sinusoidal commutation, the absolute rotor position information is very crucial in order to produce synchronized voltage waveforms for the motor. To get fine position information, the motors are equipped with Encoders or Resolvers. The quadrature encoder gives relative position of the rotor, but to perform FOC, it is necessary to know the absolute position. The absolute position can be established using Hall and Encoder.

This demo design uses the motor, which has two pole pairs. For every electrical cycle there are six Hall state changes that provides 60 degrees of precision. Each state corresponds to an electrical angle and we have six angle steps.
Figure 1-14 shows the Hall event changes and the corresponding angle information of the motor used in the demo design.

Figure 1-14 • Hall State Change Vs Corresponding Angle for CCW Direction

The Hall state change provides the information of the new electrical angle. Commutate the motor with angle information from HALL only and wait and HALL state reaches the Zero angle position. Then, log the Encoder count as zero angle count, and increment the angle using the Encoder count.

Quadrature Encoder Overview

Usually the quadrature encoders are incremental encoders. The incremental encoder uses two channels to sense the position of the rotor. The two channels are 90 degree apart from each other. The incremental encoder simply generates the equally distributed constant pulses per revolution. Using two channels the relative position, direction, and speed of the rotor can be derived.

Figure 1-15 shows the output of two channels of incremental encoder for clockwise and anti-clockwise directions.

Figure 1-15 • Quadrature Encoder
Some of the incremental encoders provide a third output called index signal, which generates a single pulse per revolution. This can be used to correct the counts if there are any accumulated offset errors. The advantage of the incremental encoder is its capability of producing high resolution position information. Some encoders are capable of producing 10,000 pulses per revolution at higher speeds. In this design the quadrature encoder generates 360 pulses per revolution per channel.

**Figure 1-16** gives the angle calculation using HALL and Encoder.

\[
\text{New Angle} = (\text{Present Count} - \text{Zero angle count} + \text{maximum count})\%\text{Maximum Count}
\]

*EQ 1-12*
**Speed Calculation using Hall State**

The Hall state changes provide the information on the position of the motor. For every electrical cycle there are six Hall state changes. If the time taken between the Hall state changes is available then the speed can be computed. A moving average technique is used to calculate the speed and this technique averages the six of the previously calculated speed values.

\[
\text{ActualSpeed}(n) = \frac{(\text{Speed}(n) + \text{Speed}(n-1) + \ldots + \text{Speed}(n-5))}{6}
\]

*EQ 1-13*
The design blocks of the speed control operation of BLDC/PMSM motors using FOC is shown in Figure 2-1.

**Figure 2-1 • FOC Block Diagram in SmartFusion**

**Design Description**

The blocks of the FOC algorithm are architected using the Cortex-M3 processor in the MSS and FPGA fabric. The blocks are partitioned based on the best utilization of the available resources, speed, and power. For example, the speed calculation is done using the hard timer available in the MSS, which subsequently saves the FPGA logic compared to the speed calculation using FPGA logic.
The flowchart of the complete FOC algorithm is shown in Figure 2-2 and Figure 2-3.

Figure 2-2 • Software Flow of FOC
After the reset, the Cortex-M3 processor does all the initialization of all peripherals and FPGA fabric. Then it starts communicating with graphical user interface (GUI) running on host PC for all control and monitoring commands.

Figure 2-3 • Hardware Flow of FOC
Software Implementation

The MSS operates at 75 MHz frequency and performs the following functions of the FOC:

- Interface with GUI
- ADC results reading and converting to actual currents
- ACE reconfigurations
- Speed calculation using Hard Timer
- Speed ramp functionality
- Clarke transformation
- Angle calculation
- Park transformation
- Speed PI controller
- Torque and Flux PI Controllers
- Inverse Park Transformation
- Inverse Clarke Transformation
- SVPWM

The SW blocks in the Figure 2-2 on page 24 shows the program flow sequence of the control and monitoring algorithm implemented in software. After the reset, the initialization of the MSS peripherals and parameters are done. After the initialization a while loop runs indefinitely. Cortex-M3 processor receives the following interrupts from various sources and each interrupt is processed with corresponding functionality.

