



CY8CKIT-037

PSoC[®] 4 Motor Control Evaluation Kit Guide

Doc. No. 001-92562 Rev. *C

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134 USA
www.cypress.com

© Cypress Semiconductor Corporation, 2015-2018. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.

Contents



Safety Information	5
1. Introduction	6
1.1 Kit Contents	6
1.2 Getting Started	7
1.3 Additional Learning Resources.....	8
1.4 Technical Support.....	9
1.5 Documentation Conventions	9
1.6 Acronyms	9
2. Software Installation	10
2.1 Before You Begin	10
2.2 Install Kit Software.....	10
2.3 Uninstall Software.....	13
3. Kit Operation	14
3.1 CY8CKIT-037 Evaluation Kit Overview	14
3.2 Kit Operation and Configuration Guide	15
3.2.1 DC Power Supply Connector	15
3.2.2 Motor Winding Connectors	15
3.2.3 Hall Sensors Interface Connector	16
3.2.4 Connectors to CY8CKIT-042 Board.....	16
3.2.5 USB Connector.....	17
4. Hardware	18
4.1 Block Diagram Overview	18
4.2 Input Protection Circuit	19
4.3 DC/DC Switching Regulator	19
4.4 MOSFET Dual H-Bridge and Dual H-Bridge PWM Drivers.....	20
4.5 Phase Current Detecting and Processing Circuit.....	21
4.6 Hall Sensors and BEMF Sensing Circuit	22
4.7 USB-to-UART Bridge Controller Circuit	23
4.8 Connectors	23
4.9 Test Points.....	24
5. Code Examples	25
5.1 Using the Kit Code Examples	25
5.2 Configuration Jumpers for Different Motor Types	28
5.3 Sensored BLDC Motor Control Code Example.....	28
5.3.1 Sensored BLDC Background.....	28
5.3.2 Sensored BLDC Motor Control Code Example Overview.....	29

- 5.3.3 Control Schematic Overview 30
- 5.3.4 Flow Chart 31
- 5.3.5 Running the Sensored BLDC Motor Control Code Example 32
- 5.3.6 Adapting the Code Example to another Motor 35
- 5.4 Sensorless BLDC Motor Control Code Example 36
 - 5.4.1 Sensorless BLDC Background 36
 - 5.4.2 Sensorless BLDC Motor Control Code Example Overview 36
 - 5.4.3 Control Schematic Overview 37
 - 5.4.4 Firmware Introduction 38
 - 5.4.5 Running the Sensorless BLDC Motor Control Code Example 39
 - 5.4.6 Adapting the Code Example to another Motor 42
- 5.5 Sensorless FOC Motor Control Code Example 49
 - 5.5.1 Sensorless FOC Background 49
 - 5.5.2 Sensorless FOC Motor Control Code Example Overview 55
 - 5.5.3 Control Schematic Overview 56
 - 5.5.4 Firmware Introduction 58
 - 5.5.5 Running the Sensorless FOC Motor Control Code Example 60
 - 5.5.6 Adapting the Code Example to another Motor 62
- 5.6 Single Shunt FOC Motor Control Code Example 63
 - 5.6.1 Single Shunt FOC Background 63
 - 5.6.2 Single Shunt FOC Motor Control Code Example Overview 65
 - 5.6.3 Control Schematic Overview 66
 - 5.6.4 Firmware Introduction 67
 - 5.6.5 Running the Single Shunt FOC Motor Control Code Example 68
 - 5.6.6 Adapting the Code Example to another Motor 69
- 5.7 Stepper Motor Control Code Example 70
 - 5.7.1 Stepper Motor Background 70
 - 5.7.2 Stepper Motor Control Code Example Overview 71
 - 5.7.3 Control Schematic Overview 72
 - 5.7.4 Firmware Introduction 73
 - 5.7.5 Running the Stepper Motor Control Code Example 74
 - 5.7.6 Adapting the Code Example to another Motor 77
- 5.8 Bridge Control Panel Monitor Tool Guide 78
 - 5.8.1 BCP Monitoring Overview 78
 - 5.8.2 Installing the Driver for CY7C65213 USB-to-UART Bridge Controller 78
 - 5.8.3 Upper Terminal Configuration Guide 80
 - 5.8.4 Lower Terminal Configuration Guide 84
 - 5.8.5 Reading the Motor Speed Using BCP Commands 85
- A.1. Board Schematics, Board Layout, and BOM 87
- A.2. Bill of Materials 96
- A.3. No Load Components 99
- A.4. Special Installation Instructions 100
- A.5. Label 100

Safety Information




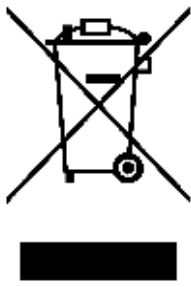
Regulatory Compliance

The CY8CKIT-037 PSoC® 4 Motor Control Evaluation Kit is intended for use as a motor control development platform for hardware or software in a laboratory environment. The board is an open-system design, which does not include a shielded enclosure. Therefore, the board may cause interference with other electrical or electronic devices in close proximity. In a domestic environment, this product may cause radio interference. In such cases, the user may be required to take adequate preventive measures. Also, this board should not be used near any medical equipment or RF devices.

Attaching additional wiring to this product or modifying the product operation from the factory default may affect its performance and cause interference with other apparatus in the immediate vicinity. If such interference is detected, suitable mitigating measures should be taken.

The CY8CKIT-037, as shipped from the factory, has been verified to meet with the requirements of CE as a Class A product.



	<p>The CY8CKIT-037 contains ESD-sensitive devices. Electrostatic charges readily accumulate on the human body and any equipment, and can discharge without detection. Permanent damage may occur to devices subjected to high-energy discharges. Proper ESD precautions are recommended to avoid performance degradation or loss of functionality. Store unused CY8CKIT-037 boards in the protective shipping package.</p>
	<p>End-of-Life/Product Recycling</p> <p>The end of life for this kit is five years from the date of manufacture mentioned on the back of the box. Contact your nearest recycler to discard the kit.</p>

General Safety Instructions

ESD Protection

ESD can damage boards and associated components. Cypress recommends that the user perform procedures only at an ESD workstation. If an ESD workstation is not available, use appropriate ESD protection by wearing an antistatic wrist strap attached to the chassis ground (any unpainted metal surface) on the board when handling parts.

Handling Boards

CY8CKIT-037 boards are sensitive to ESD. Hold the board only by its edges. After removing the board from its box, place it on a grounded, static-free surface. Use a conductive foam pad if available. Do not slide the board over any surface.

1. Introduction



Thanks for your interest in the CY8CKIT-037 PSoC[®] 4 Motor Control Evaluation Kit (EVK). This kit enables engineers to evaluate Cypress' PSoC 4 family of devices for motor control applications. Based on this kit, customers can create control solutions for three major motor types: Permanent Magnet Synchronous Motor (PMSM), stepper, and Brushless DC (BLDC).

The Cypress PSoC family of devices integrates abundant internal resources for motor control applications, such as Timer Counter Pulse Width Modulator (TCPWM), SAR ADC, comparators, opamps, and universal digital blocks (UDBs). In addition, an Arm[®] Cortex[®] core enables high-performance motor control solutions on PSoC devices. Headers provided on the EVK board allow you to connect it to the [CY8CKIT-042 PSoC 4 Pioneer Kit](#) board to create a complete motor control system.

This kit guide provides the circuit structure of the Motor Control EVK board and explains how to configure it to create a solution for different motor types. It provides five code examples that cover sensed and sensorless BLDC control, PMSM sensorless field-oriented control (FOC), and stepper motor microstepping control. It also introduces the Bridge Control Panel (BCP) as a debugging tool in the motor control development process.

1.1 Kit Contents

The CY8CKIT-037 PSoC 4 Motor Control EVK includes the following:

- CY8CKIT-037 Motor Control Evaluation Board
- 24-V/2.1-A AC-DC Adapter
- BLDC motor (BLY172S-24V-4000) with sinusoidal back electromotive force
- USB Standard-A to Mini-B cable
- Configuration Jumpers
- Fuse
- Screwdriver
- Quick Start Guide

Figure 1-1. CY8CKIT-037 PSoC 4 Motor Control EVK Contents



Inspect the contents of the kit. If you find any part missing, contact your nearest Cypress sales office for assistance: www.cypress.com/support.

1.2 Getting Started

To learn the solution quickly and apply it to your design, refer to the CY8CKIT-037 PSoC 4 Motor Control Evaluation Kit Quick Start Guide inside the kit box or in the installation directory. The default location for the kit documents is:

```
<Install_Directory>\CY8CKIT-037 Motor Control EVK\<version>\Documentation
```

This guide will help you get acquainted with the CY8CKIT-037 PSoC 4 Motor Control EVK:

- The [Introduction](#) chapter describes the basic information of the CY8CKIT-037 PSoC 4 Motor Control Evaluation Kit.
- The [Software Installation](#) chapter describes the installation of the kit software.
- The [Kit Operation](#) chapter describes the major features of the CY8CKIT-037 PSoC 4 Motor Control Evaluation Kit.
- The [Hardware](#) chapter describes the hardware content of the CY8CKIT-037 PSoC 4 Motor Control Evaluation Kit and the hardware operation.
- The [Code Examples](#) chapter describes the multiple projects that will help you understand how to evaluate different supported output standards on this kit.
- The [Appendix](#) captures DC/AC Measurements of Clock Outputs, Schematics, Fab Drawing, and the bill of materials (BOM).

1.3 Additional Learning Resources

Cypress provides a wealth of data at www.cypress.com to help you select the right PSoC device for your design, and to help you to quickly and effectively integrate the device into your design. For a comprehensive list of resources, see [KBA86521, How to Design with PSoC 3, PSoC 4, and PSoC 5LP](#). The following is an abbreviated list for PSoC 4:

- Overview: [PSoC Portfolio](#) and [PSoC Roadmap](#).
- Product Selectors: [PSoC 4 Product Selector](#). In addition, [PSoC Creator](#) includes a device selection tool.
- Datasheets: Describe and provide electrical specifications for the [PSoC 4000](#), [PSoC 4100](#), [PSoC 4200](#), [PSoC 4100M](#), [PSoC 4200M](#), and [PSoC 4200L](#) device families.
- [CapSense Design Guide](#): Learn how to design capacitive touch-sensing applications with the PSoC 4 family of devices.
- [Application Notes](#) and [Code Examples](#): Cover a broad range of topics, from basic to advanced. Many of the application notes include code examples. Visit the [PSoC 3/4/5 Code Examples webpage](#) for a list of all available PSoC Creator code examples.
- [Technical Reference Manuals \(TRM\)](#): Provide detailed descriptions of the architecture and registers in each PSoC 4 device family.
- Development Kits:
 - [CY8CKIT-046](#), [CY8CKIT-044](#), [CY8CKIT-042](#), and [CY8CKIT-040](#) are easy-to-use and inexpensive development platforms. These kits include connectors for Arduino-compatible shields and Digilent Pmod peripheral modules.
 - [CY8CKIT-049](#) and [CY8CKIT-043](#) are low-cost prototyping platforms for sampling PSoC 4 devices.
 - [MiniProg3](#) device provides an interface for flash programming and debug.
- [Knowledge Base Articles \(KBA\)](#): Provide design and application tips from experts on using the device.
- PSoC Creator Training: Visit www.cypress.com/go/creatorstart/creatortraining for a comprehensive list of video trainings on PSoC Creator.

Learning from Peers: Visit www.cypress.com/forums to meet enthusiastic PSoC developers discussing the next generation embedded systems on Cypress Developer Community Forums

1.4 Technical Support

For assistance, go to www.cypress.com/support, or contact our customer support at +1(800) 541-4736 Ext. 3 (in the USA), or +1 (408) 943-2600 Ext. 3 (International).

1.5 Documentation Conventions

Table 1-1. Document Conventions for Guides

Convention	Usage
Courier New	Displays file locations, user-entered text, and source code: C:\...cd\icc\
<i>Italics</i>	Displays file names and reference documentation: Read about the <i>sourcefile.hex</i> file in the <i>PSoC Designer User Guide</i> .
[Bracketed, Bold]	Displays keyboard commands in procedures: [Enter] or [Ctrl] [C]
File > Open	Represents menu paths: File > Open > New Project
Bold	Displays commands, menu paths, and icon names in procedures: Click the File icon and then click Open .
Times New Roman	Displays an equation: $2 + 2 = 4$
Text in gray boxes	Describes cautions or unique functionality of the product.

1.6 Acronyms

Table 1-2. List of Acronyms used in this Document

Acronym	Definition
BCP	Bridge Control Panel
BEMF	Back Electromotive Force
BLDC	Brushless DC
BOM	Bill of Materials
EVK	Evaluation Kit
FOC	Field-Oriented Control
IDE	Integrated Design Environment
IPMSM	Interior Permanent Magnet Synchronous Motor
ISR	Interrupt Service Routine
PMSM	Permanent Magnet Synchronous Motor
SPMSM	Surface Permanent Magnet Synchronous Motor
SVPWM	Space Vector Pulse Width Modulation

2. Software Installation



This chapter describes the steps to install the software tools and packages on a PC for using the CY8CKIT-037 PSoC 4 Motor Control EVK.

2.1 Before You Begin

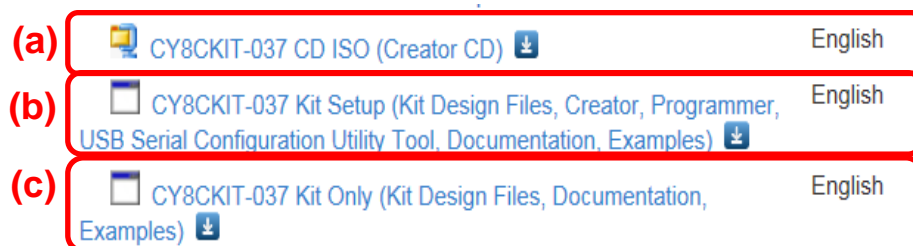
All Cypress software installations require administrator privileges, but these are not required to run the software after it is installed. Close any other Cypress software that is currently running before installing the kit software.

2.2 Install Kit Software

Follow these steps to install the CY8CKIT-037 PSoC 4 Motor Control EVK software:

1. Download the CY8CKIT-037 EVK software from www.cypress.com/CY8CKIT-037. The CY8CKIT-037 software is available in the following formats for download (see Figure 2-1):
 - a. **CY8CKIT-037 DVD ISO:** This file is a complete package, stored in a CD/DVD-ROM image format that you can use to create a CD/DVD or extract using an ISO extraction program such as WinZip or WinRAR. The file can also be mounted similar to a virtual CD/DVD using virtual drive programs such as Virtual CloneDrive and MagicISO. This file includes all the required software, utilities, drivers, hardware files, and user documents.
 - b. **CY8CKIT-037 Evaluation Kit Complete Setup:** This installation package contains the files related to the CY8CKIT-037 Evaluation Kit. However, it does not include the Windows Installer or Microsoft .NET Framework packages. If these packages are not available on your computer, the installer directs you to download and install them from the Internet.
 - c. **CY8CKIT-037 Evaluation Kit Only:** This executable file installs only the CY8CKIT-037 EVK contents, which include code examples, hardware files, and user documents. This package can be used if all the software prerequisites (listed in step 4) are installed on your PC.

Figure 2-1. Available Formats for Downloading EVK Software



2. If you have downloaded the ISO file, mount it on a virtual drive. If you do not have a virtual drive to mount, extract the ISO contents using the appropriate ISO extractor (such as MagicISO or PowerISO). Double-click *cyautorun.exe* in the root directory of the extracted content or the mounted ISO if the “Autorun from CD/DVD” option is not enabled on the PC. The installation window will appear automatically (see Figure 2-2).

Note: If you are using the “CY8CKIT-037 Kit Setup” or “CY8CKIT-037 Kit Only” file, then go to step 3 for installation.

3. Click **Install CY8CKIT-037 EVK** to start the kit installation, as shown in [Figure 2-2](#).

Figure 2-2. Kit Installer Startup Screen



4. Select the folder where you want to install the files related to the CY8CKIT-037 PSoC 4 Motor Control EVK. Choose the directory and click **Next**.

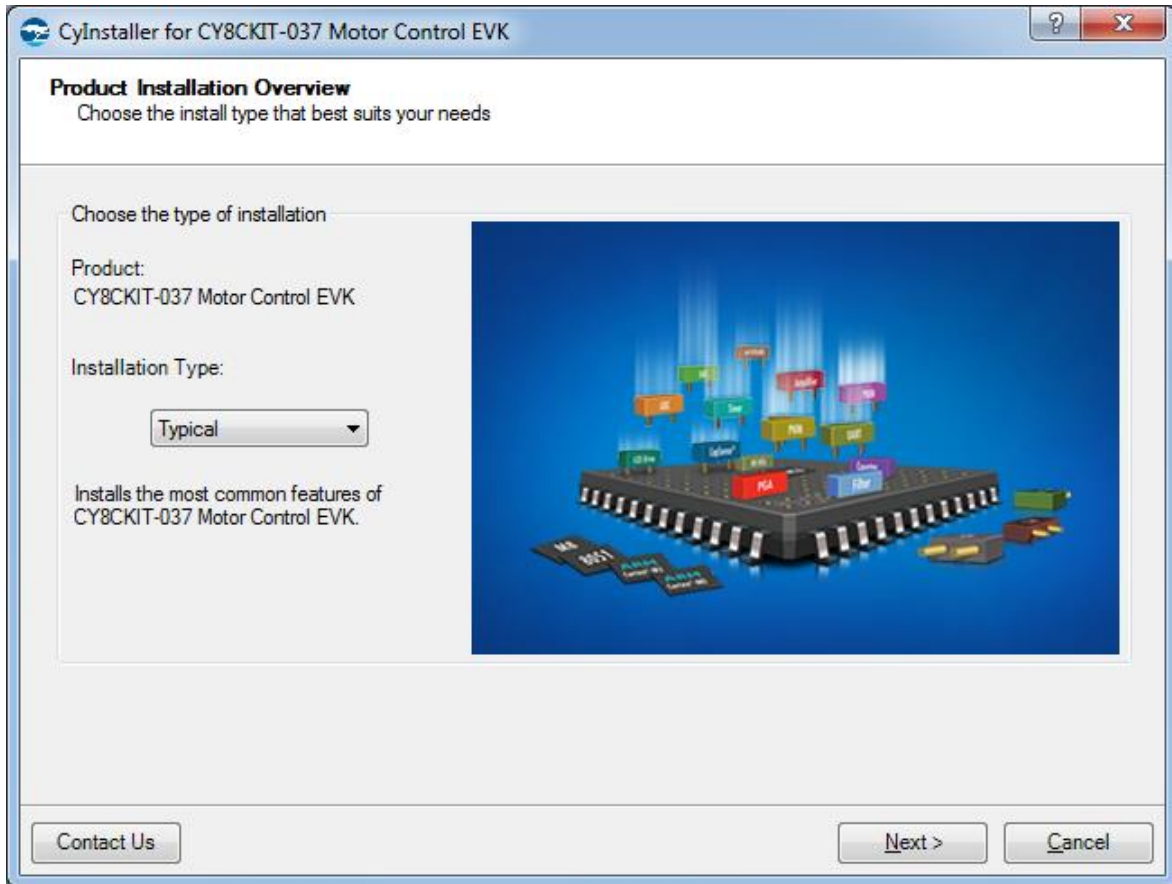
Note: When you click the **Next** button, the CY8CKIT-037 installer automatically installs the required software, if it is not present on your computer.