- Interrupt from the universal asynchronous receiver/transmitter (UART): UART data from the GUI is processed. The GUI communicates with MSS through UART to program the configuration and control parameters of motor. Once the configuration and control parameters are programmed, then it enables the PWM generation in fabric. The fabric generates an interrupt to the ACE to sample the current values. After the sampling is done the ACE generates an interrupt to the MSS.

- Interrupt from general purpose input/output (GPIO) 1: This indicates HALL signal changes, which come from the FPGA fabric. Current count of the configured timer is read and speed calculation is done in this interrupt. The timer is reconfigured for the next speed calculation. This speed is the actual speed of the motor and it is passed to the Speed PI controller as one of the inputs to calculate the speed error from the reference speed.

- Interrupt from the ACE: This indicates the availability of ADC sampling results for motor phase currents. The FPGA fabric triggers the ACE for sampling of motor phase currents, when all the bottom switches of the inverter bridge are ON. This is the starting point for the FOC calculations. Once the sampling of the motor phase currents are done, the ACE issues an interrupt to the Cortex-M3 processor. The Cortex-M3 processor reads the results and converts the ADC results into the actual currents. Then the FOC calculations are done in this Interrupt. Below are the functions executed in the interrupt handler.
  - Speed PI Controller
  - Clarke Transformation
  - Angle calculation
  - Park transformation
  - Speed PI controller
  - Torque and Flux PI Controllers
  - Inverse Park Transformation
  - Inverse Clarke Transformation
  - SVPWM

- Interrupt from FPGA fabric: This indicates the PWM tick, which is at 20 kHz rate. In this interrupt handler, the ACE sequences are reconfigured for the motor phase currents during the current PWM period when all the low-side switches are ON.
Table 2-1 shows the details of the functions implemented in MSS:

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>init_system()</td>
<td>This function initializes the different peripherals of MSS.</td>
</tr>
<tr>
<td>2</td>
<td>uart0_rx_handler()</td>
<td>This is the interrupt handler and it collects 3-Bytes from the GUI running on host PC. The first byte is ID and the subsequent two bytes are data.</td>
</tr>
<tr>
<td>3</td>
<td>UartMsgHandler()</td>
<td>This function decodes all the parameters and configures the different sections of the algorithm, which are programmed from the GUI running on host PC.</td>
</tr>
<tr>
<td>4</td>
<td>SendStreamDATA()</td>
<td>This function sends the speed and configuration data requested by the GUI running on the host PC. It always sends 3-Bytes of data. The first byte is ID and the subsequent two bytes are data.</td>
</tr>
<tr>
<td>5</td>
<td>Timer1_Config()</td>
<td>The Hard Timer 1 in MSS is configured to calculate the number of clock cycles between the one of the HALL signal events, this count is used for the motor actual speed calculation in RPM. Timer 1 is configured for one-shot mode that only generates an interrupt once when its down-counter reaches 0. It is explicitly reloaded to start decrementing again, whenever there is an interrupt from the HALL signal. The timer values are derived based on the minimum RPM specified by the user or the lower RPM is taken as 1. This specifies that if there is timer interrupt then it indicates that the motor is running below the minimum RPM specified by the user or motor is stopped.</td>
</tr>
<tr>
<td>6</td>
<td>Timer1_IRQHandler()</td>
<td>This interrupt service routine (ISR) is mapped to the timer expiry, which indicates that the motor is running below the min speed of the motor specified by the user or that the Motor is stopped.</td>
</tr>
<tr>
<td>7</td>
<td>GPIO1_IRQHandler()</td>
<td>This ISR is mapped to one of the HALL signals, In this function the current value of the time is read and the value is converted into the actual motor speed in RPM. The timer is reconfigured for the next Hall event to calculate the Motor speed in RPM at that time.</td>
</tr>
<tr>
<td>8</td>
<td>Angle_Estimation()</td>
<td>This implements the angle estimation logic using HALL and Encoder. The Encoder gives only relative position, but to commute the motor absolute position is required. The calculation of Absolute position using Hall and Encoder is explained in &quot;Angle Estimation Using Hall Sensor State and Quadrature Encoder&quot; on page 18.</td>
</tr>
<tr>
<td>9</td>
<td>ACE_PC0_Flag0_IRQHandler()</td>
<td>ISR is mapped to the ACE GP0 interrupt, which indicates completion of the current sampling. In this function the Motor currents are read from the ACE and translated to the original values. This interrupt occurs at PWM frequency, because the ACE is triggered from FPGA fabric for every PWM cycle to enable the ACE to sample the Motor phase currents. This also calls the FOC_Calculation() function.</td>
</tr>
</tbody>
</table>
Table 2-1 • The Functions Implemented in MSS (continued)

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| 10      | FOC_Calculation() | This function also includes 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 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. Then it calls the below functions:  
  • Angle_Estimation() for angle estimation using Hall and Encoder  
  • Clarke  
  • Park  
  • Speed PI controller  
  • Torque PI controller  
  • Flux PI controller  
  • Inverse Park  
  • Inverse Clarke  
  • SVPWM  
  The Angle is calculated using the function Angle_Estimation(). The same angle along with Phase currents is used in the Clarke and Park transformations to calculate $I_q$ and $I_d$. |
| 11      | PIController_Speed() | This is the speed PI controller, it calculates the speed error (Desired speed – actual speed) and it provides the $I_q$ reference value to the Torque PI controller. |
| 12      | Clarke()          | This function implements the Clarke transformations to generate the $I_a$ and $I_b$. |
| 13      | Park()            | This function implements the Park transformations to generate $I_q$ and $I_d$ from $I_a$, $I_b$ and angle. |
| 14      | PIController_Flux() | This is the $I_d$ current PI controller. The reference $I_d$, which is Zero, and the actual $I_d$ calculated based on the phase current measurements is fed as inputs to this PI controller. |
| 15      | PIController_Torque() | This is the $I_q$ current PI controller. The reference $I_q$, which is the output from the speed PI controller, and the actual $I_q$ calculated based on the phase current measurements is fed as inputs to this PI controller. |
| 16      | Inverse_Park()    | This implements the Park transformation to calculate the $V_a$ and $V_b$ from $V_q$, $V_d$ and angle. |
| 17      | Inverse_Clarke()  | This implements the Clarke transformation to calculate the $V_a$, $V_b$, and $V_c$ from $V_a$, and $V_b$. |
Table 2-1 • The Functions Implemented in MSS (continued)

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>Svpwm()</td>
<td>This function implements the svpwm to generate on time values for PWM A, PWM B, and PWM C using ( V_a ), ( V_b ), and ( V_c ). The on time values are programmed to FPGA fabric to generate corresponding PWMs to the inverter bridge.</td>
</tr>
<tr>
<td>19</td>
<td>Fabric_IRQHandler()</td>
<td>This is the ISR, mapped to the GPIO 7, which indicates the start of the current sampling. This occurs once in every PWM cycle. In this function, the ACE is restarted for current sampling.</td>
</tr>
</tbody>
</table>

Table 2-2 shows the macros defined in “bldc_foc.h” file used to configure the different parameters of the FOC algorithm.

Table 2-2 • The Macros Defined in “bldc_foc.h” File

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PWM_PERIOD_VAL</td>
<td>This value defines the PWM period. To set PWM frequency as 20 KHz (50 µs), the value should be 3750 (50 x (10^{-6}) x FPGA Frequency).</td>
</tr>
<tr>
<td>2</td>
<td>PWM_DEAD_TIME_VAL</td>
<td>This value defines the dead time for inverter bridge, the value is (\sim) 1 µs.</td>
</tr>
<tr>
<td>3</td>
<td>FAB_FEQ</td>
<td>This defines the frequency at which the FPGA logic is running.</td>
</tr>
<tr>
<td>4</td>
<td>PWM_FREQ_IN_US</td>
<td>This defines the PWM period in micro seconds.</td>
</tr>
<tr>
<td>5</td>
<td>PWM_COUNT</td>
<td>This defines the PWM count value, for 20 KHz (50 µs), the value should be 3750 (50 x (10^{-6}) x FPGA Frequency)</td>
</tr>
</tbody>
</table>

Table 2-3 shows the address mapping for different parameters in FPGA fabric.