Following are the required software:

- a. PSoC Creator 4.2 or later: This software is also available at www.cypress.com/psoccreator.
- b. PSoC Programmer 3.27.1 or later: This is installed as part of PSoC Creator installation (www.cypress.com/programmer).

- Choose the **Typical**, **Custom**, or **Complete** installation type (select **'Typical'** if you do not know which one to select) in the Product Installation Overview window, as shown in [Figure 2-3](#). Click **Next** after you select the installation type.

Figure 2-3. Product Installation Overview Window



- Read the License agreement and select **I accept the terms in the license agreement** to continue with the installation. Click **Next**.
- When the installation begins, a list of packages appears on the installation page. A green check mark appears next to each package after successful installation.
- Enter your contact information or select the option **Continue without Contact Information**. Click **Finish** to complete the CY8CKIT-037 kit installation.
- After the installation is complete, the kit contents are available at the following location:
<Install_Directory>\CY8CKIT-037 Motor Control EVK.

Default location:

- Windows OS (64-bit):
C:\Program Files(x86)\Cypress\CY8CKIT-037 Motor Control EVK
- Windows OS (32-bit):
C:\Program Files\Cypress\CY8CKIT-037 Motor Control EVK

Note: For Windows 7/8/8.1/10 users, the installed files and the folder are read-only.

To use the installed code examples, follow the steps outlined in [Using the Kit Code Examples](#) section. These steps will create an editable copy of the example in a path that you choose so the original installed example is not modified.

2.3 Uninstall Software

The software can be uninstalled using one of the following methods:

- Go to **Start > Control Panel > Programs and Features** for Windows 7 or **Add/Remove Programs** for Windows XP and select the **Uninstall** button.
- Go to **Start > All Programs > Cypress > Cypress Update Manager** and select the **Uninstall** button next to the product that needs to be uninstalled.

3. Kit Operation

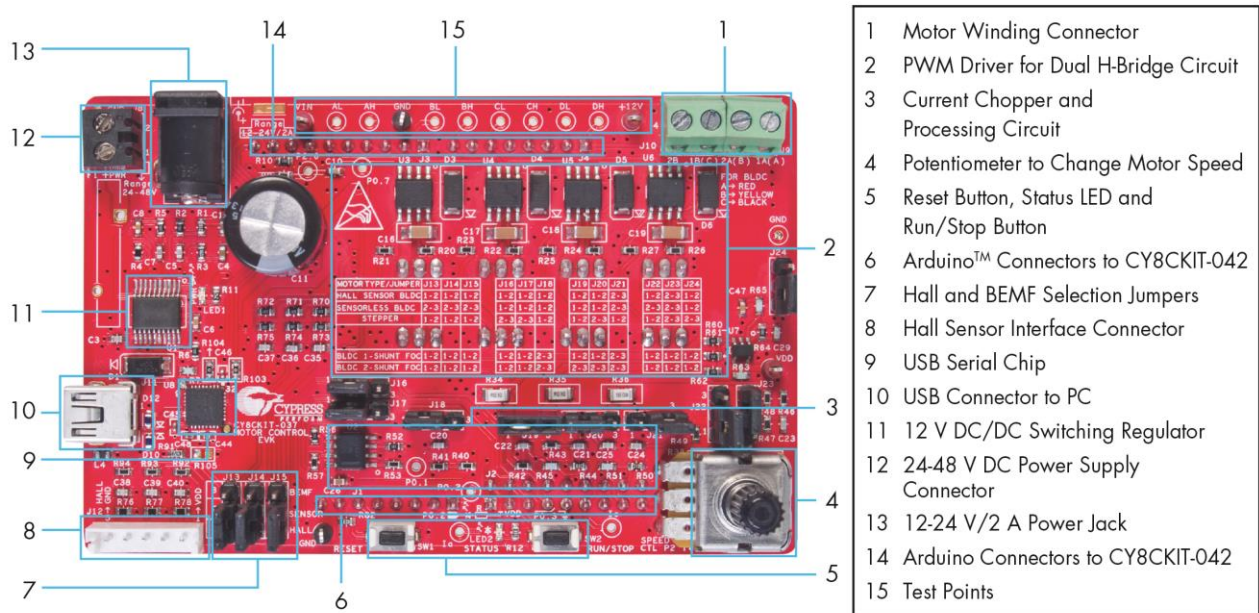


This chapter introduces you to the features of the CY8CKIT-037 EVK. It describes the connectors of the board and the usage of these connectors.

3.1 CY8CKIT-037 Evaluation Kit Overview

The motor control system can be separated into two parts: the driver board and the controller board. The CY8CKIT-037 Motor Control EVK is the driver board, which contains the DC/DC power circuit, dual H-bridge circuit, motor current and bus voltage sampling and processing circuit, protection circuit, user configuration circuit, and connectors to the controller board. The controller board receives the signals, implements the proper algorithm to process them, and then generates control signals to the driver board to run the motor. Figure 3-1 shows the EVK board and its general description. CY8CKIT-037 EVK is the driver board; the CY8CKIT-042 kit works as the controller board. They are interfaced with Arduino-compatible connectors.

Figure 3-1. General Description of EVK Board



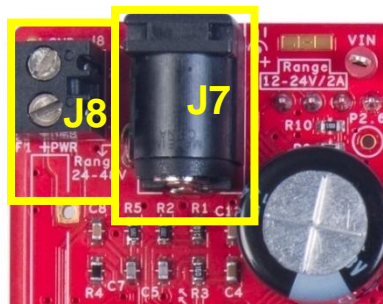
3.2 Kit Operation and Configuration Guide

3.2.1 DC Power Supply Connector

There are two connectors for DC power input, J7 and J8, as shown in [Figure 3-2](#). J7 is a standard power jack for DC adapters with 12 V or 24 V DC output. J8 enables the kit to get input power from a general variable DC power supply with 24 V to 48 V DC output voltage. The input current of both connectors can be up to 2 A. For schematic details, see [Input Protection Circuit](#). CY8CKIT-042 gets 12 V from the DC/DC converter on CY8CKIT-037. The LDO on CY8CKIT-042 generates the 3.3 V supply for the PSoC device if the VDD on CY8CKIT-042 is configured as 3.3 V. If the VDD on CY8CKIT-042 is configured as 5 V, CY8CKIT-042 needs 5 V from the USB cable.

CAUTION Do not connect power to both J7 and J8 simultaneously. Also, when using J7, connect the power cable to it before loading the AC supply at the AC/DC adapter to avoid a potential spark.

Figure 3-2. Connectors for DC Power Input



3.2.2 Motor Winding Connectors

J9 and J10 provide four pins for motor windings, as shown in [Figure 3-3](#). The BLDC and PMSM motors use three pins, while the stepper motor application needs all four pins.

Connection details are as follows:

For BLDC and PMSM:

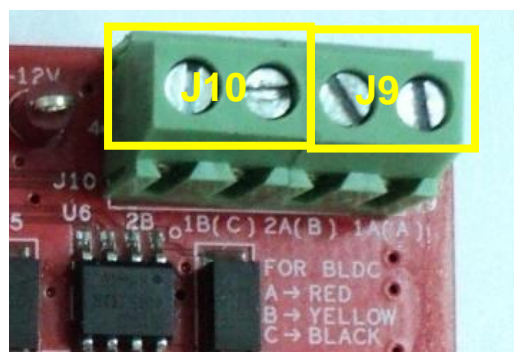
- Pin 1–1A (A) → Motor winding A – Red
- Pin 2–2A (B) → Motor winding B – Yellow
- Pin 3–1B (C) → Motor winding C – Black

For stepper motor:

- Pin 1–1A (A) → Motor winding 1 terminal A – Blue
- Pin 2–2A (B) → Motor winding 1 terminal B – Yellow
- Pin 3–1B (C) → Motor winding 2 terminal A – Red
- Pin 4–2B → Motor winding 2 terminal B – Green

For schematic details, see [Connectors](#).

Figure 3-3. Connectors to Motor Windings



3.2.3 Hall Sensors Interface Connector

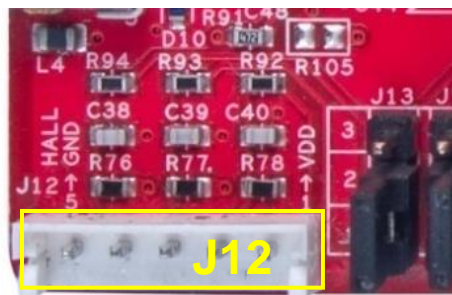
J12 is the Hall Sensors Interface Connector for the Sensored BLDC Motor Control code example, as [Figure 3-4](#) shows. It is a five-pin connector.

The details of each pin are as follows:

- VDD: Hall sensor board VDD – Red
- HALLA: Hall sensor A output – Blue
- HALLB: Hall sensor B output – Green
- HALLC: Hall sensor C output – White
- GND: Hall sensor board ground – Black

For schematic details, refer to [Hall Sensors and BEMF Sensing Circuit](#).

Figure 3-4. Hall Sensors Connector

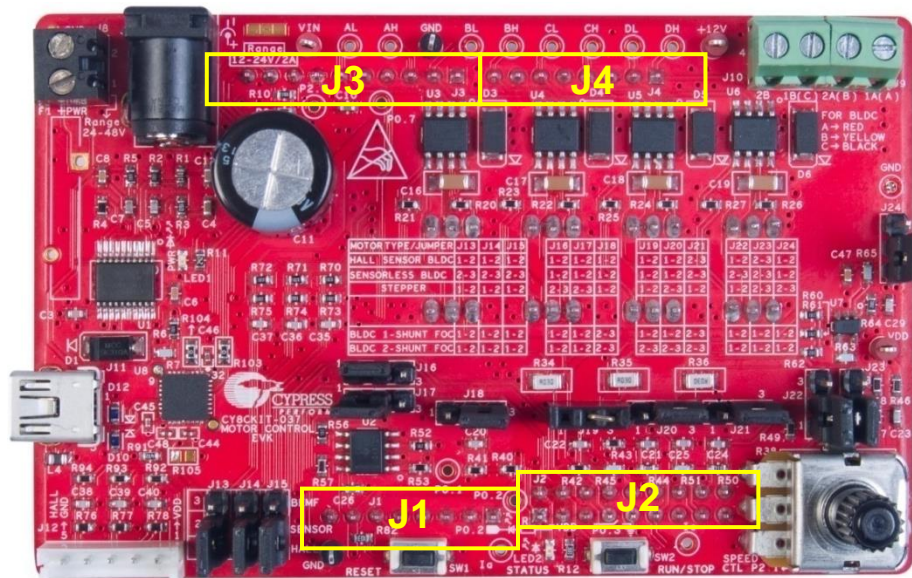


3.2.4 Connectors to CY8CKIT-042 Board

[Figure 3-5](#) shows the Arduino-compatible connectors to the CY8CKIT-042 board: J1, J2, J3, and J4. You can plug the EVK board into the CY8CKIT-042 board through these connectors. For schematic details, refer to [Connectors](#).

Note: Pin connectors J1, J2, J3, and J4 may bend when plugging the EVK board into CY8CKIT-042.

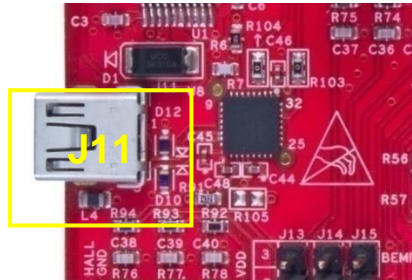
Figure 3-5. Connectors to CY8CKIT-042 Board



3.2.5 USB Connector

To monitor the real-time parameters while the motor is running, you can use the Bridge Control Panel (BCP) (refer to [Bridge Control Panel Monitor Tool Guide](#)) to get data through the USB-to-UART bridge circuit on the EVK board during the debugging process. The USB-to-UART bridge circuit on the EVK board can be connected to the PC using a USB Cable connected at the J11 connector, as shown in [Figure 3-6](#). For schematic details of the USB-to-UART bridge circuit part, refer to [USB-to-UART Bridge Controller Circuit](#).

Figure 3-6. USB-to-UART Bridge Controller Circuit



4. Hardware

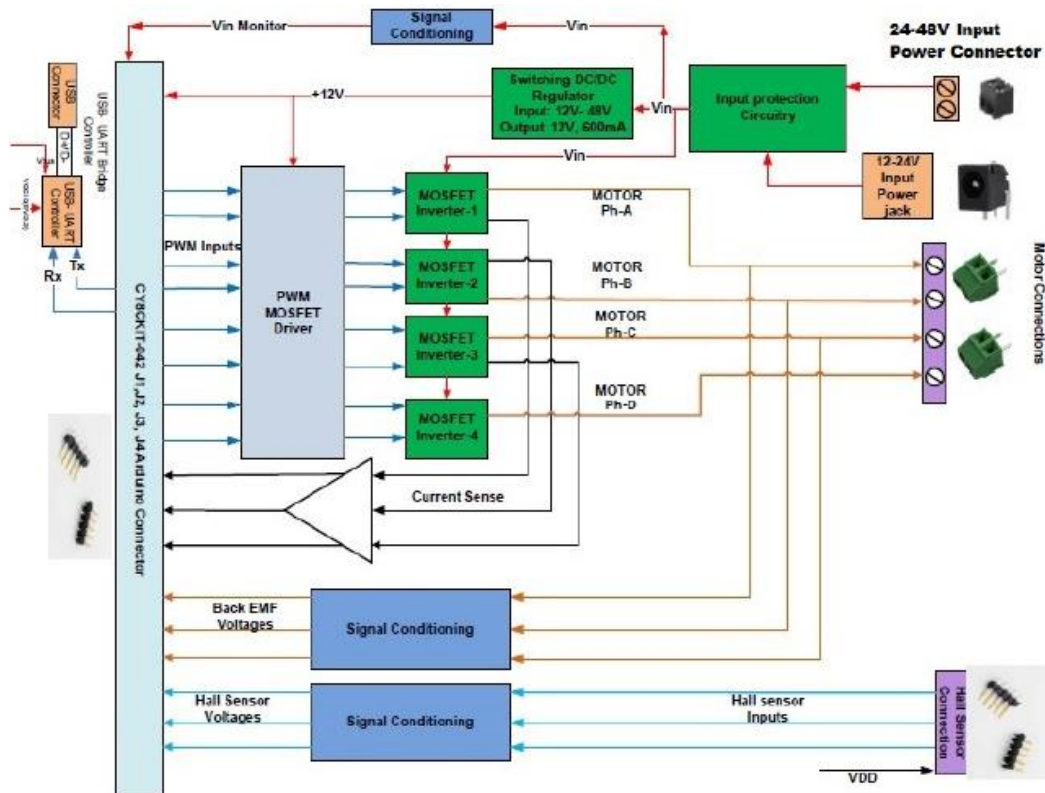


4.1 Block Diagram Overview

Figure 4-1 illustrates the CY8CKIT-037 hardware block diagram. The major features are as follows:

- Input protection circuit
- DC/DC switching regulator
- MOSFET dual H-bridge and dual H-bridge PWM drivers
- Phase current detecting and processing circuit
- Hall and back electromagnetic force (BEMF) sensing circuit
- USB-to-UART bridge controller circuit
- Connectors
- Test points and LEDs

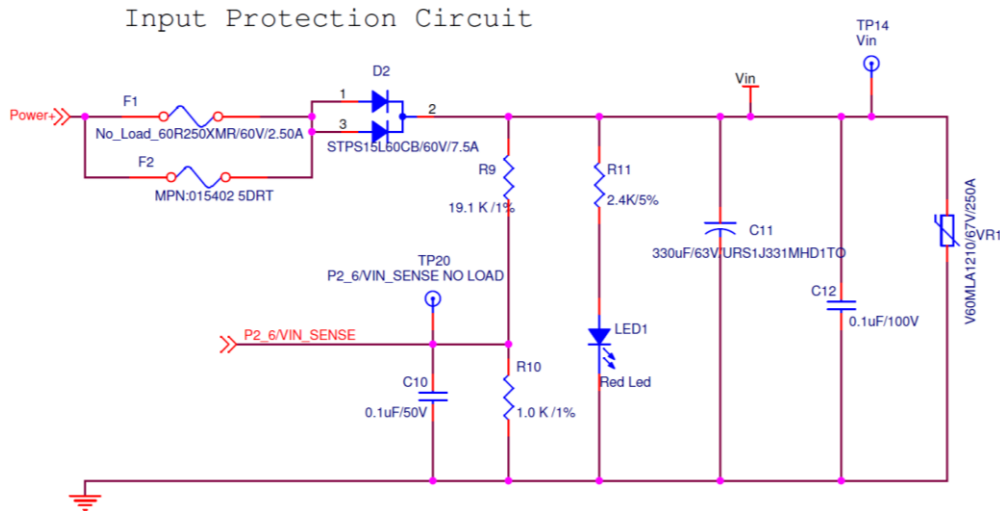
Figure 4-1. CY8CKIT-037 Hardware Block Diagram



4.2 Input Protection Circuit

Figure 4-2 shows the input protection circuit that consists of three parts: overcurrent protection by fuse F1 or F2, power supply reverse protection by Schottky diode D2, and input voltage inrush protection by varistor VR1. Notice that there are two parallel fuses: F1 is a PTC resettable fuse, and F2 is a common thermal fuse. F2 is populated onboard by default. You can also use PTC fuse F1 to replace F2 according to your requirements.

Figure 4-2. Input Protection Circuit



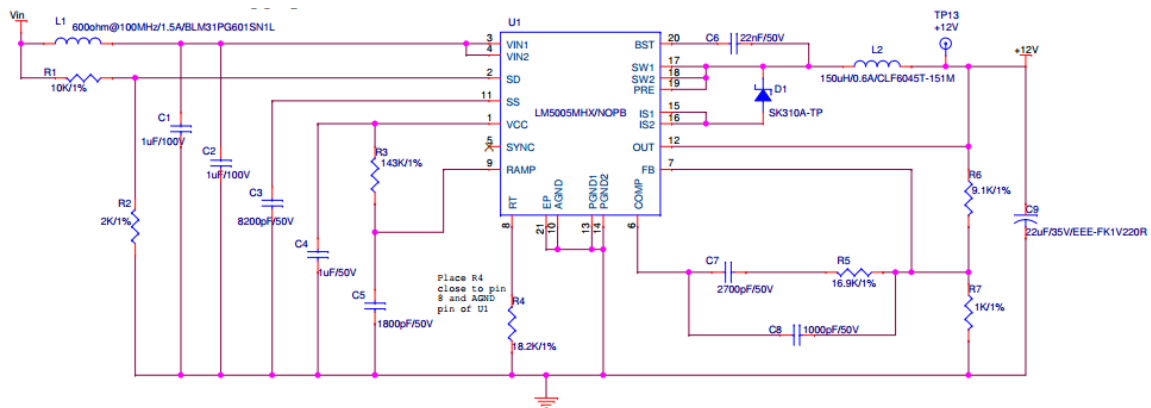
CAUTION

Do not populate both F1 and F2 on the EVK board. Otherwise, when an overcurrent condition occurs, F1 and F2 cannot cut off the circuit, which will lead to component damage on the board.