Table 2-3 • The Address Mapping for Different Parameters in FPGA Fabric

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Register Name</th>
<th>Register Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PWM_PERIOD_REG_ADDR</td>
<td>0x40050A00UL</td>
<td>This defines the address of the PWM period register.</td>
</tr>
<tr>
<td>2</td>
<td>PWM_DEAD_TIME_REG_ADDR</td>
<td>0x40050A10UL</td>
<td>This defines the address of the Dead time register.</td>
</tr>
<tr>
<td>3</td>
<td>PWM_EN_REG_ADDR</td>
<td>0x40050A14UL</td>
<td>This defines the address of the PWM enable/disable register.</td>
</tr>
<tr>
<td>4</td>
<td>HALL_REG_ADDR</td>
<td>0x40050A18UL</td>
<td>This defines the address of the HALL pattern register.</td>
</tr>
<tr>
<td>5</td>
<td>ENCODER_MAX_CNT_REG_ADDR</td>
<td>0x40050A20UL</td>
<td>This defines the address of Maximum Count of pulses that should be counted by the Encoder.</td>
</tr>
<tr>
<td>6</td>
<td>ENCODER_POSITION_REG_ADDR</td>
<td>0x40050A24UL</td>
<td>This defines the address of the Current count of the Encoder.</td>
</tr>
</tbody>
</table>
Hardware Implementation

The logic implemented in the FPGA fabric runs on 75 MHz clock from the clock conditioning circuit (CCC) and does the functions of the FOC as mentioned below:

- PWM generation with dead time
- APB interface
- HALL synchronization and pattern detection
- Encoder Interface with filtering

The HW blocks in the Figure 2-3 on page 25 shows the program flow sequence of the blocks implemented in the FPGA fabric. After the reset from the MSS the Advanced Peripheral Bus (APB) slave implemented in FPGA fabric starts communicating with the MSS through the fabric interface controller (FIC). The APB slave decodes all the configuration data from the Cortex-M3 processor and assigns the data like PWM period, and dead time values to PWM Generation module. Once the PWM period and dead time are configured and PWM generation is enabled from Cortex-M3 processor, then the PWM count starts ticking and the count is passed to PWM compare match unit for every clock. The compare match unit generates 3 PWM signals. These three PWM signals are fed to the dead time controller. It generates 3 PWM signals to the high side of Inverter Bridge and 3 PWM signals to the low-side of Inverter Bridge.

The Encoder logic synchronizes the encoder pulses and filters out any noise, and also counts the number of pulses. Once the counter reaches the maximum that is configured by MSS then the counter rolls over to Zero and starts counting again. The Cortex-M3 can read number of Encoder pulses to derive the angle at any given time.

The HALL synchronization logic synchronizes the HALL signals and decodes the pattern, which can be consumed by the Cortex-M3 processor to calculate the angle. Table 2-4 shows the details of the modules implemented in FPGA fabric.

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Module Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>foc_top</td>
<td>This is the top-level module, which integrates all the sub modules.</td>
</tr>
<tr>
<td>2</td>
<td>apb_if.vhd</td>
<td>This implements the APB interface to communicate MSS with Fabric. It decodes all the commands from the MSS and programs the corresponding block.</td>
</tr>
<tr>
<td>3</td>
<td>pwm_gen.vhd</td>
<td>This takes the PWM ON time values from the Sine Commutation logic, compares against the PWM count, and generates the PWM signals.</td>
</tr>
<tr>
<td>4</td>
<td>pwm_count.vhd</td>
<td>This generates sync pulse for every PWM period and PWM current count to generate the PWM signals. The width of the PWM counter is 14 bit and it is an edge aligned.</td>
</tr>
<tr>
<td>5</td>
<td>pwm_comp.vhd</td>
<td>This generates the complementary signals for low-side switches.</td>
</tr>
<tr>
<td>6</td>
<td>hall_sync.vhd</td>
<td>The HALL synchronization logic synchronizes the HALL signals and decodes the pattern.</td>
</tr>
<tr>
<td>7</td>
<td>encoder.vhd</td>
<td>Synchronizes the encoder pulses and filters out any noise, and also counts the number of pulses. Once the counter reaches the maximum that is configured by MSS then the counter rolls over to Zero and starts counting again.</td>
</tr>
</tbody>
</table>
SVPWM Generation

The Svpwm() implements the SVPWM logic as mentioned in "Sinusoidal Voltage and SVPWM Generation" on page 12.

The $V_a$, $V_b$, and $V_c$ from the Inverse Clarke are modified using the below equation:

- Phase A PWM On time period = $V_a - \frac{(\text{Max}(V_a, V_b, V_c) + \text{Min}(V_a, V_b, V_c))}{2}$
- Phase B PWM On time period = $V_b - \frac{(\text{Max}(V_a, V_b, V_c) + \text{Min}(V_a, V_b, V_c))}{2}$
- Phase C PWM On time period = $V_c - \frac{(\text{Max}(V_a, V_b, V_c) + \text{Min}(V_a, V_b, V_c))}{2}$

EQ 2-1

Figure 2-4 shows the SVPWM of Phase A.

Figure 2-4 • The SVPWM of Phase A
Design Customizations

You can customize this solution to adapt the motors. All the Motor parameters, PI controller parameters and current offsets are programmable through the provided GUI. Figure 2-5 shows the parameters that can be configured through GUI.

![Image of GUI for FOC](image)

Figure 2-5 • The GUI for FOC

All the configurable parameters are set using the **Set** button and each one’s authenticity of configuration can be confirmed by clicking the **Get** button.

1. **Motor Configuration:**
   - Pole Pair: Defines the number of poles of the Motor Speed loop configurations.

2. **Speed Loop Configuration:**
   - \( K_p \): This is the proportional gain constant and this value is directly used in the speed PI controller without any scaling.
   - \( K_i \): This is the Integral gain constant and this value is directly used in the speed PI controller without any scaling.

3. **Torque loop configurations:**
   - \( K_p \): This is the proportional gain constant and this value is directly used in the Flux and Torque PI controllers without any scaling.
   - \( K_i \): This is the Integral gain constant and this value is directly used in the Flux and Torque PI controllers without any scaling.

4. **Speed configuration:** This specifies the desired speed of the motor.
5. **Phase Current offset configuration:** The motor phase currents are measured using the voltage across the shunt resistors connected to each phase. The phase currents are alternating with positive and negative voltage levels. The measured voltages are amplified and an offset voltage is added to maintain the voltage to the ADC at positive levels only. The Zero current voltage value is mapped to the 1.67 voltage that is mapped to the ADC direct input and the offset is used to determine the Zero current of the Motor. These parameters are provided to you as you can change these values if you are using your own board.
   - Phase A ADC offset: defines ADC input voltage corresponds to the Zero Current Phase A
   - Phase B ADC offset: defines ADC input voltage corresponds to the Zero Current Phase B
   - Phase C ADC offset: defines ADC input voltage corresponds to the Zero Current Phase C

6. **Speed Ramp Configuration:** You can specify the ramp value to ramp up or ramp down the speed. Some of the applications need ramp profile to avoid sudden changes in the current.

7. **Direction Control:** This specifies the direction as Clock Wise (CW) and Counter Clock Wise (CCW).