4.3 DC/DC Switching Regulator

Figure 4-3 shows the DC/DC switching regulator circuit on the EVK board. It converts the voltage V_{in} (output of the input protection circuit, 0.5 V to 1 V lower than the input voltage) to +12 V. This switching converter is a buck regulator using LM5005. LM5005 has an internal 75 V N-channel buck switch with an output current capability of 2.5 A. It is designed to take an input in the range of 7 V–75 V and provides 12 V output with 1.5 A current capability when the input voltage is higher than 13 V.

Figure 4-3. Switching DC/DC Converter



When the voltage at the SD pin of LM5005 is less than 1.225 V, the IC will go into an inactive state. The external R1 and R2 divider with 10 kΩ and 2 kΩ will give exactly 1.225 V at 7.35 V input. If the input goes below 7.35 V, LM5005 will be in an inactive state.

The +12 V output of the regulator provides power to the dual H-bridge driver chips and the PSoC 4 Pioneer Kit. The power supply in the range of 2.7 V to 5.5 V, needed by the other circuits and chips on the EVK board is provided from the controller board through the Arduino headers.

4.4 MOSFET Dual H-Bridge and Dual H-Bridge PWM Drivers

This part (see Figure 4-4) contains four half-bridge circuits and four driver circuits. For BLDC and PMSM control, you can use three half-bridge circuits to form a standard three-phase inverter. For the stepper motor, you can control two windings of the stepper motor separately via the dual H-bridge.

Figure 4-4 shows the driver circuit, containing four ASIC driver chips (IR2101) and relative peripheral components. IR2101 can source 100 mA or sink 210 mA current to the MOSFETs. Each chip can drive two MOSFETs of a half-bridge, and its high-side PWM operates with a floating supply by a peripheral bootstrap circuit.

Figure 4-4. Dual H-Bridge Driver Circuit

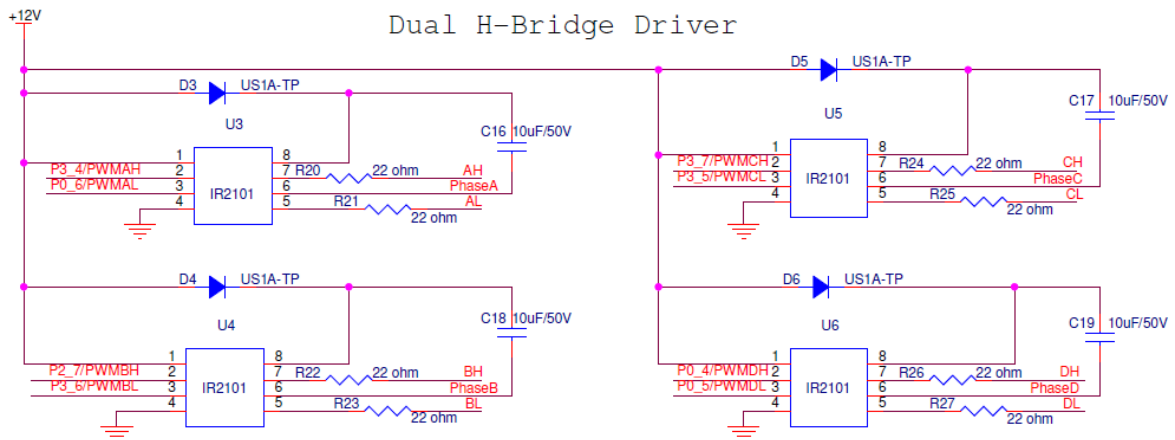
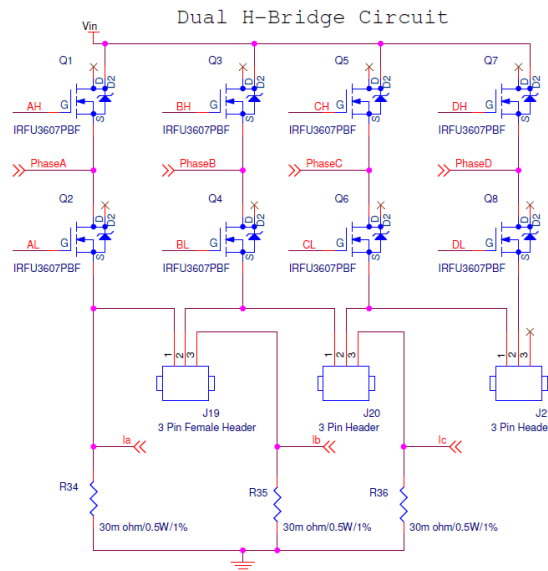


Figure 4-5 shows the MOSFET dual H-bridge. There are four half bridges in total, each one serving a motor phase or winding. Considering the compatibility for different motor control types, the population of components J19, J20, and J21 needs to be adjusted to fit different motor type applications.

Figure 4-5. MOSFET Dual H-Bridge



4.5 Phase Current Detecting and Processing Circuit

As Figure 4-5 shows, three sensing resistors (R34, R35 and R36) are inserted into three half-bridges. When the motor is running, the winding current floats through these resistors. So, detecting the voltage on the sensing resistors can get the phase current of the motor windings. Because of the sensing resistors' low value and the noise of the running motor, the voltage signal needs to be amplified. Figure 4-6 shows the current detecting and processing circuit for winding A and winding C.

The core of this circuit is the amplifier. The internal opamps of the PSoC 4 on the controller board are used. J18 and J23 are used only for PMSM FOC control because in the FOC application, the winding current can be both positive and negative. J18 and J23 add 1/2 VDD bias voltage to the sensing signal to make it always positive.

J22 is used only for the sensorless BLDC application, because the sensorless BLDC firmware uses an inside low-power comparator (LPComp). P1[1] is dedicated to the negative terminal of the comparator, which needs to be connected to the Vin divider circuit (P2[6]/Vin). R49 will change the divider proportion. Pins 2 and pin 3 of J22 need to be shorted to remove R49 from the circuit in the Sensorless BLDC Motor Control code example.

Figure 4-6. Current Detecting and Processing Circuit

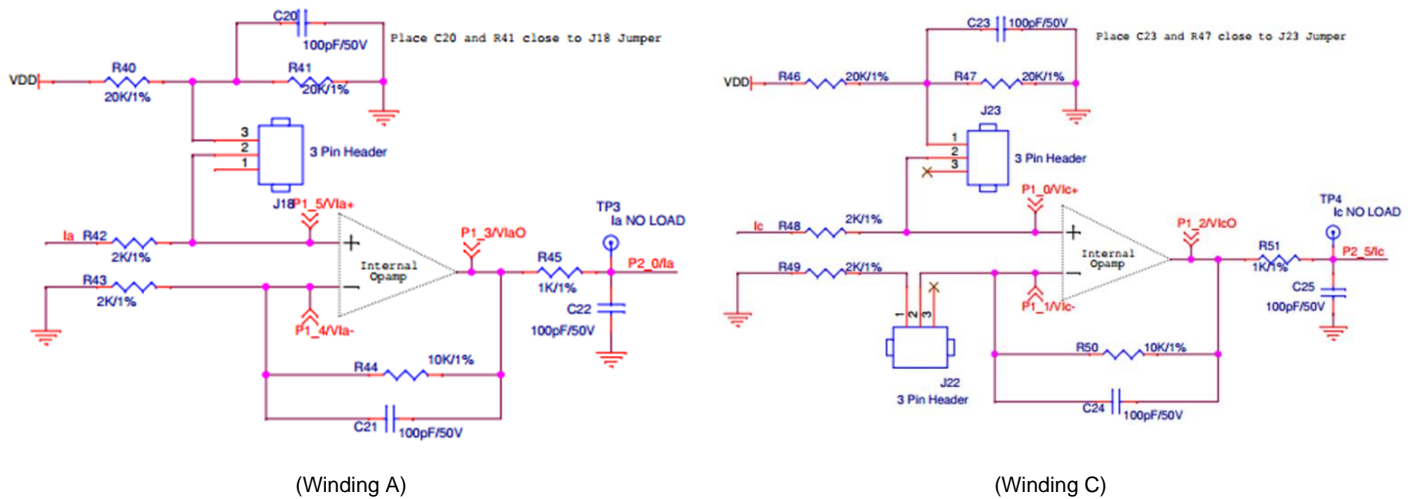


Figure 4-7 shows the external two-phase winding current comparing circuit, which is used only in the stepper motor application. The positive terminal input is the current sensing signal after amplifying and filtering, and the negative terminal input is the current reference set by the internal IDAC. The comparator output is routed to the PWM kill terminal to shut down the PWM output, forcing the winding current to follow the IDAC current reference.

Figure 4-7. Two-Phase Winding Current Comparing Circuit

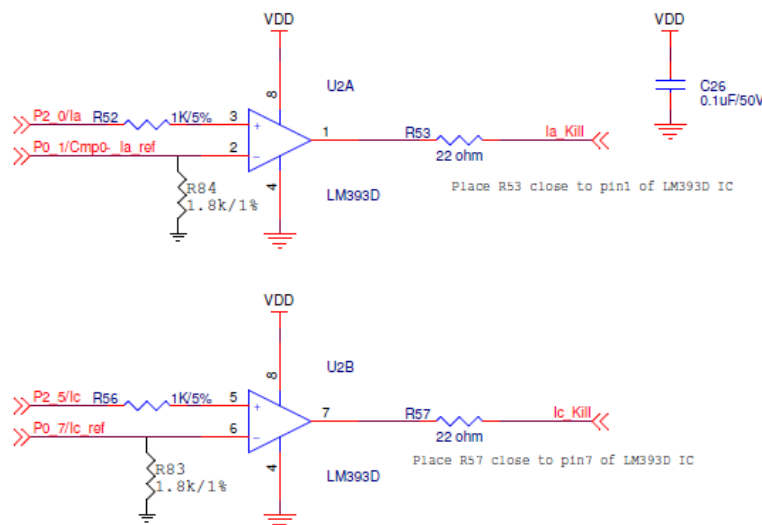
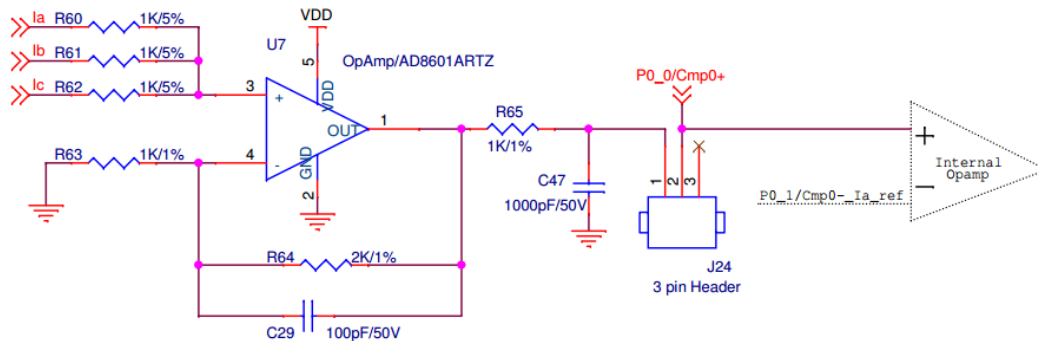


Figure 4-8 shows the overcurrent protection circuit. R60, R61, and R62 add the three-phase winding current, and the external opamp U7 amplifies and filters the sum of the currents. Its output is routed to the positive terminal of the internal comparator to be compared with the overcurrent threshold set by the IDAC.

Figure 4-8. Overcurrent Protection Circuit



4.6 Hall Sensors and BEMF Sensing Circuit

Both sensed and sensorless BLDC motors can be rotated with the EVK board. The sensed BLDC gets its rotor position from the Hall sensors, and the sensorless BLDC gets its rotor position by detecting the BEMF signal from the stator winding. Figure 4-9 shows the Hall sensor interface. The Hall sensor output is in an open-collector structure, so the interface adds pull-up resistors to make the PSoC device receive the correct Hall sensor signal.

Figure 4-9. Hall Sensor Interface

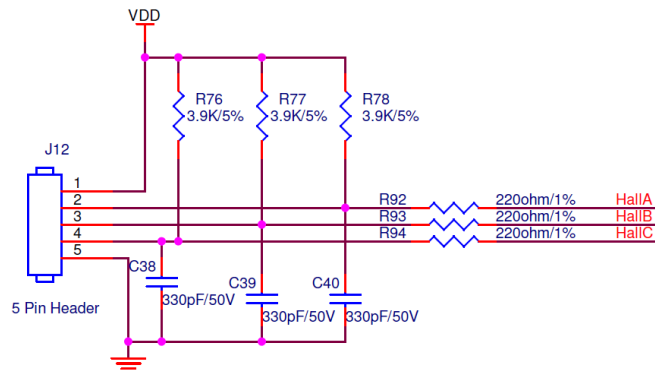


Figure 4-10 shows the voltage divider and filter circuit for BEMF detection. Three phases of the Hall sensor and BEMF signals are routed to jumpers J13, J14, and J15, as shown in Figure 4-11. Users can select sensed or sensorless configuration via the three jumpers.

Figure 4-10. Sensorless BLDC BEMF Detecting Circuit

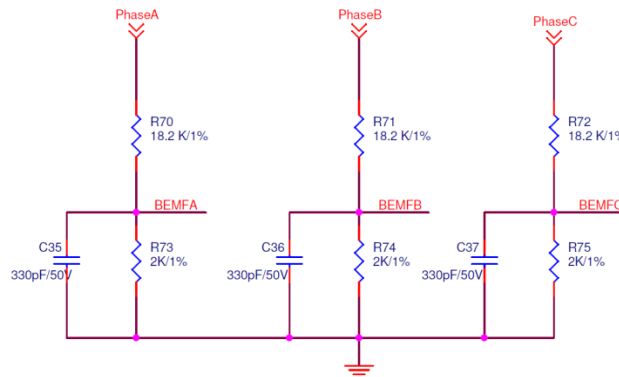
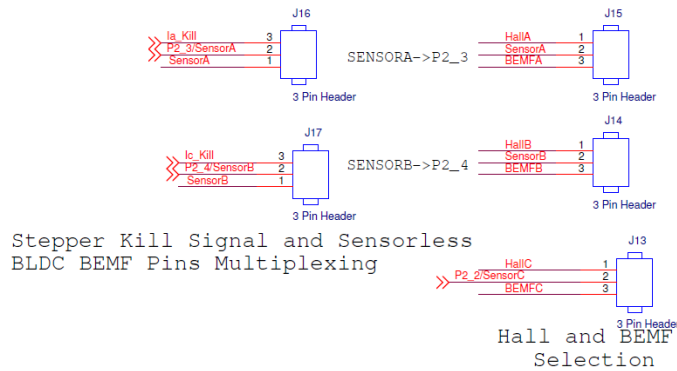


Figure 4-11 shows that the two Hall sensor and BEMF signals, as well as the two external comparator output kill signals of the stepper motor, are routed to J16 and J17. Since the GPIO pin numbers of PSoC 4 are limited, these signals need to share two GPIO pins. Be sure to configure J16 and J17 correctly for the BLDC and stepper motor applications.

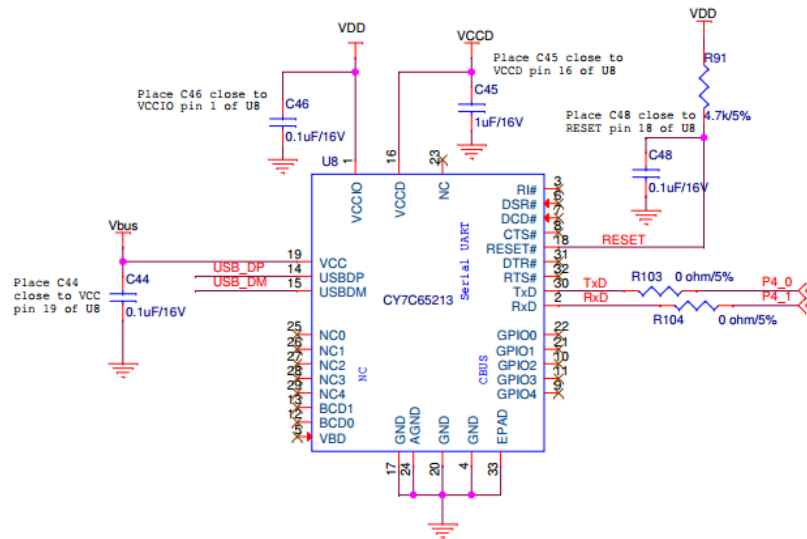
Figure 4-11. Signal Configuration Jumpers



4.7 USB-to-UART Bridge Controller Circuit

Figure 4-12 shows the USB-to-UART bridge controller used for transferring data, such as speed and winding current, that you want to monitor on the BCP. The bridge controller receives the data from PSoC 4 on CY8CKIT-042 over UART and transmits it over USB to be monitored using the BCP software.

Figure 4-12. USB-to-UART Bridge Controller



4.8 Connectors

There are several connectors on the EVK board. Figure 4-13 shows the input power connectors. J7 is the standard power jack for DC adapters, with an output from 12 V to 24 V. J8 can accept an input voltage from 24 V to 48 V through raw wire from a general variable DC power supply. The input current of both connectors can be as high as 2 A.

Figure 4-13. Input Power Supply Connectors

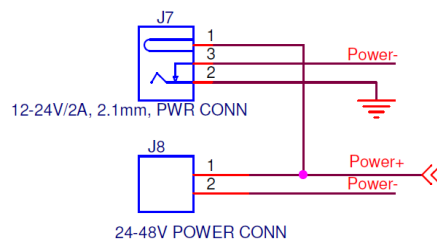


Figure 4-14 shows the motor winding connectors. The two connectors provide four pins in total. The BLDC motor contains three windings with a common neutral point inside the motor. It uses only three pins. The stepper motor contains two independent windings, so it needs all four pins.