Besides these configurations, if you want to customize the design further, you should change the code. Below are some of the possible customizations:

1. **Changing PWM frequency:** This may be required in different aspects like reducing the motor noise and better control. The PWM generation is implemented in the FPGA fabric, but the PWM period is programmable through software by changing PWM_PERIOD_VAL, PWM_CNT_OFFSET, pwm_freq_in_us, and pwm_count macros. The maximum width of the PWM is 13 bit, and it is unsigned. To set the PWM frequency as 20 KHz (50 µs) the value of the PWM_PERIOD_VAL and pwm_count macros should be 3750 (50 x 10^(-6) x FPGA frequency). The value of PWM_CNT_OFFSET should be 1875. The value of pwm_freq_in_us should be 50.

2. **Changing the Dead time:** The Dead time control is implemented in the FPGA fabric, but the Dead time is programmable through software by changing PWM_DEAD_TIME_VAL macro. The maximum width is 13 bit, and it is unsigned. To set the Dead time as 1 µs, the value of the PWM_DEAD_TIME_VAL macro should be 75 (1 x10^(-6) x FPGA frequency).

3. **HALL pattern change for angle:** The HALL pattern is different for different motors and the angle corresponding to each motor is also different for different motors. To change the HALL pattern edit the HALL_ELE_ANGL array.

4. **PI Controller scaling changes:** The PI controller is tuned to the supplied motor. While the speed PI controller output is scaled to 12 bit, Torque and Flux PI controllers output is scaled to 16 Bit. This scaling is done based on the ADC currents and their scaling values in subsequent Clarke and Park transformations, keeping in mind that the values should be normalized to unity in Inverse Clarke transformation before deriving the ON time periods.

5. **Configuration changes:** All the possible configuration changes can be done using the macros defined in the bldc_foc.h.
Performance Details

Figure 2-5 shows the Software and FPGA performance details.

Table 2-5 • Blocks Implemented in FPGA Fabric and in MSS

<table>
<thead>
<tr>
<th>Blocks Implemented in FPGA Fabric (Operating frequency 75 MHz)</th>
<th>Components Used</th>
<th>Time for Execution</th>
<th>Resource Utilization</th>
<th>Utilization (%) for A2F200</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 PWM (Edge aligned) Generation with dead time control</td>
<td>13-bit Adder, Comparators</td>
<td>Continuous generation</td>
<td>751 out of 4608</td>
<td>(16.30%)</td>
</tr>
<tr>
<td>2 APB Interface</td>
<td>Decode logic</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Blocks Implemented in MSS</th>
<th>Components Used</th>
<th>Time for Execution&lt;sup&gt;1&lt;/sup&gt;</th>
<th>Time for Execution&lt;sup&gt;2&lt;/sup&gt;</th>
<th>Code Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ADC results conversion</td>
<td></td>
<td>~1.6 µs</td>
<td>~1.6 µs</td>
<td>~ 50 KB</td>
</tr>
<tr>
<td>2 FOC execution time&lt;sup&gt;3&lt;/sup&gt;</td>
<td></td>
<td>~5.4 µs</td>
<td>~6.8 µs</td>
<td></td>
</tr>
<tr>
<td>3 UI Interface through UART</td>
<td>Not required. Is done when the Cortex-M3 is free</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. When MSS runs at 100 MHz and code is executed from eSRAM.
2. When MSS runs at 75 MHz and code is executed from eSRAM.
3. FOC control loop is executed on every PWM cycle.
3 – 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 corresponding SmartFusion device accordingly.

Programming the Kit

1. Download the programming file as mentioned in the "Design and Programming Files" on page 47 and unzip it to C:\Microsemi.
2. Connect both USB cables to the evaluation board and verify that LEDs D11, D15, and LED1 turn on.
3. Launch FlashPro v10.0 (or later) Figure 3-1.

4. Click the New Project or click File > New Project.
5. In the New Project dialog box, type HALL_ENC_PROG in the Project Name field. Specify the Project Location as C:\Microsemi\SF_DMC_FOC_HALL_ENC_M.
6. Choose **Single device** for Programming mode as shown in Figure 3-2.

7. Click **OK**. The FlashPro GUI is displayed.
8. Click **Configure Device** and browse to the STAPL file
   
   C:\Microsemi\SF_DMC_FOC_HALL_ENC_M\A2F\DMC_FOC_HALL_ENC_M.stp

9. Click **Program** to write to the device.
10. Once the programming has been successful, the screen should look as shown in Figure 3-3.

![Figure 3-3 • End of Programming in FlashPro](image)

11. Unplug both USB cables from the SmartFusion Evaluation Kit Board.
Connecting the SmartFusion Evaluation/Development Kit with the Trinamic Kit

When you are 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.

**Note:** Switch off all power supplies while connecting/disconnecting the SmartFusion Development/Evaluation Kit Board from the TMCM-AC-840 Daughter Board. There is a small air gap remaining between the SmartFusion board and Trinamic’s daughter board as shown in Figure 3-4.

![Figure 3-4 • H3 Board Connector Interface to the SmartFusion Kit](image_url)

Connections for Programming the Kit

**Programming with the SmartFusion Development Kit**

1. Connect both the USB cables supplied with the kit to J9 and J15 LCPS interface via the LC programmer board.
2. Connect 5V power supply to J1.

**Programming with the SmartFusion Evaluation Kit**

Connect both USB cables supplied with the kit to the USB/UART interface and the USB program and debug interface.
Connecting the BLDC Motor with the Trinamic Kit

Switch off all power supplies when connecting or disconnecting any motor to/from the TMCM-AC-840 Daughter Board. Connect the 3 phases of BLDC Motor with the HALL and Encoder to the 3-pin connector H1 (UVW). Additionally, Hall Sensor signals can be connected to the 5-pin Hall signals connector H4 (+5 V, GND, H1, H2, H3). The Encoder Sensor signals can be connected to the 5-pin Encoder signals connector H5 (+5 V, GND, A, B, N, +5 V, GND can be combined with H4). Anaheim Motor (BLWS231D-40V-4000-360SI-01) with Encoder sensor can be used to run this design. See Figure 3-5 for an example.

Motor Wiring Details

By default the measured voltages across the shunt resistors, which represent motor phase currents, are connected to ABPS channels of the SmartFusion. ABPS channels voltage range is from -2.5 Volts to +2.5 Volts (with 12-Bits ADC resolution to represent complete range), but measured voltages, which represent motor phase currents are always positive.

Only 0 to +2.5 V range of the ABPS channels is used to represent the motor phase currents. The negative range is unused. If the measured voltages are connected to Direct ADC channels of the SmartFusion, which support 0 to + 2.5 V input range (with 12-Bits ADC resolution to represent complete range), then the resolution per ADC bit is increased compared to ABPS channels. This improves performance of the control loop. This can be done with small wiring on the board.

- Connect CUR1 Pin of the TMC daughter card Header K1 to the Pin ADC7 on the same header.
- Connect CUR2 Pin of the TMC daughter card Header K1 to the Pin AC1 on the same header.

Figure 3-5 • Connecting BLDC Motor with Trinamic Kit
The CUR1 and CUR2 are opposite to each other on the K1 header. And if you connect as mentioned above then these look like cross connections. Use the short wire to connect as much as possible. Figure 3-6 shows the connection details.

**Figure 3-6 • Motor Currents 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 48 V for VM603 (BLDC driver) as shown in Figure 3-7.

**Figure 3-7 • BLDC Driver Power Supply Connector**
Use DC external regulated power supply or power supply adaptor, which can give 40 V output. The power supply connection example is shown in Figure 3-8.

Figure 3-8 • Power Supply Connection
1. Identify your COM port for the USB to UART Bridge in Device Manager as shown in Figure A-1.

![Device Manager](image)

**Figure A-1 • Device Manager**

2. Download and install the Microsoft .NET Framework 4 client profile from the Microsoft website:
3. Figure A-2 is the GUI for the FOC of the BLDC Motor.