Figure 4-14. Motor Winding Connectors

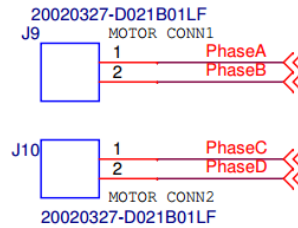
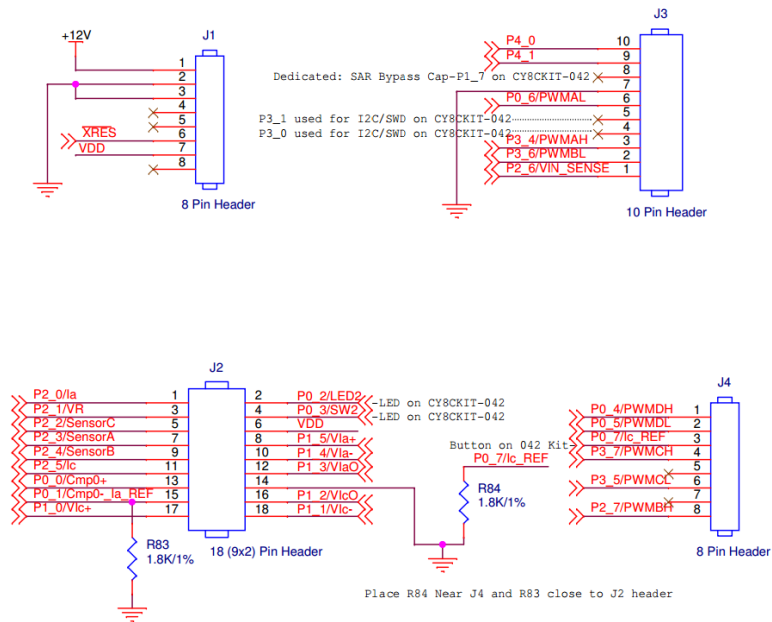


Figure 4-15 shows the connectors to the CY8CKIT-042 board. The layout of connectors J1, J2, J3, and J4 on the EVK board is designed to be pin-to-pin compatible with J1, J2, J3, and J4 on the Pioneer Kit, so users can plug the EVK board directly into the Pioneer Kit.

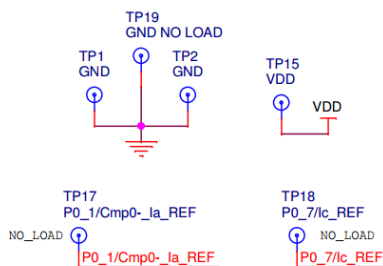
Figure 4-15. Connectors to CY8CKIT-042



4.9 Test Points

The EVK board provides many test points of critical signals, making it convenient for users to debug during the development process. Test points cover driver PWMs, power output and ground, winding current, and IDAC reference output. You can observe and monitor these signals using probes. Figure 4-16 shows some instances of test points.

Figure 4-16. Test Points



CAUTION Do not power the kit through these test points to avoid damage to the board.

5. Code Examples



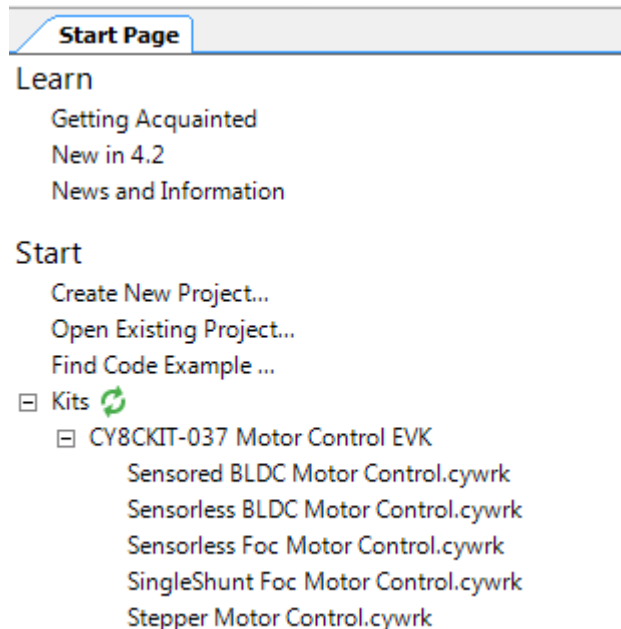
The CY8CKIT-037 EVK can be evaluated using code examples, which can help you accelerate the motor control development processes that configure the device with PSoC 4.

5.1 Using the Kit Code Examples

Follow the steps below to use the installed code examples.

1. Launch PSoC Creator from Start > All Programs > Cypress > PSoC Creator<version> > PSoC Creator <version>.
2. On the Start page, click **CY8CKIT-037 Motor Control EVK** under **Start > Kits**. The code examples shipped with the kit appear (see [Figure 5-1](#)).

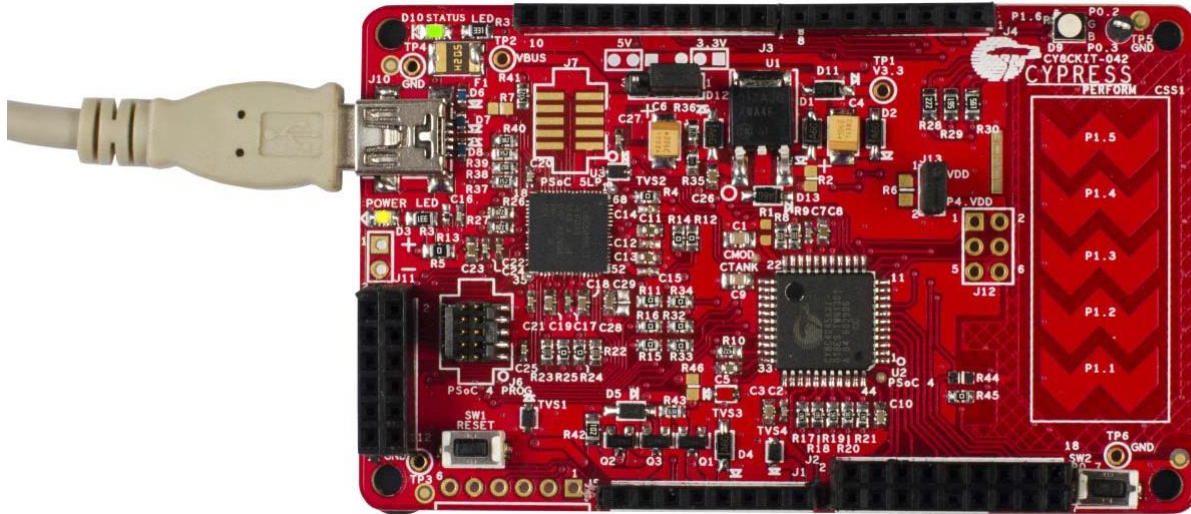
Figure 5-1. Product Installation Overview Window



3. Click on the desired code example, select a location to save the example and click **OK**.
4. Build the code example by choosing **Build > Build <Project_Name>**. After the build process is successful, a `.hex` file is generated.

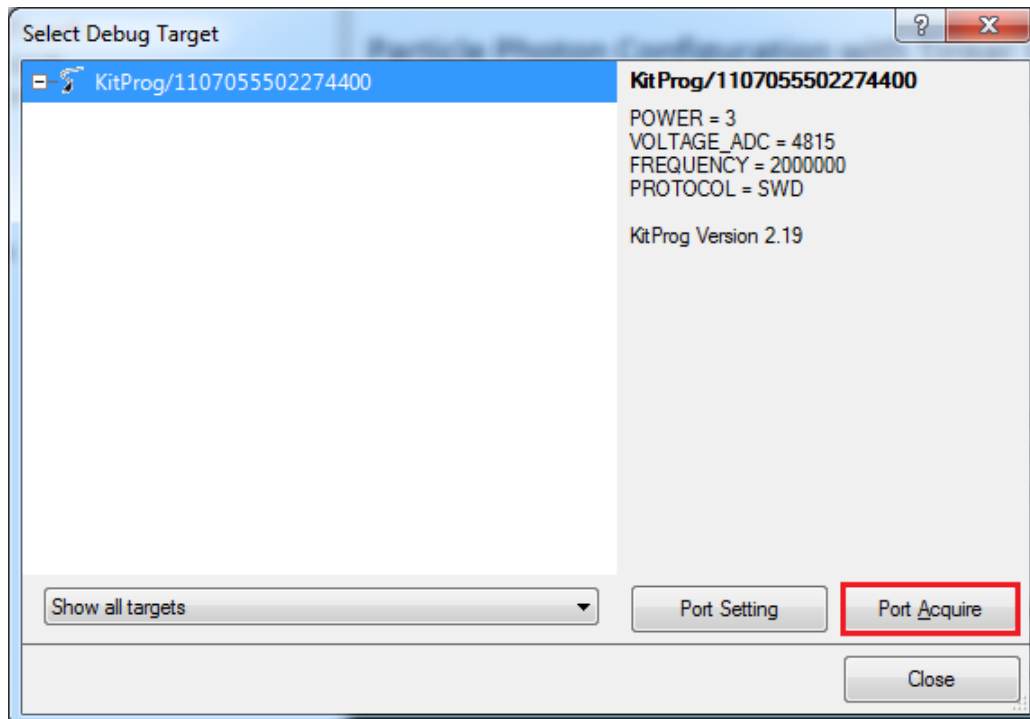
- To program, connect the PSoC 4 Pioneer Kit to the computer using the USB cable connected to port J10.

Figure 5-2. Connect USB Cable to J10



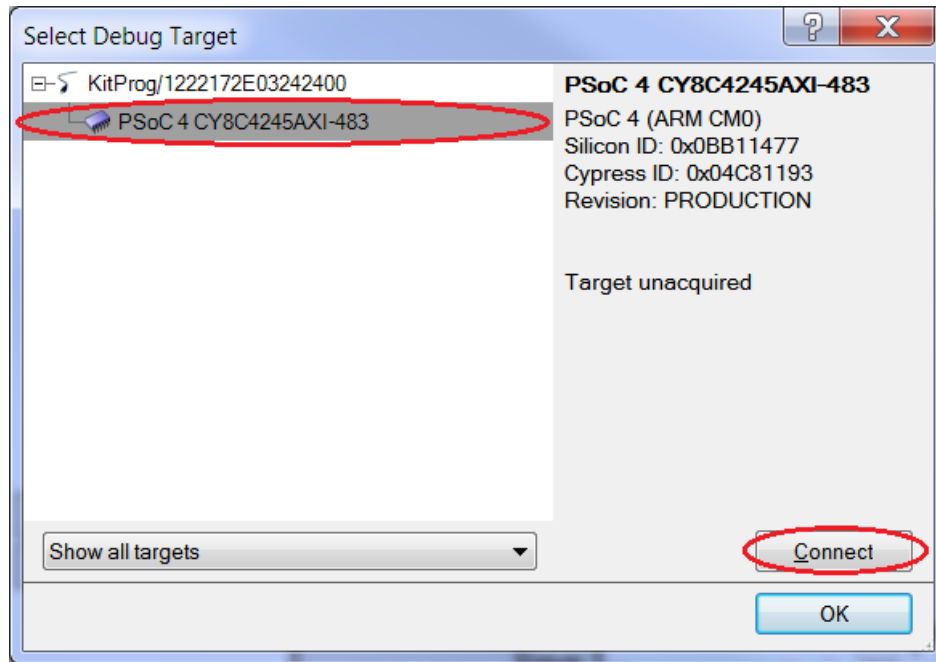
- Choose **Debug > Program** in PSoC Creator.
- If the device is yet to be acquired, the Select Debug Target window will appear. Select **KitProg<serial_number>** and click **Port Acquire**, as shown in Figure 5-3.

Figure 5-3. Acquire Device from PSoC Creator



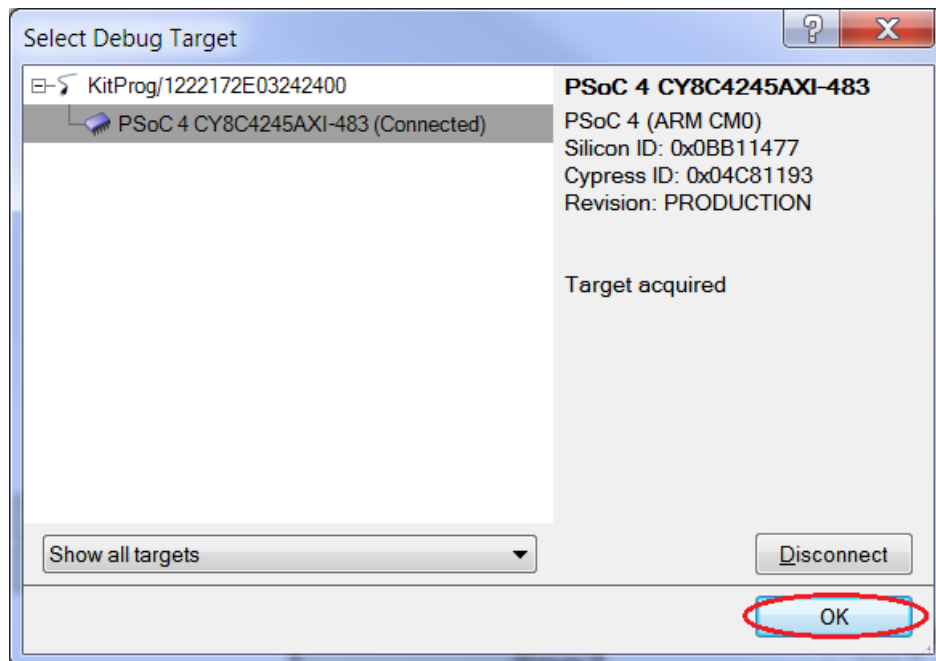
- After the device is acquired, it is shown in a tree structure below the **KitProg/<serial_number>**. Now, click the **Connect** button.

Figure 5-4. Connect Device from PSoC Creator



- Click **OK** to exit the window and start programming.

Figure 5-5. Program Device from PSoC Creator



5.2 Configuration Jumpers for Different Motor Types

CY8CKIT-037 supports three motor types: BLDC, PMSM, and stepper. When switching between different motor types or algorithms, the EVK board needs to be reconfigured via jumpers J13–J24. The jumper configuration table is printed on the top layer of the EVK board, as shown in Figure 5-6.

Figure 5-6. J13–J24 Configuration Table for Different Motors

MOTOR TYPE/JUMPER	J13	J14	J15	J16	J17	J18	J19	J20	J21	J22	J23	J24
HALL SENSOR BLDC	1-2	1-2	1-2	1-2	1-2	1-2	1-2	1-2	2-3	1-2	2-3	1-2
SENSORLESS BLDC	2-3	2-3	2-3	1-2	1-2	1-2	1-2	1-2	2-3	2-3	2-3	1-2
STEPPER	1-2	1-2	1-2	2-3	2-3	1-2	1-2	2-3	1-2	1-2	2-3	2-3
BLDC 1-SHUNT FOC	1-2	1-2	1-2	1-2	1-2	2-3	1-2	1-2	2-3	1-2	1-2	1-2
BLDC 2-SHUNT FOC	1-2	1-2	1-2	1-2	1-2	2-3	2-3	2-3	2-3	1-2	1-2	1-2

These jumpers are located in several different circuit parts. To learn about the principle of configuration, refer to [Hardware](#).

Note: When you prepare to change the configuration jumper settings for different motor types, ensure that all power sources are turned off; otherwise, the kits may be damaged.

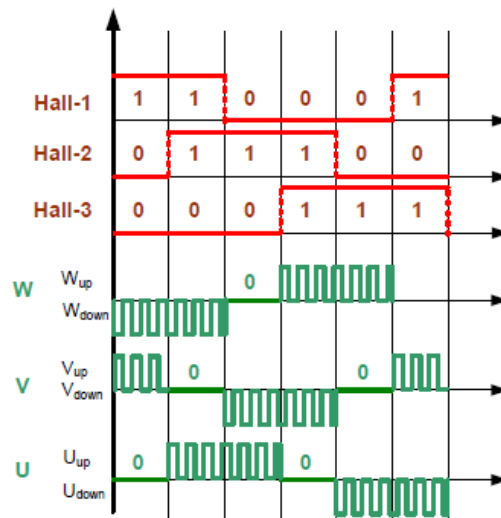
5.3 Sensored BLDC Motor Control Code Example

5.3.1 Sensored BLDC Background

BLDC motors are widely used in industrial applications, home appliances, and vehicle systems. Such motors consist of a multi-pole permanent magnet placed on the rotor and several windings. In the BLDC motor, the commutation is controlled electronically. The motor requires the stator windings to be energized in a particular sequence. To implement this sequence, it is important to know the rotor position. The simplest way is to use rotor position sensors. The sensors can be optical, magnetic (Hall or magneto-resistance effect based), or inductive. The Hall sensor is selected in most applications for its high accuracy and low cost.

Hall sensors are embedded in the stator. When the rotor magnetic poles pass near the Hall sensors, they supply a high or low signal, indicating that the north or south poles are passing nearby. The rotor position is derived from the exact combination of the three Hall sensor signals. Three position sensors can provide six effective states (except 000 and 111) and separate a whole electrical space into six parts, each having a 60-degree electrical angle. Figure 5-7 gives a timing diagram of the sensor output and the required motor driving voltage. The optional use of a PWM provides speed or torque control as shown in phases U, V, and W. The duty cycle of the modulated output control signal (PWM) is varied to change the speed and torque of the motor.

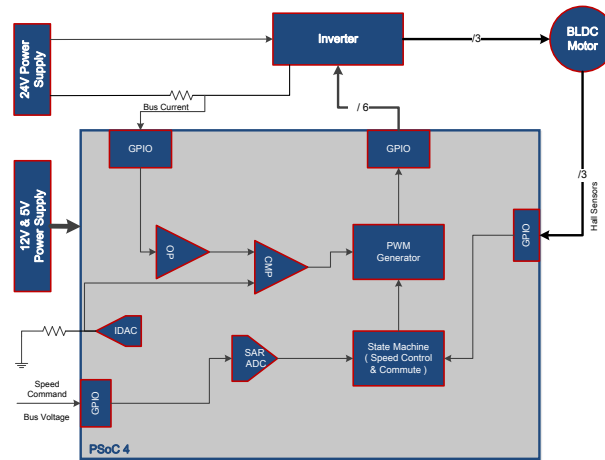
Figure 5-7. BLDC Sensor Output Versus Commutation Timing



5.3.2 Sensored BLDC Motor Control Code Example Overview

Figure 5-8 illustrates the block diagram of the Sensored BLDC Motor Control code example based on PSoC 4.

Figure 5-8. Block Diagram of Sensored BLDC Motor Control Code Example based on PSoC 4



The input control signals to the PSoC 4 device are as follows:

- **Speed command:** Analog input pin that measures the voltage across a potentiometer to set the desired speed of rotation (one analog input pin).
- **Motor current detection:** Analog input pin that detects and cuts off the power device driver to protect the motor when an overcurrent condition is detected (see **Error! Reference source not found.**) (one analog input pin).
- **Hall sensors:** Three digital input pins connected to the outputs of the Hall sensors from the motor. These sensor inputs provide the position of the motor and are used to control the commutation by varying the PWM output signals to the power driver (three digital input pins).
- **Start/stop control:** Digital input connected to a switch to start and stop motor rotation (one digital input pin).