**Figure A-2 • GUI for the FOC of the BLDC Motor**

**Description of Options**

1. In the Serial Port Configuration, select the correct COM port as specified in the Device Manager and set the **Baud Rate** to 115200.

2. Click **Connect** to establish connection with the COM port.

   The GUI is programmed with the default configurations to run the Anaheim Motor (BLWS231D-40V-4000-360SI-01).

   **Motor Configuration:** The Pole Pair represents the number of poles of the motor, and for the default motor it is 4 poles. Refer to the *motor datasheet* for configuring the number of poles. Any wrong configuration of the number of poles affects the actual speed calculation and the closed loop operation.

   **Speed Loop Configurations:** $K_i$ and $K_p$ gain constants Configurations for Speed loop (Default $K_i=80$; $K_p=1500$).

   **Torque and Flux Loop Configurations:** $K_i$ and $K_p$ gain constants Configurations for Torque and Flux loop (Default $K_i=3$; $K_p=1500$).

   **Speed Configuration:** Typically the value should be between 200 and 5000 RPM. For default motor, the desired speed is set to 2000 rpm as default. You can always change and observe the motor performance.

   **Speed Ramp Configuration:** (Default value=500): The default value of the ramp up rate is 500 RPM/s. The maximum value should be less than the desired speed. Minimum value should be greater than zero. Higher the RAMP value, lesser is the time to reach the desired speed. Click either enable of disable to choose the ramp functionality.

   **Direction Control:** Clockwise (CW) or Counter-Clockwise (CCW) rotation.
The design can be customized or modified as per your requirements. Any changes in the HDL files should go through the Libero SoC flow. Any changes in the C file should follow the flow mentioned in Building Executable Image in Release Mode and Loading into eNVM tutorial for more information on building an application in release mode.

Conclusion

This user's guide describes the features of the SmartFusion cSoC FPGAs to develop an effective FOC 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 FOC design uses the inbuilt fabric, ACE, and MSS along with other peripherals. These features enable the solution for better integrity, low power, more reliability, and security.
You can download the design files from the Microsemi SoC Products Group website:
www.microsemi.com/soc/download/rsc/?f=SF_DMC_FOC_HALL_ENC_M_DF
The design zip file consists of Libero SoC and Softconsole projects and programming file (*.stp) for A2F200. Refer to the Readme.txt file included in the design file for directory structure and description.

You can download the programming files (*.stp) in release mode from the Microsemi SoC Products Group website:
www.microsemi.com/soc/download/rsc/?f=SF_DMC_FOC_HALL_ENC_M_PF
C – 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, 408.643.6913

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

Website

You can browse a variety of technical and non-technical information on the SoC home page, at 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_iter@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.
Index

A
ADC current sampling window 17

C
Clarke Transformation 11
Conclusion 41
Connection of BLDC Motor with Trinamic Kit 35
Connection of SmartFusion Evaluation 34
Connections for Programming the Kit 34
contacting Microsemi SoC Products Group
customer service 43
email 43
web-based technical support 43
customer service 43

D
Description of options 40
Design customizations 32
Design Description 23
Development Kit with Trinamic Kit 34

F
Field Oriented Control Theory 8

H
Hardware Implementation 30

I
Inverse Clarke transformation 12
Inverse Park Transformation 12

M
Microsemi SoC Products Group
e-mail 43
web-based technical support 43
website 43
Motor Wiring Details 35

P
Performance details 34
Phase current measurement using ADC 15
Power Supply Connection 36
product support
customer service 43
email 43
My Cases 44
outside the U.S. 44
technical support 43
website 43
Programming the Kit 31
Programming with SmartFusion Development Kit 34
Programming with SmartFusion Evaluation Kit 34
PWM generation 14

S
Sinusoidal Voltage and SVPWM Generation 12
Software Implementation 26
Speed Calculation using Hall State 21
SVPWM generation 31

tech support
ITAR 44
My Cases 44
outside the U.S. 44
technical support 43
Transformations 10

U
User Interface 39

W
web-based technical support 43