Outputs from the PSoC 4 device are power device driver signals.

- PWM signals to the high side of the MOSFET driver (three digital output pins)
- PWM signals to the low side of the MOSFET driver (three digital output pins)

5.3.3 Control Schematic Overview

The Sensored BLDC Motor Control code example firmware is developed in PSoC Creator 4.2. Its schematic is separated into two pages according to their function. Figure 5-9 shows the “PWM Drive and Commutate” schematic. The three Hall sensor signals are imported by pins “Hall1”, “Hall2”, and “Hall3”. The “LUT_Cmut” outputs the PWM signal to the motor windings according to the Hall signals and its internal commutation table. “PWM_Drive” generates the real-time duty cycle to follow the user’s RPM request.

Figure 5-9. PWM Drive and Commutate Schematic

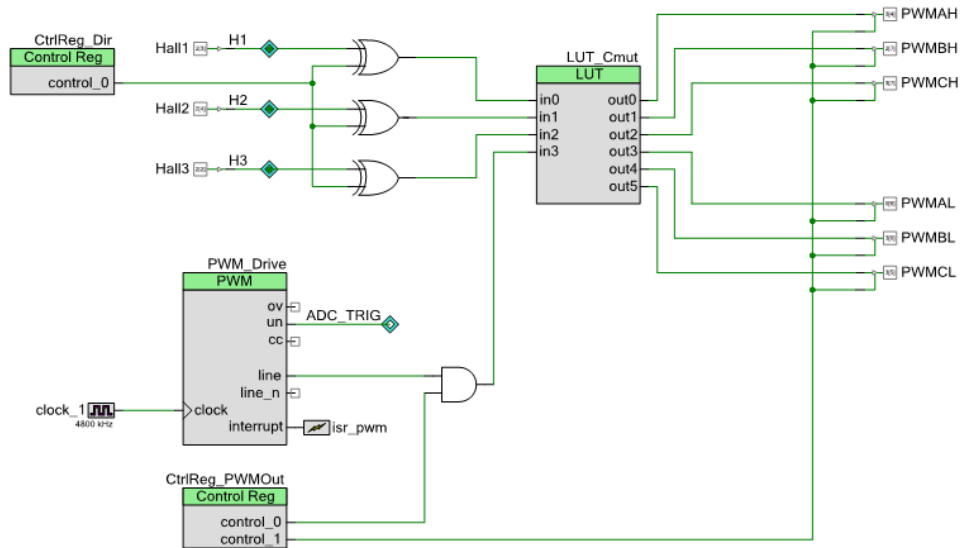


Figure 5-10 shows the “ADC sampling” schematic. “ADC_SAR_Seq_1” is the SAR ADC to detect and measure the bus voltage and potentiometer voltage.

Figure 5-10. ADC Sampling Schematic

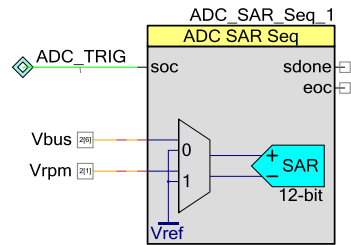


Figure 5-11 shows the “Speed Measurement” schematic. “Counter_Spd” uses one Hall signal to measure the RPM of the motor. It can work as real-time feedback of the closed-loop speed control.

Figure 5-11. Speed Measurement Schematic

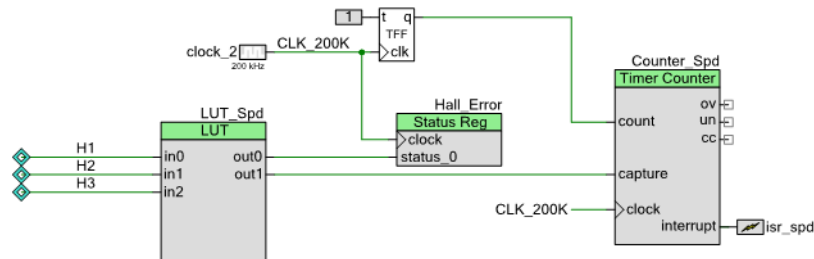
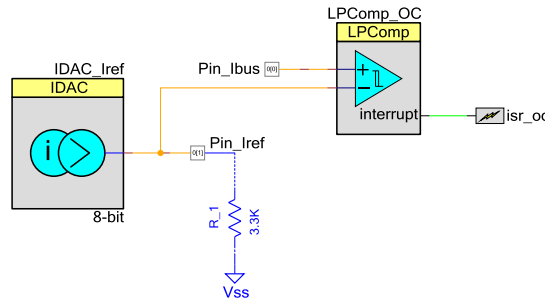


Figure 5-12 shows the “Overcurrent Protection” schematic. The motor current is detected by the sensing resistor and amplified by the external or internal opamp. The current signal is routed to the positive terminal of the internal low-power comparator “LPComp_OC”. The negative terminal of the comparator is the overcurrent threshold set by the internal IDAC “IDAC_Iref.” In an overcurrent condition, “LPComp_OC” will trigger an interrupt, and an overcurrent action will be executed in its ISR.

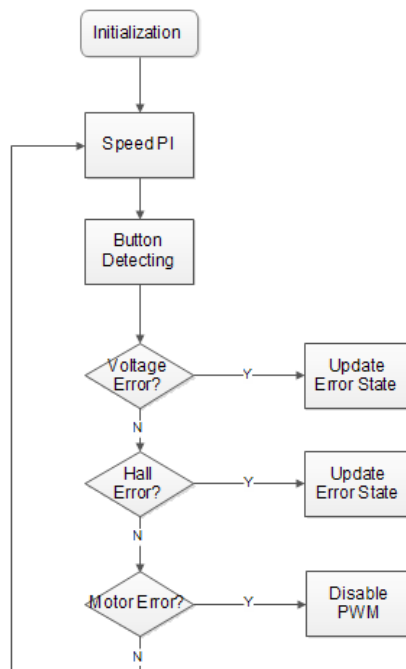
Figure 5-12. Overcurrent Protection Schematic



5.3.4 Flow Chart

Figure 5-13 shows the flow chart of the main loop function. The project schematic in Figure 5-9 shows that hardware performs the commutation. So, the firmware processes only the closed-loop speed control, button detecting, and protective action.

Figure 5-13. Main Loop Function Flow Chart



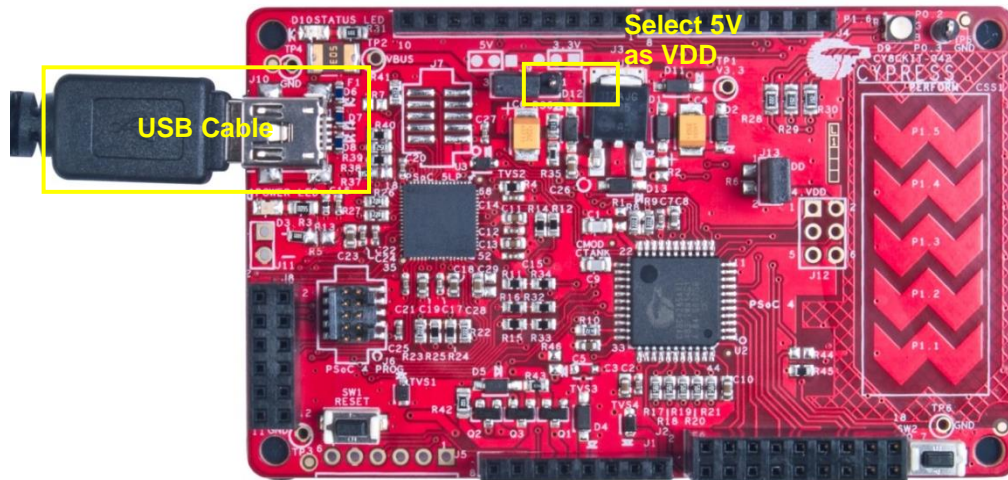
5.3.5 Running the Sensored BLDC Motor Control Code Example

5.3.5.1 Step 1 – Configure CY8CKIT-042

Select 5V as the VDD power at jumper J9 on the Pioneer Kit, and always keep the USB cable connected to the Pioneer Kit because the Pioneer Kit does not provide a 5 V converter. It gets 5 V power directly from the USB port of the PC.

Figure 5-14 shows the CY8CKIT-042 configuration.

Figure 5-14. CY8CKIT-042 Configuration for Sensored BLDC Motor Control Code Example

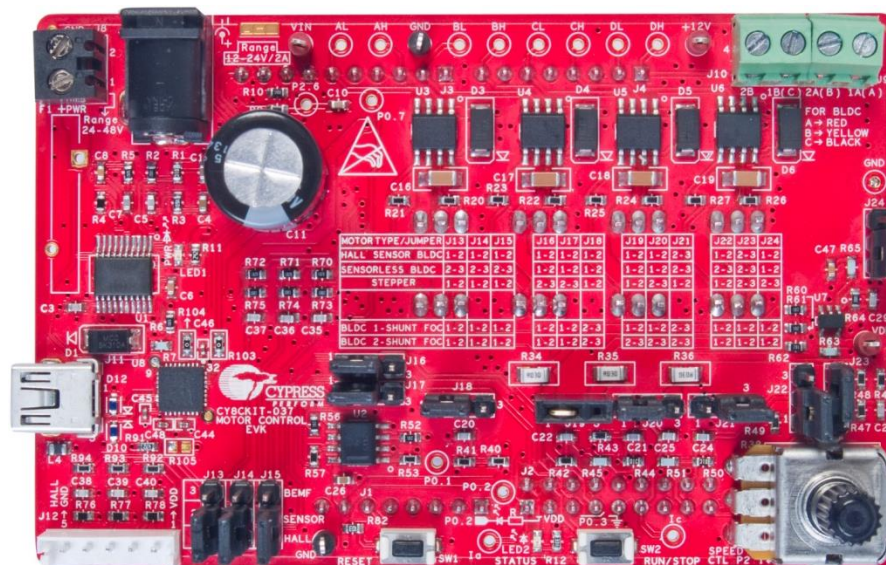


5.3.5.2 Step 2 – Configure CY8CKIT-037

Configure the board via jumpers J13–J24 for the Sensored BLDC Motor Control code example (“HALL SENSOR BLDC” row in Figure 5-6). The CY8CKIT-037 is configured for sensored BLDC control by default. If you have not made any changes since receiving it from Cypress, bypass this step and go to Step 3 – Plug CY8CKIT-037 into CY8CKIT-042.

Figure 5-15 shows the jumper overview after configuration.

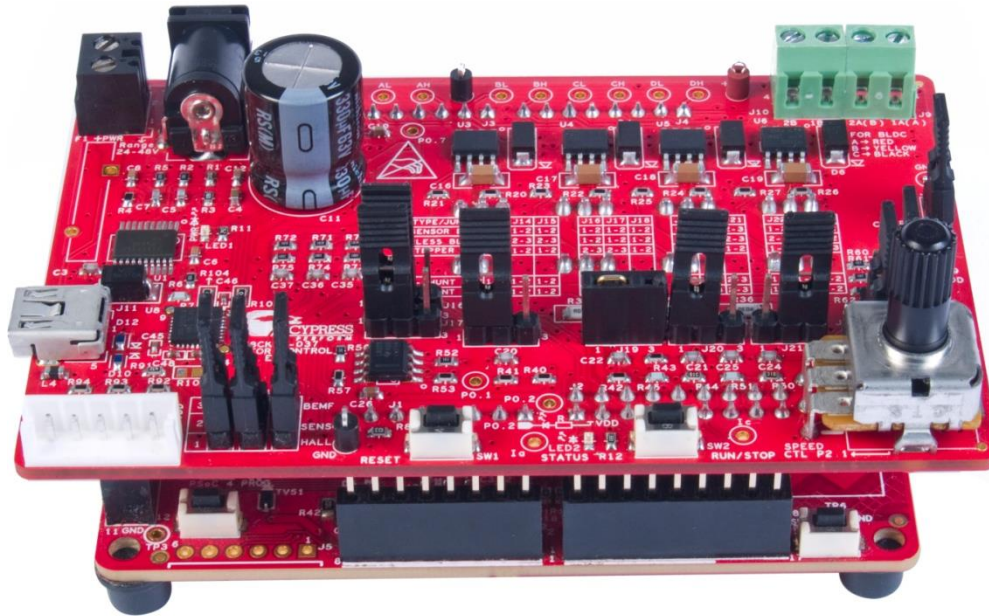
Figure 5-15. Configuration for Sensored BLDC Motor Control Code Example



5.3.5.3 Step 3 – Plug CY8CKIT-037 into CY8CKIT-042

Plug the EVK board into the Pioneer Kit via connectors J1–J4, as shown in [Figure 5-16](#).

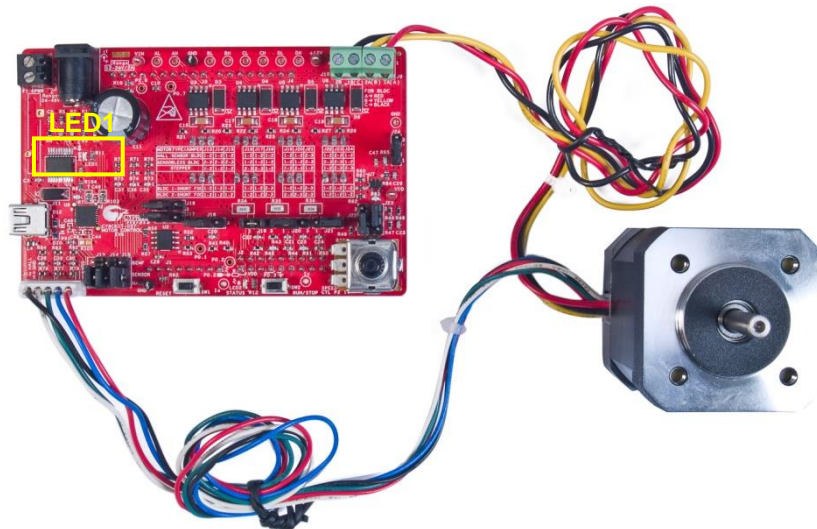
Figure 5-16. Plug CY8CKIT-037 EVK into CY8CKIT-042 Pioneer Kit



5.3.5.4 Step 4 – Connect the Power Supply and Motor

Connect the BLDC motor to the EVK board, including windings to J9/J10 and Hall sensor to J12. The wire sequence and color for the motor windings and Hall sensor should be exactly as shown in [Figure 5-17](#). Refer to [Motor Winding Connectors](#) and [Hall Sensors Interface Connector](#) for instructions. Connect a 24 V power adapter to J7 on CY8CKIT-037. Connect a USB cable to CY8CKIT-042 at connector J10; connect the other end of the USB cable to the PC. LED1 glows, indicating that the kit is powered. If it is not glowing, the fuse F2 may be broken. Replace F2 with the backup fuse provided with the kit.

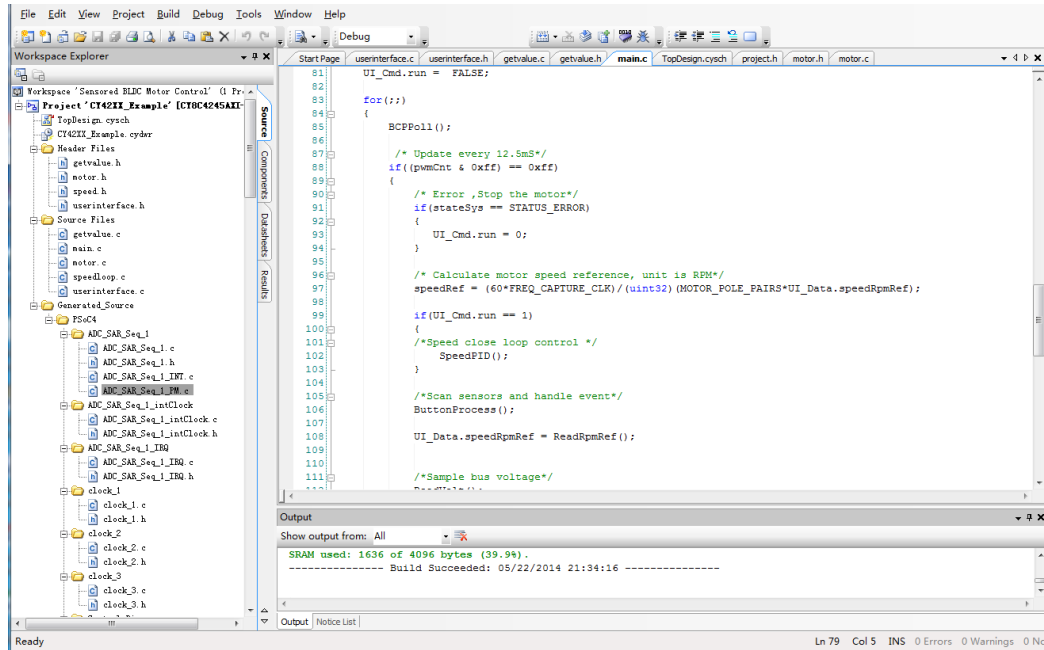
Figure 5-17. Connect Motor and Power Supply



5.3.5.5 Step 5 – Build the Project and Program PSoC 4

Open the Sensored BLDC Motor Control code example in PSoC Creator 4.2 or later version, as shown in [Figure 5-18](#). Choose **Build > Build Sensored BLDC Motor Control** to start building the project, and then choose **Debug > Program** to program the chip.

Figure 5-18. Open the Sensored BLDC Motor Control Code Example

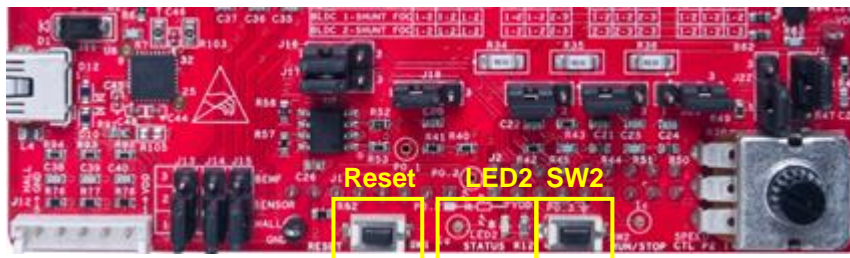


CAUTION During the debugging process, if you modify the code in the firmware, make sure that the changes do not turn on both the high-side and low-side MOSFETs.

5.3.5.6 Step 6 – Press the SW2 Button to Start Motor Rotation

Ensure that the USB cable from the CY8CKIT-042 is connected to the PC in this step; otherwise, the motor will not rotate. Press the **SW2** button to start motor rotation (see [Figure 5-19](#)). Rotate the potentiometer clockwise to increase the motor speed or vice versa. If the motor does not rotate and you observe LED2 blinking, it indicates that an error has occurred. If so, ensure that step 1 through step 5 have been executed correctly. Then press the **Reset** button and press the **SW2** button again. If LED2 still blinks, there must be a problem in the hardware or software. Contact [Cypress](#) for technical support.

Figure 5-19. Buttons and Status LED



CAUTION Do not remove jumpers while the kit is powered. This may damage the kit.

5.3.6 Adapting the Code Example to another Motor

This code example can be changed to rotate other sensed BLDC motors. Some parameters in the project firmware need to be modified according to specific motor characteristics and functional requirements. These parameters are defined as a struct in *motor.h*.

```
typedef struct
{
uint8   Dir; /*Direction*/

uint16  speedRpm; /* Actural motor speed*/
uint16  speedRpmRef; /* motor speed command value*/

uint16  kp;   /* Proportional coefficient of PID*/
uint16  ki;   /* Integral coefficient of PID*/

uint16  maxSpeedRpm; /* Motor parameters*/
uint16  minSpeedRpm; /* Motor parameters*/
uint8   polePairs; /*POLE PAIRS*/
uint8   maxCurr; /*Over current threshold*/
}UI_DATA;
```

The struct variable is in *motor.c* and is initialized in *motor.c* by function *Init_UI_FW*.

```
UI_DATA UI_Data;

void Init_UI_FW(void)
{
/* Setting UI Initial parameter*/
UI_Data.Dir = CLOCK_WISE;
UI_Data.maxSpeedRpm = 4000;
UI_Data.minSpeedRpm = 500;
UI_Data.speedRpmRef = 1000;
UI_Data.polePairs = 4;
UI_Data.maxCurr = MAX_CURR_MEDIUM;
UI_Data.kp = 500;
UI_Data.ki = 50;
}
```

The *Init_UI_FW* function initializes the parameters before the motor begins rotating, and the initializing value is specific to the motor shipped in the kit package. If you want to use your own motor and have different functional requirements such as RPM or direction, change the initializing value in this function.

5.4 Sensorless BLDC Motor Control Code Example

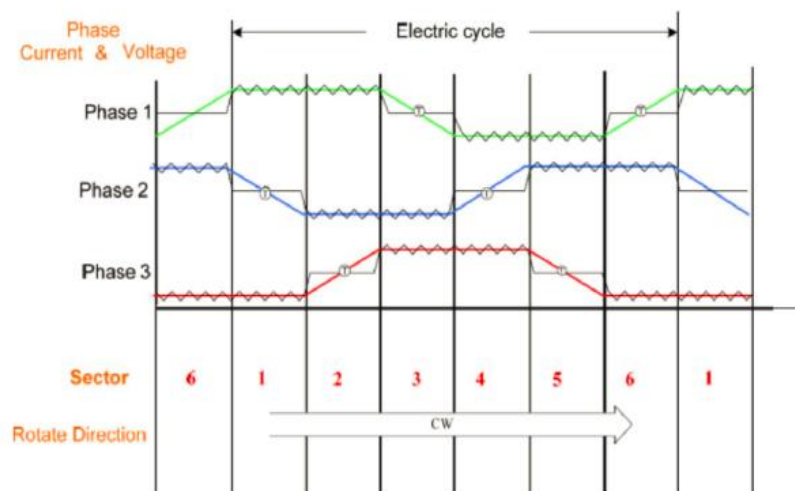
5.4.1 Sensorless BLDC Background

Various methods can be employed to control the BLDC motor. The simplest way is to use Hall position sensors. However, sensors increase cost and add reliability problems in motors operating in harsh environments where demands for sensor robustness are high. The increasing power of embedded computing, coupled with lower prices for power semiconductors and microcontrollers, has allowed more sophisticated methods of motor control. One popular technique is to use a BEMF signal, which is induced by revolving the rotor permanent magnet around the drive coils.

Most BLDC motors have a three-phase winding topology with a star connection. It is driven by energizing two phases simultaneously, while the other phase is kept afloat. The key to BLDC commutation is to sense the rotor position and then energize the phases that produce the maximum amount of torque. The rotor travels 60 electrical degrees at every commutation step. The appropriate stator current path is activated when the rotor is 120 degrees away from alignment with the corresponding stator magnetic field. It is then deactivated when the rotor is 60 degrees from alignment. The next circuit is activated and the process repeats.

Figure 5-20 shows that every commutation sequence has one winding energized positive, the second is negative, and the third winding is left open. The voltage polarity of BEMF crosses from positive to negative or from negative to positive (zero-crossing) between two commutations. Ideally, the zero-crossing of BEMF occurs at 30 electrical degrees after the last commutation and 30 electrical degrees prior to the next commutation. By measuring the zero-crossing of BEMF and the 30-degree time interval, the controller can perform the commutation without a position sensor.

Figure 5-20. General Operating Waveforms of Phase Voltages and Currents



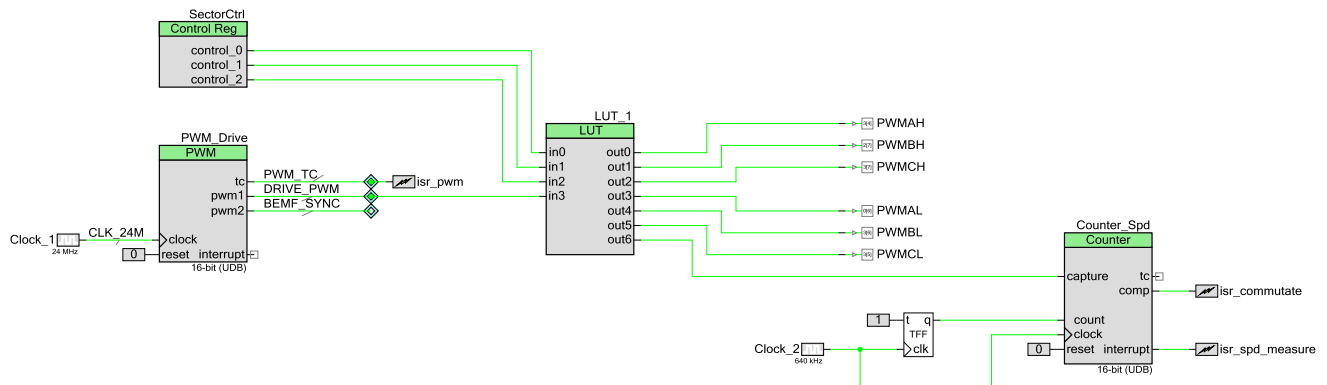
5.4.2 Sensorless BLDC Motor Control Code Example Overview

Figure 5-21 illustrates the block diagram of the Sensorless BLDC Motor Control code example based on PSoC 4. The input control signals to the PSoC 4 device are the following:

- **Speed command:** An analog input pin that measures the voltage across a potentiometer to set the desired speed of rotation (one analog input pin).
- **Motor current detection:** An analog input pin to detect and cut off the power device driver to protect the motor when an overcurrent condition is detected (see [Control Schematic Overview](#)) (one analog input pin).
- **Half Vbus:** An analog input pin that routes half-bus voltage to the negative terminal of the internal low-power comparator.
- **BEMF divider voltage:** Three analog input pins connected to the outputs of the BEMF divider circuit. These BEMF inputs are routed to the internal AMUX, which selects the proper BEMF signal sequentially to compare with the half Vbus to get the zero-crossing point of the non-energized winding.
- **Start/stop control:** A digital input connected to a switch to start and stop rotation of the motor (one digital input pin).

Figure 5-23 shows the “PWM Drive and Commutation” schematic. “SectorCtrl” indicates the current rotor position. It is dynamically changed in the firmware by the zero-crossing detecting ISR. “LUT_1” outputs the PWM signal to the motor windings according to the “SectorCtrl” output and its internal commutation table. “PWM_Drive” generates the right duty cycle to follow the user’s RPM request. “Counter_Spd” is used to measure motor speed; it works as a real-time feedback of the closed-loop speed control.

Figure 5-23. PWM Drive and Commutation Schematic



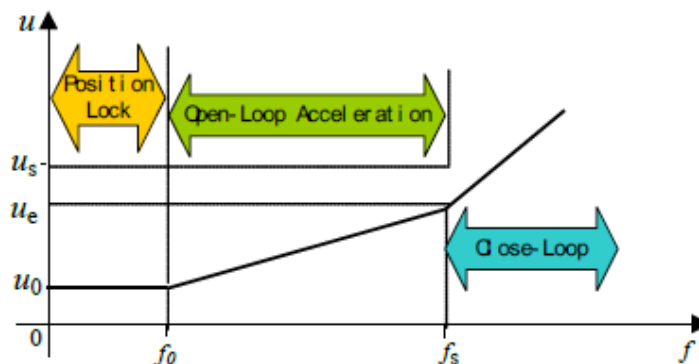
5.4.4 Firmware Introduction

The Sensorless BLDC Motor Control code example employs the typical three-step startup algorithm. The BEMF amplitude of BLDC is proportional to the motor speed. If the motor stops, there is no BEMF. When the motor rotates at a low speed, the BEMF is too weak to be detected. Therefore, before the BEMF can be measured by the firmware, there is a “free-running” stage in which the BLDC motor rotates step by step to acquire the initial BEMF zero-crossing signal before running sensorless control.

When rotating at a certain speed with open-loop stepping stimulation, the rotor position is approximately 90 electrical degrees ahead of that when run correctly under sensorless control. As a result, the BEMF zero-crossings cannot be sensed. To observe the zero-crossings, it is necessary to accelerate the motor at a certain rate.

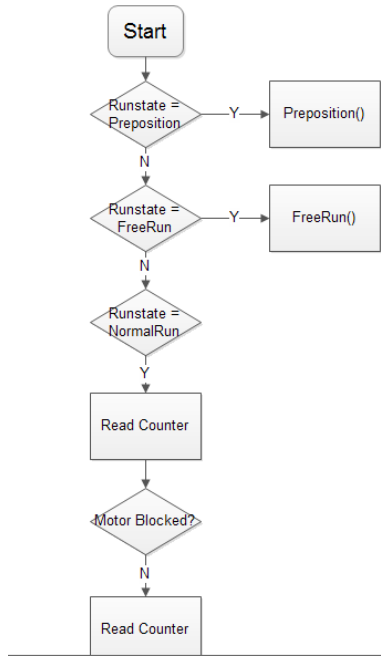
In this stage, the PWM duty cycle increases gradually, and the commutating period is longer than usual. A ramp-up table in the firmware defines the timeout of every commutating period. The value of the table content is a result of experiments based on different motor types. When sufficient valid zero-crossing events are detected, the firmware enters the normal synchronous running stage. If the firmware cannot detect sufficient valid zero-crossing events during a period of time, an error occurs and the BLDC must stop and wait for the next round of the free-running stage. Figure 5-24 shows the starting sequence.

Figure 5-24. Starting Sequence for Sensorless BLDC Motor



The main part of the sensorless startup and normal running control algorithm is in the PWM ISR. [Figure 5-25](#) shows a flow chart of the PWM ISR.

Figure 5-25. PWM ISR Flow Chart



5.4.5 Running the Sensorless BLDC Motor Control Code Example

Perform these steps to run the Sensorless BLDC Motor Control code example.

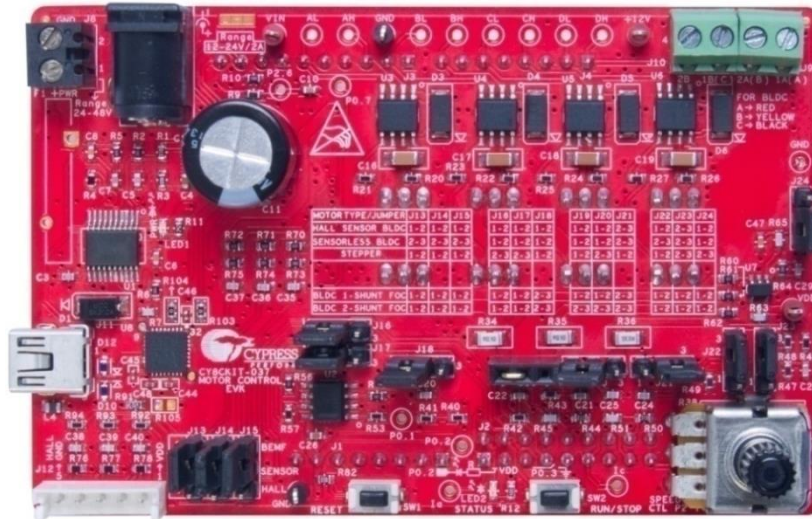
5.4.5.1 Step 1 – Configure CY8CKIT-042

1. Select 5V as the VDD power at jumper J9 on the Pioneer Kit. Ensure to keep the USB cable connected to the Pioneer Kit because the Pioneer Kit does not provide a 5 V converter. It gets 5 V power directly from the USB port of the PC. [Figure 5-14](#) shows the configuration.

5.4.5.2 Step 2 – Configure CY8CKIT-037

1. Configure the board via jumpers J13–J24 for the Sensorless BLDC Motor Control code example (“SENSORLESS BLDC” row in Figure 5-6). Figure 5-26 shows the EVK board configured for the Sensorless BLDC Motor Control code example.

Figure 5-26. EVK Board Configuration for Sensorless BLDC Motor Control Code Example



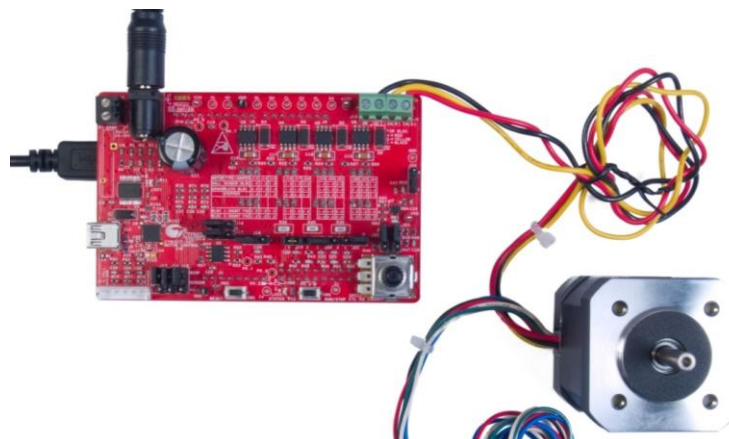
5.4.5.3 Step 3 – Plug CY8CKIT-037 into CY8CKIT-042

1. Plug the EVK board into the Pioneer Kit via connectors J1–J4, as shown in Figure 5-16.

5.4.5.4 Step 4 – Connect the Power Supply and Motor

1. Connect the BLDC windings to J9 and J10 on the EVK board.
2. Connect the 24 V power adapter to J7, and connect a USB cable from J10 on CY8CKIT-042 to the PC, as shown in Figure 5-27. LED1 glows, indicating that the kit is powered. If it is not glowing, the fuse **F2** may be broken. Replace **F2** with the backup fuse provided with the kit.

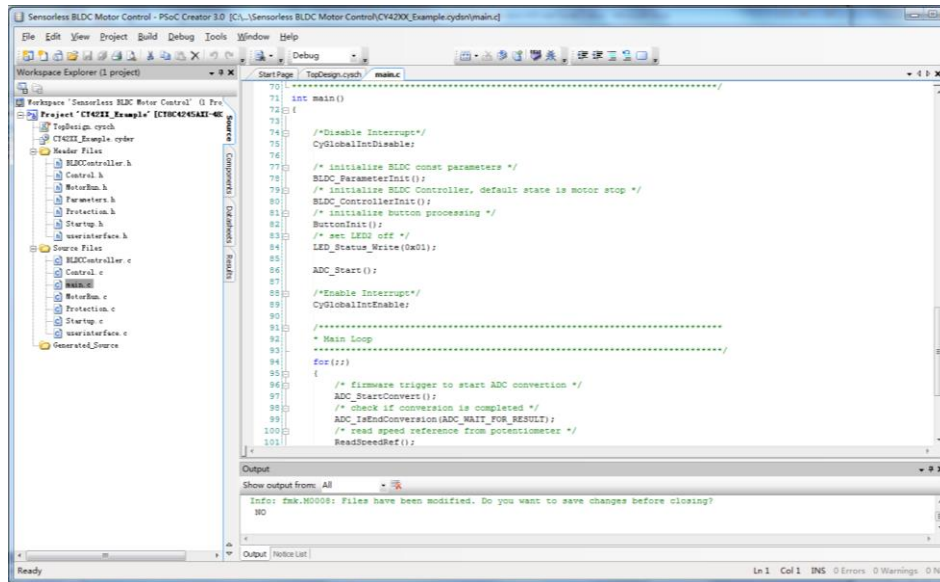
Figure 5-27. Connect Motor and Power Supply



5.4.5.5 Step 5 – Build the Project and Program PSoC 4

1. Open the Sensorless BLDC Motor Control code example in PSoC Creator 4.2 or later version, as shown in [Figure 5-28](#).
2. Choose **Build > Build Sensorless BLDC Motor Control** to start building the project, and then choose **Debug > Program** to program the chip.

Figure 5-28. Open the Sensorless BLDC Motor Control Code Example



CAUTION

During the debugging process, if you modify the code in the firmware, make sure that the changes do not turn on both the high-side and low-side MOSFETs.

5.4.5.6 Step 6 – Press the SW2 Button to Start Motor Rotation

1. Ensure that the USB cable from the CY8CKIT-042 is connected to the PC in this step; otherwise, the motor will not rotate.
2. Press the **SW2** button to start motor rotation (see [Figure 5-19](#)). If the motor does not rotate and you observe LED2 blinking, it indicates that an error has occurred. Ensure that step 1 through step 5 have been executed correctly. Then, press the **Reset** button and press the **SW2** button again. If LED2 still blinks, there must be a problem in the hardware or software. To debug this problem, contact [Cypress](#) for technical support.

CAUTION

Do not remove jumpers while the kit is powered.

5.4.6 Adapting the Code Example to another Motor

This code example can be changed to rotate other sensorless BLDC motors. Some parameters in the project firmware need to be modified according to specific motor characteristics and functional requirements.

5.4.6.1 Tuning Parameters Overview

This code example provides two structure definitions for tuning purposes: `BLDC_Config_T` and `BLDC_Control_T`. “BLDC_Config_T” is defined in the `parameters.h` file. It contains all the constant parameters used in BLDC rotation. The following code illustrates the members of the “BLDC_Config_T” structure.

```

typedefstruct _BLDC_Config
{
    /*-----*
    * motor parameters *
    *-----*/
    uint8 polePairNumber;
    /*-----*
    * general parameters *
    *-----*/
    Direction_T direction;
    uint16 initSpeedRefRpm;
    /*-----*
    * PID parameters *
    *-----*/
    Uint32 kp;
    Uint32 ki;
    /*-----*
    * preposition parameters *
    *-----*/
    uint16 prepositionTime;
}

```

```

    uint16_t prepositionDuty;

    /*-----*/
    * startup parameters *
    /*-----*/

    uint16_t startDuty;
    uint16_t freerunDuty;
    uint16_t startCheckPeriod;

    uint8_t startStageWait;
    uint8_t decStageInterval;

    uint8_t freerunStageWait;
    uint8_t accStageInterval;

    uint8_t accStageWait;
    uint8_t accDutyStep;

    uint8_t accTimeStep;

    /*-----*/
    * normal run parameters *
    /*-----*/

    uint8_t speedCloseLoopWait;
    uint8_t zcCheckSkipCount;
}BLDC_Config_T;

```

A variable of “BLDC_Config_T” is defined in the *BLDCController.c* file as follows.

```
BLDC_Config_T BLDC_Config;
```

The parameters in “BLDC_Config_T” are initialized in the *BLDC_ParameterInit* function (see the following code), which is executed only once when the system powers up. Generally, these parameters should not be changed during motor rotation. A number of macros are also defined in the *parameters.h* file to provide the initial value. You can modify the macro values for tuning based on application requirements or different motors. Note that macro values should be modified before the code example is compiled. Once the project is compiled, changing macro values has no effect.

However, if some additional debugging code is developed to support run-time tuning via the communication interface, such as RS232, SPI, or I²C, real-time updating of the parameters in the “BLDC_Config” variables is possible. Note that the code example included with the kit does not include this kind of debugging code. Kit users must develop it themselves.

```
void BLDC_ParameterInit(void)
{
    /* motor pole-pair number */
    BLDC_Config.polePairNumber = MOTOR_POLE_PAIR_NUM;

    /* motor rotation direction, options are CLOCK and COUNTER_CLOCK */
    BLDC_Config.direction = CLOCK;

    /* speed reference in RPM unit */
    BLDC_Config.initSpeedRefRpm = INIT_SPEED_REF_RPM;

    /* kp for PID control */
    BLDC_Config.kp = KP_INITIAL_VALUE;
    /* ki for PID control */
    BLDC_Config.ki = KI_INITIAL_VALUE;

    BLDC_Config.startDuty = START_PWM_DUTY;
    BLDC_Config.freerunDuty = FREERUN_PWM_DUTY;
    BLDC_Config.startCheckPeriod = START_ZC_CHECK_PERIOD;
    BLDC_Config.startStageWait = START_STAGE_WAIT_COUNT;
    BLDC_Config.decStageInterval = DEC_STAGE_INTERVAL;
    BLDC_Config.freerunStageWait = FREERUN_STAGE_WAIT_COUNT;
    BLDC_Config.accStageInterval = ACC_STAGE_INTERVAL;
    BLDC_Config.accStageWait = ACC_STAGE_EXEC_COUNT;
    BLDC_Config.accDutyStep = ACC_STAGE_DUTY_STEP;
    BLDC_Config.accTimeStep = ACC_STAGE_TIME_STEP;

    BLDC_Config.prepositionTime = PREPOSITION_WAIT_TIME;
    BLDC_Config.prepositionDuty = PREPOSITION_PWM_DUTY;

    BLDC_Config.speedCloseLoopWait = SPEED_CLOSE_LOOP_DELAY;
    BLDC_Config.zcCheckSkipCount = ZC_CHECK_SKIP_COUNT;
}
```

Another structure is “BLDC_Control_T,” which is defined in the *BLDCController.h* file. It contains the system running status, such as the current running speed in RPM, the output of the PID control, and so on. This structure is updated in real time during motor rotation. Monitoring these values is helpful in understanding system performance and troubleshooting. The following code illustrates the members of the “BLDC_Control_T” structure.

```

typedef struct _Run_Control
{
    uint8      runFlag;
    Error_Code_T  errorCode;

    uint8      sector;
    uint8      checkFallingEdge;
    uint8      inNormalRun;

    uint16     speedMeasuredRpm;
    uint16     speedRefRpm;
    uint16     speedGivenRpm;

    uint16     commutateStamp;
    uint16     zeroCrossStamp;
    uint16     zeroCrossPeriod;
    uint16     delayTime;

    int32      pidOutput;
    int32      pidSpeedErr;
}BLDC_Control_T;
  
```

A variable of "BLDC_Control_T" is defined in the *BLDCController.c* file as follows:

```
BLDC_Control_T BLDC_Control;
```

A macro (see the following code) is defined to reset the values of "BLDC_Control" in the *BLDCController.c* file. This macro is executed in the *BLDC_ControllerInit* function when the system powers up, and in the *BLDC_Start* function before the motor starts rotation every time.

```

#define BLDC_CONTROL_INIT_CODE          {\
    BLDC_Control.runFlag = FALSE;\
    BLDC_Control.errorCode = NO_ERROR;\
    BLDC_Control.sector = 1;\
    BLDC_Control.checkFallingEdge = FALSE;\
    BLDC_Control.inNormalRun = FALSE;\
    BLDC_Control.speedMeasuredRpm = 0;\
    BLDC_Control.speedRefRpm = 0;\
    BLDC_Control.speedGivenRpm = 0;\
    BLDC_Control.commutateStamp = 0;\
    BLDC_Control.zeroCrossStamp = 0;\
    BLDC_Control.zeroCrossPeriod = 0;\
  }
  
```

```

BLDC_Control.delayTime = 0;\
BLDC_Control.pidOutput = 0;\
BLDC_Control.pidSpeedErr = 0;\
}

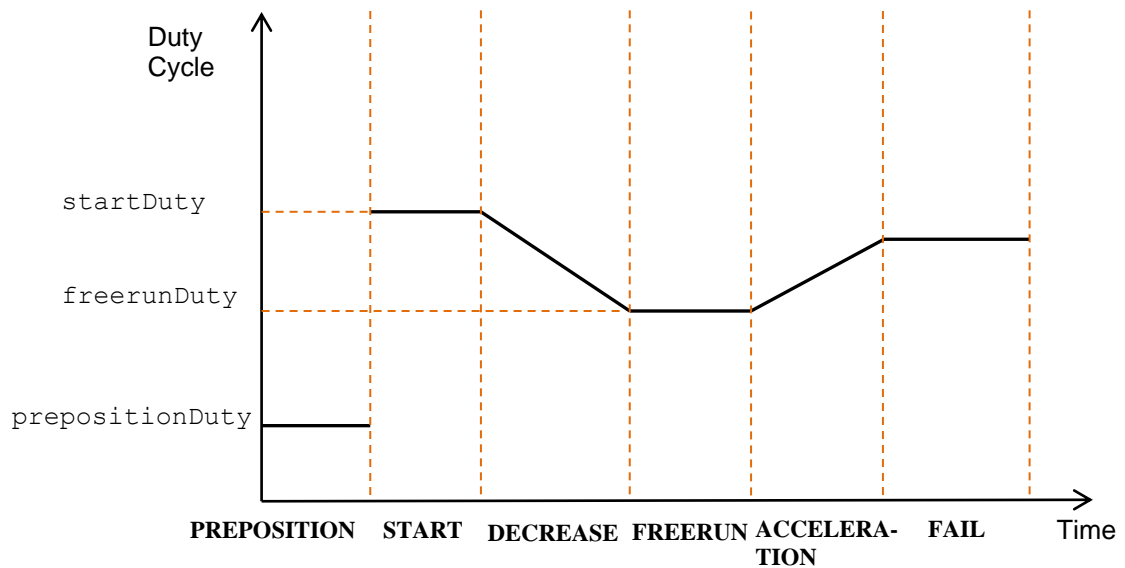
```

5.4.6.2 Tuning Startup Procedure

The startup procedure is important for the sensorless BLDC application. At the beginning of startup, the BEMF is not high enough to detect a stable zero-crossing point, which is used in sensorless control for commutation. The BLDC motor must rotate at a certain speed to generate a high enough BEMF in open-loop mode for a specified time. Then the system switches to position closed-loop mode for zero-crossing point tracking and commutation on demand. After position closed-loop mode is stable, the system enables speed closed-loop mode to force motor rotation close to a given speed reference. Once speed closed-loop mode is stable, the BLDC motor works in normal run mode.

The tuning startup procedure is performed for many possible reasons. One is that a different startup load is applied to the motor; another is that a changing motor requires performance tuning to match the new motor characteristics. The code example provides a multistage startup procedure, as shown in Figure 5-29. This procedure contains six stages, and the PWM duty cycle varies for the different stages.

Figure 5-29. Multistage Startup Procedure



A firmware counter is used to detect the zero-crossing for each stage. Stage switching happens only when the firmware counter is timed out. “startCheckPeriod” sets the timeout period for the firmware counter.

```

uint16 startCheckPeriod;

```

Stage PREPOSITION aligns the motor rotor to a given position for a while and prepares it for the starting rotation. Two parameters are tunable for this stage. “prepositionTime” is a threshold for an internal counter that counts how many times stage PREPOSITION is executed. A larger “prepositionTime” value lengthens the duration of the PREPOSITION stage. “prepositionDuty” is the duty cycle for stage PREPOSITION. A larger “prepositionDuty” value generates a larger preposition torque.

```

uint16 prepositionTime;
uint16 prepositionDuty;

```

Stage START makes the motor rotate from the stop state. Change “startDuty” for a different load status. A larger value generates a larger startup torque for a heavy load. “startStageWait” indicates the duration for which START is executed. It is a threshold for an internal counter that counts how many times stage START is executed. A larger “startStageWait” value lengthens the duration of stage START.

```
uint16 startDuty;  
uint8 startStageWait;
```

Stage DECREASE follows stage START. In this stage, the duty cycle of the PWM is decreased by 1 every time until the duty reaches the “freerunDuty”, which is the minimum duty cycle for the startup procedure. Then the motor maintains rotation in accordance with the “freerunDuty” duty cycle in stage FREERUN. “decStageInterval” sets the interval between two decrease operations. A larger value results in a slower slope of the decrease curve. “freerunStageWait” indicates how long stage FREERUN is executed. It is a threshold for an internal counter that counts how many times stage FREERUN is executed. A larger “startStageWait” value lengthens the duration of stage FREERUN.

```
uint16 freerunDuty;  
uint8 decStageInterval;  
uint8 freerunStageWait;
```

After stage FREERUN, the motor starts to accelerate for a higher BEMF in stage ACCELERATION. “accStageInterval” sets the interval between two increase operations. “accStageWait” indicates how long the stage ACCELERATION is executed. A larger “accStageWait” value lengthens the duration of stage ACCELERATION. “accDutyStep” sets the increase step for the duty cycle, while “accTimeStep” sets the decrease step for the period of commutation in open loop.

```
uint8 accStageInterval;  
uint8 accStageWait;  
uint8 accDutyStep;  
uint8 accTimeStep;
```

If the system cannot detect a valid continuous zero-crossing point in the previous stages, the startup procedure is considered a failure. The system running stage is set as “FREERUNFAILED,” and error code is set as “FREERUN_ERROR.”

If a continuous zero-crossing is detected within a sequence of timeout events, the system switches to position closed-loop mode, no matter which stage of the procedure is occurring. The system then waits for a specified time to enable speed closed-loop mode. This waiting time is specified by “speedCloseLoopWait”. It is a threshold for an internal counter that counts how many times position closed-loop mode is executed. A larger “speedCloseLoopWait” value lengthens the duration of position closed-loop mode.

```
uint8 speedCloseLoopWait;
```


5.4.6.3 Tuning for Different BLDC Motors

When changing the motor, some motor-related parameters or macros need modification, as listed in [Table 5-1](#).

Table 5-1. Motor-Related Parameters for Sensorless BLDC Motor Control Code Example

No.	Parameter	Location	Comment
1	polePairNumber	BLDC_Config_T	Pole-pair number for motor
2	direction	BLDC_Config_T	Desired rotation direction
3	NORMAL_DC_VOLTAGE	Parameters.h	DC bus voltage in volts
4	MAX_SPEED_REF_RPM	Parameters.h	Maximum rotation speed
5	MIN_SPEED_REF_RPM	Parameters.h	Minimum rotation speed
6	START_PWM_DUTY	Parameters.h	The duty cycle of startup. Its value depends on input voltage, motor characteristic, and load status. Generally, it is tuned based on the try-observe method. The more the load, the larger the duty cycle value.
7	FREERUN_PWM_DUTY	Parameters.h	Default duty cycle for free run in open loop
8	INIT_SPEED_REF_RPM	Parameters.h	Initial speed reference after motor enters closed loop. Generally, it should be 15 to 25 percent of the maximum speed if no value is specified.

5.4.6.4 Status LED Display and Error Code

Multiple error states are defined in the *BLDCController.h* file as follows.

```
typedef enum _Error_Code
{
    NO_ERROR,                /* no error happens */
    ZC_CHECK_ERROR,         /* zero-crossing detection failure */
    OV_ERROR,               /* overvoltage happens */
    UV_ERROR,               /* undervoltage happens */
    FREERUN_ERROR,         /* freerun stage fails, fail to enter close loop */
    ANY_ERROR,             /* for any unknown error */
    ERROR_SIZE             /* variable to store count of error types */
}Error_Code_T;
```

A status LED is used to display the system status. When the motor is rotating normally, the LED is turned on. When the motor is stopped normally, the LED is turned off. When an error occurs, the status LED flashes. The number of flash times indicates the error code. For example, if the status LED flashes three times, the UV_ERROR has occurred.

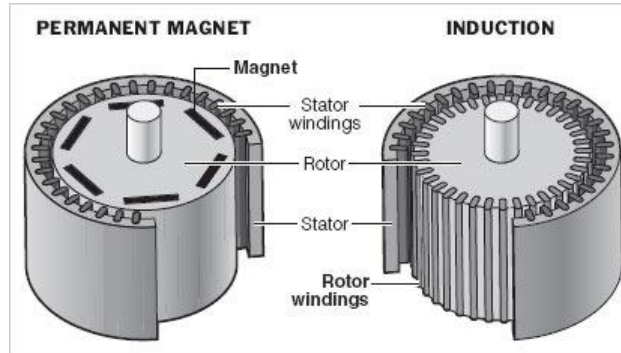
5.5 Sensorless FOC Motor Control Code Example

5.5.1 Sensorless FOC Background

5.5.1.1 PMSM Introduction

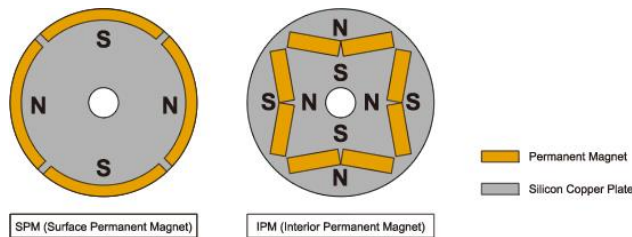
PMSM is the most popular motor type in industrial, home appliance, and automotive medium- and high-end applications. As [Figure 5-30](#) shows, PMSM has a structure similar to that of the AC induction motor. PMSM has a wound stator and a permanent magnet rotor that provide sinusoidal flux distribution in the air gap, making the BEMF inform a sinusoidal shape. The construction of the stator and rotor can provide lower rotor inertia and high power efficiency and reduce the motor size.

Figure 5-30. PMSM Structure



Depending on how magnets are attached to the rotor, PMSM motors can be classified into two types: surface PMSM (SPMSM) and interior PMSM (IPMSM). [Figure 5-31](#) illustrates the differences between these two types. SPMSM mounts all magnet pieces on the surface, and IPMSM places magnets inside the rotor. SPMSM is widely used for simple manufacturing and assembling, while IPMSM optimizes performance by the specific placement of magnets. “PMSM” refers to “SPMSM” in the remainder of this guide, unless otherwise specified.

Figure 5-31. SPMSM and IPMSM Structural Differences



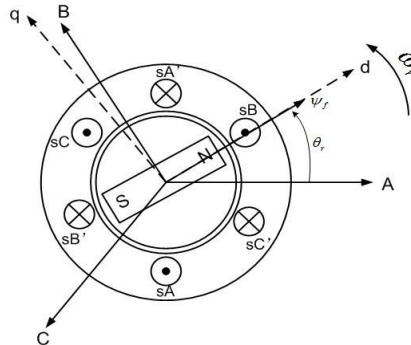
5.5.1.2 PMSM Mathematic Model

A mathematical model of PMSM motors is essential for control and performance analysis. Before discussing any model, some assumptions should be set.

- PMSM winding connection is in “Y” type; any delta connections need to convert to “Y” type.
- Magnetic saturation is neglected.
- Eddy currents and hysteresis losses are negligible.

Figure 5-32 illustrates a three-phase stator reference frame, which is the basic PMSM model. In this frame, the axis of **A**, **B**, **C** is aligned with the direction of the PMSM stator windings; Ψ_f is the flux linkage vector of the rotor magnet. The rotor rotates at angular speed ω_r and generates an angle θ_r between Ψ_f and phase **A**.

Figure 5-32. Three-Phase Stator Reference Frame



The voltage on the stator winding can be presented as:

$$\begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix} = \begin{bmatrix} R_a & 0 & 0 \\ 0 & R_b & 0 \\ 0 & 0 & R_c \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + \rho \begin{bmatrix} \Psi_a \\ \Psi_b \\ \Psi_c \end{bmatrix}$$

Where,

$[u_a \ u_b \ u_c]^T$ Stator voltage vector

$diag[R_a \ R_b \ R_c]$ Stator resistance

$[i_a \ i_b \ i_c]^T$ Stator current vector

ρ Differential operator d/dt

$[\Psi_a \ \Psi_b \ \Psi_c]^T$ Stator flux linkages

The stator winding flux linkages can be written as the sum of the flux linkages due to their own excitation, mutual flux linkages resulting from other winding currents, and flux linkages from magnet sources of the rotor. As stator windings have 120° angles in mechanical space, the stator flux linkages can be expressed as follows:

$$\begin{bmatrix} \Psi_a \\ \Psi_b \\ \Psi_c \end{bmatrix} = \begin{bmatrix} L_{aa}(\theta_r) & M_{ab}(\theta_r) & M_{ac}(\theta_r) \\ M_{ba}(\theta_r) & L_{bb}(\theta_r) & M_{bc}(\theta_r) \\ M_{ca}(\theta_r) & M_{cb}(\theta_r) & L_{cc}(\theta_r) \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + \Psi_f \begin{bmatrix} \cos \theta_r \\ \cos(\theta_r - 120^\circ) \\ \cos(\theta_r + 120^\circ) \end{bmatrix}$$

Where,

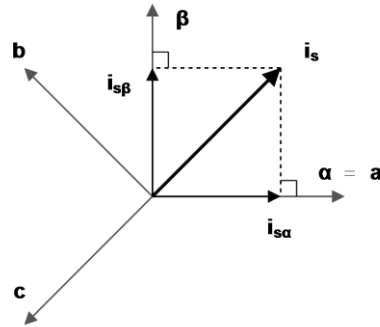
L_{aa}, L_{bb}, L_{cc} Equivalence inductance of stator

M_{aa}, M_{bb}, M_{cc} Mutual equivalence inductance of stator

Ψ_f Amplitude of rotor flux linkages

This model is high order, strongly coupled, and nonlinear. It is difficult to analyze and control torque and flux; therefore, a two-phase d-q model is developed to simplify the three-phase model. The d-q model is based on a rotating two-phase reference frame that is aligned with the rotor flux. It first converts the three-phase reference frame (a, b, c) to a two-phase stationary reference frame (α , β) using the Clarke transformation (see Figure 5-33).

Figure 5-33. Reference Frame Conversion Using Clarke Transformation



The current vector $[i_\alpha \ i_\beta]^T$ in the (α , β) frame is:

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}$$

For “star” type winding connections, the sum of three-phase currents is zero.

$$i_a + i_b + i_c = 0$$

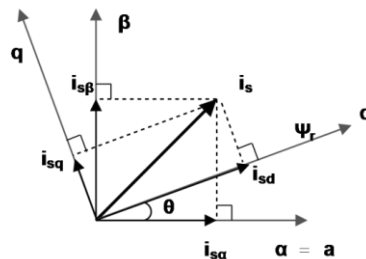
Therefore, the current vector on the α -axis and β -axis can be presented as:

$$\begin{aligned} i_\alpha &= i_a \\ i_\beta &= \frac{1}{\sqrt{3}}(i_a + 2i_b) \end{aligned}$$

Then a Park transformation converts the (α , β) frame to a rotating two-phase reference frame (d, q) aligned with the rotor flux (see Figure 5-34). The current vectors in the (d, q) frame are:

$$\begin{aligned} i_d &= i_a \cos \theta + i_\beta \sin \theta \\ i_q &= -i_a \sin \theta + i_\beta \cos \theta \end{aligned}$$

Figure 5-34. Reference Frame Conversion Using Park Transformation



Now the voltage in the d-q frame can be calculated by i_d and i_q .

$$\begin{aligned} u_d &= Ri_d + \frac{d\Psi_d}{dt} - \omega_r \Psi_q \\ u_q &= Ri_q + \frac{d\Psi_q}{dt} + \omega_r \Psi_d \end{aligned}$$

$$\Psi_d = L_d i_d + \Psi_f$$

$$\Psi_q = L_q i_q$$

The torque equation is:

$$\begin{aligned} T_e &= \frac{3}{2} P_n (\Psi_d i_q - \Psi_q i_d) \\ &= \frac{3}{2} P_n [\Psi_f i_q - (L_q - L_d) i_d i_q] \end{aligned}$$

Where,

P_n = Pole-pair number of rotor

Note that L_d and L_q have no relationship with electrical angle θ , and L_q is equal to L_d for SPMSM; therefore, the torque can be expressed as

$$T_e = \frac{3}{2} P_n \Psi_f i_q$$

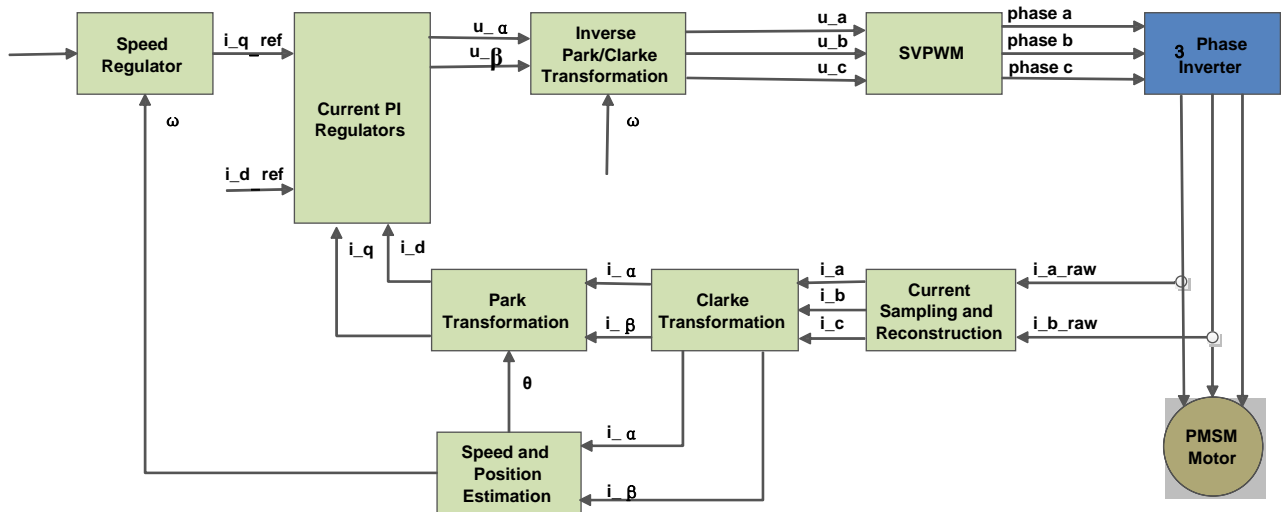
Both P_n and Ψ_f are related to motor characters, and they are not affected by motor operation. Therefore, torque is proportional only to the q-axis current i_q in the d-q model, which provides a simple method to control torque generation.

5.5.1.3 Field-Oriented Control

FOC, also called “vector control,” employs the d-q model. It is a control technique used in PMSM and AC induction motors for high-performance motor applications. It can operate smoothly over a wide speed range and is capable of fast acceleration and deceleration. It can easily control the motor torque to reduce ripple and achieve smooth rotation with lower noise and vibration.

Figure 5-35 illustrates the control block diagram for FOC. Compared to the basic d-q model, it introduces four new blocks: Speed and Position Estimation, Speed Regulator, Current PI Regulators, and Space Vector PWM (SVPWM).

Figure 5-35. Sensorless FOC Block Diagram



Speed and Position Estimation samples the feedback signal from the motor and calculates the angular speed and position of the rotor. This information is employed in the Clarke and Park transformations to convert three-phase currents to i_d and i_q in the d-q model.

Speed Regulator and Current PI Regulators employ control algorithms to ensure that rotation speed and d-q current remain close to the commanded values, which are set by the user or host system. The output of Current PI Regulators is transformed to three-phase voltages by inverse Clarke and Park transformation.

The SVPWM, also called “SVM”, is used to generate a sinusoidal current waveform to feed to the stator coils. Suppose VT_1 through VT_6 are the six power transistors that shape the output, and VT_1, VT_3, VT_5 are the upper transistors. When an upper transistor is switched on, the corresponding lower transistor is switched off.

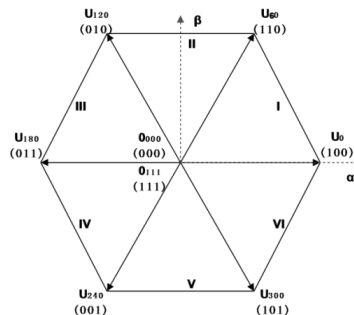
If recording the ON state of the upper transistor as “1” and the OFF state as “0”, there are eight possible combinations of ON and OFF states. Corresponding to these different combinations are eight outputs, as shown in Table 5-2. u_a, u_b, u_c are the phase (line-to-neutral) voltages, while u_{ab}, u_{bc}, u_{ac} are the line-to-line voltages.

Table 5-2. Output Combination in Three-Phase Frame

ON/OFF State			Line-to-Line Voltage (Vdc)			Phase Voltage (Vdc)		
VT_1	VT_3	VT_5	u_{ab}	u_{bc}	u_{ac}	u_a	u_b	u_c
1	0	0	1	0	-1	2/3	-1/3	-1/3
1	1	0	0	1	-1	1/3	1/3	-2/3
0	1	0	-1	1	0	-1/3	2/3	-1/3
0	1	1	-1	0	1	-2/3	1/3	1/3
0	0	1	0	-1	1	-1/3	-1/3	2/3
1	0	1	1	-1	0	1/3	-2/3	1/3
0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0

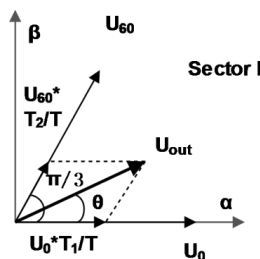
The eight combinations in Table 5-2 can be converted to six nonzero vectors and two zero vectors, as shown in Figure 5-36. The nonzero vectors form the axes of a hexagon, and the angle between any adjacent two nonzero vectors is 60° . This divides the hexagon into six sectors: I through VI. The zero vectors are at the origin and apply zero voltage to a three-phase load. The eight vectors, called the “basic space vectors”, are denoted by $U_0, U_{60}, U_{120}, U_{180}, U_{240}, U_{300}, 0_{000}$, and 0_{111} .

Figure 5-36. Space Vectors



Assume U_{out} is the desired reference voltage in the (α, β) frame. The reference voltage U_{out} can be produced by a combination of basic space vectors using the SVPWM technique. Figure 5-37 shows an example. U_{out} is in sector I ($U_0 \sim U_{60}$), and the period of PWM is T . T_1 is the respective duration for U_0 appearance; T_2 is the respective duration for U_{60} appearance; T_0 is the duration of zero vectors.

Figure 5-37. Voltage Vector in Sector I



Then, the U_{out} can be expressed as:

$$T = T_1 + T_2 + T_0$$

$$U_{out} = U_{60} \times \frac{T_2}{T} + U_0 \times \frac{T_1}{T}$$

Therefore,

$$|U_{out}| \cos \theta = |U_{60}| \times \frac{T_2}{T} \times \cos \frac{\pi}{3} + |U_0| \times \frac{T_1}{T}$$

$$|U_{out}| \sin \theta = |U_{60}| \times \frac{T_2}{T} \times \sin \frac{\pi}{3}$$

Then,

$$T_2 = mT \sin \theta$$

$$T_1 = mT \sin \left(\frac{\pi}{3} - \theta \right)$$

$$m = \sqrt{3} \times \frac{|U_{out}|}{U_{dc}}$$

$$|U_{out}| = \sqrt{u_\alpha^2 + u_\beta^2}$$

According to the ON/OFF state of the upper transistor in U_{60} , U_0 , and zero vector, the U_{out} can be generated by changing the duty cycles of PWMs applied to the inverter.

5.5.1.4 Sensorless FOC Technique

The Park transformation requires the rotor position information, which is presented as the angle θ between the rotor flux linkage and α -axis. This is calculated by the Speed and Position Estimation block based on the motor feedback signals.

In a sensed design, the feedback signal comes from specific sensors, such as Hall sensors or optical encoders. These sensors not only increase the total cost of the motor system, but they also require maintenance, as they may fail during the motor's lifetime. The sensorless technique removes the physical sensors and replaces them with an algorithm in a microcontroller.

The idea of the sensorless technique is to estimate position based on the BEMF due to its relationship with angle θ . The BEMF can be measured directly. However, the measurement circuitry has to handle high voltages; therefore, this technique is not encouraged.

The two-phase voltages in the (α, β) frame can be expressed as:

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = \begin{bmatrix} R_s + \rho L & 0 \\ 0 & R_s + \rho L \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix}$$

Where,

- R_s Resistance of winding phase
- L $L_d = L_q = L$ for SVPWM (where L is inductance)
- ρ Differential operator d/dt
- $\begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix}$ BEMF excited by rotor magnet

In the digital domain, the equation can be changed to:

$$\frac{i_\alpha(n+1) - i_\alpha(n)}{T_s} = \left(-\frac{R_s}{L_s} \right) i_\alpha(n) + \frac{1}{L_s} [u_\alpha(n) - e_\alpha(n)]$$

Where,

- T_s Period of PWM on inverter

Solving for i_α as:

$$i_\alpha(n+1) = \left(1 - T_s \frac{R_s}{L_s} \right) i_\alpha(n) + \frac{T_s}{L_s} [u_\alpha(n) - e_\alpha(n)]$$

Note that R_s and L_s are the motor characteristics, which can be measured by equipment. T_s is a constant system parameter, $i_\alpha(n)$ is a sampling result in the last control cycle, and $u_\alpha(n)$ is the calculation result in the last control cycle. Therefore, if $e_\alpha(n)$ is given as $e_\alpha^*(n)$, an estimated current value $i_\alpha^*(n+1)$ can be obtained. The "*" here indicates it is a value estimated by algorithm and not directly measured.

Comparing $i_{\alpha}^*(n+1)$ with the actual current value $i_{\alpha}(n+1)$ sampled by ADC, an error between these two can be fed to adjust the given value $e_{\alpha}^*(n)$ for better estimation. This process is repeated until the error between $i_{\alpha}^*(n+1)$ and $i_{\alpha}(n+1)$ is as small as required. Then the given value $e_{\alpha}^*(n)$ represents the actual BEMF $e_{\alpha}(n)$.

The $e_{\beta}(n)$ can be obtained in the same way, and the angle θ can be calculated as:

$$\theta(n) = \arctan \frac{-e_{\alpha}(n)}{e_{\beta}(n)}$$

Angular speed ω_r is calculated by accumulating θ over m samples and is multiplied by speed constant K .

$$\omega_r = \sum_{n=1}^m [\theta(n) - \theta(n-1)] * K$$

Thus, the position and speed information are retrieved from the estimated BEMF.

5.5.2 Sensorless FOC Motor Control Code Example Overview

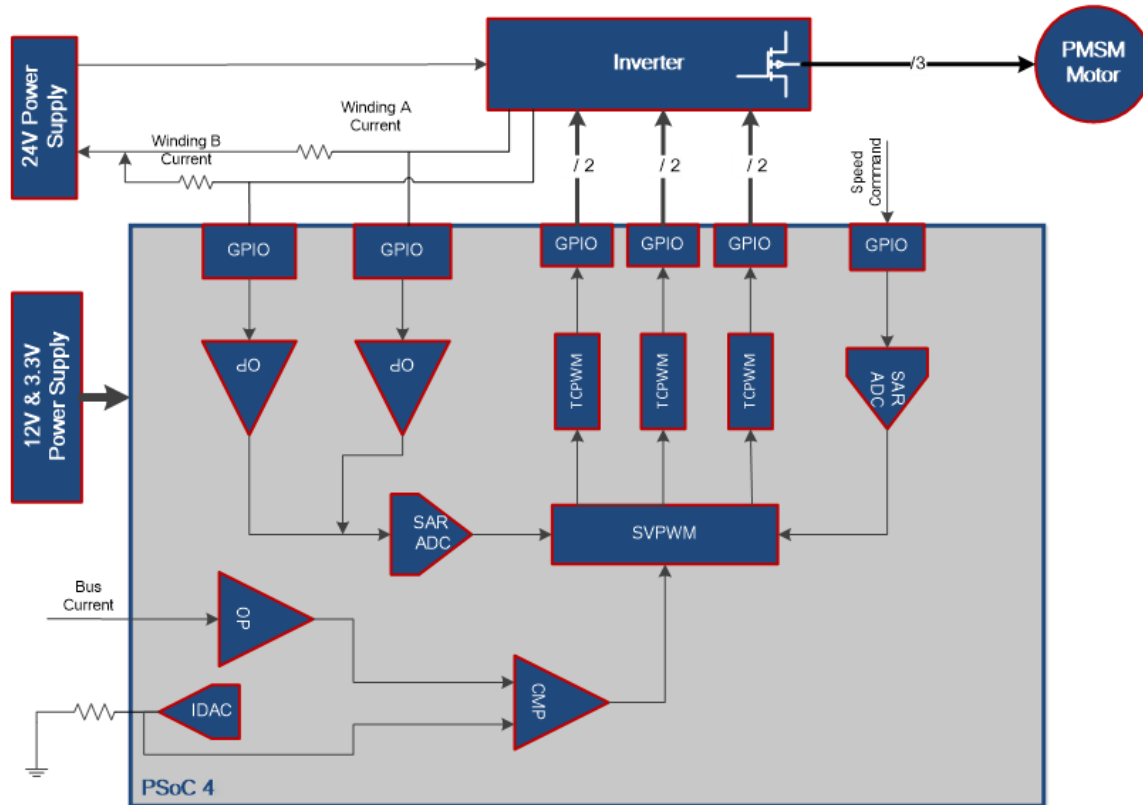
Figure 5-38 illustrates the block diagram of the Sensorless FOC Motor Control code example based on PSoC 4. The input control signals to the PSoC 4 device are the following.

- **Speed command:** An analog input pin that measures the voltage across a potentiometer to set the desired speed of rotation (one analog input pin).
- **Motor current measurement:** Two analog input pins to detect the motor winding current by sensing resistors. Then the signal is routed to internal opamps to be amplified and filtered.
- **Start/Stop control:** Digital input connected to a switch to start and lock rotation of the motor (one digital input pin).

Outputs from the PSoC 4 device are power device driver signals.

- PWM signals to the high side of the MOSFET driver (three digital output pins)
- PWM signals to the low side of the MOSFET driver (three digital output pins)
- Overcurrent limit voltage set by the internal IDAC and an outside resistor

Figure 5-38. Sensorless FOC Motor Control Code Example Block Diagram



5.5.3 Control Schematic Overview

The Sensorless FOC Motor Control code example firmware is developed in PSoC Creator 4.2. Its schematic has one page. [Figure 5-39](#) shows the PWM design in the Sensorless FOC Motor Control code example schematic. The three TCPWM Components generate three couples of the center-aligned and complemented SVPWM signal according to the FOC algorithm.

Figure 5-39. PWM Schematic

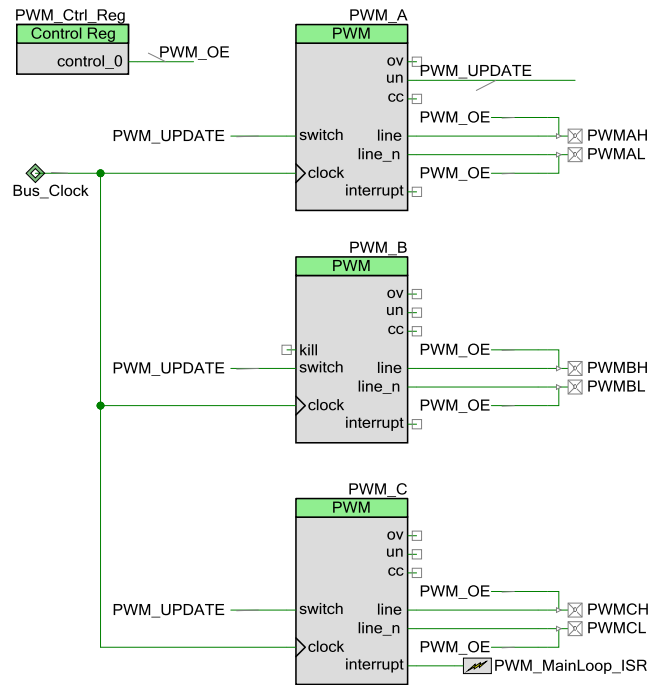


Figure 5-40 shows the ADC design in the Sensorless FOC Motor Control code example schematic. SAR ADC is triggered in every PWM cycle by the “cc” signal, and it samples for the following signals: two-phase winding currents, bus voltage, and potentiometer divider voltage. “Opamp_A” and “Opamp_B” are internal opamps to amplify and filter the motor winding currents combining the external resistors and capacitors network. PWM_D is synchronous with the carrier frequency, the set of cc point is 0.5µs advanced than un point, so that could get a max output duty.

Figure 5-40. ADC Schematic

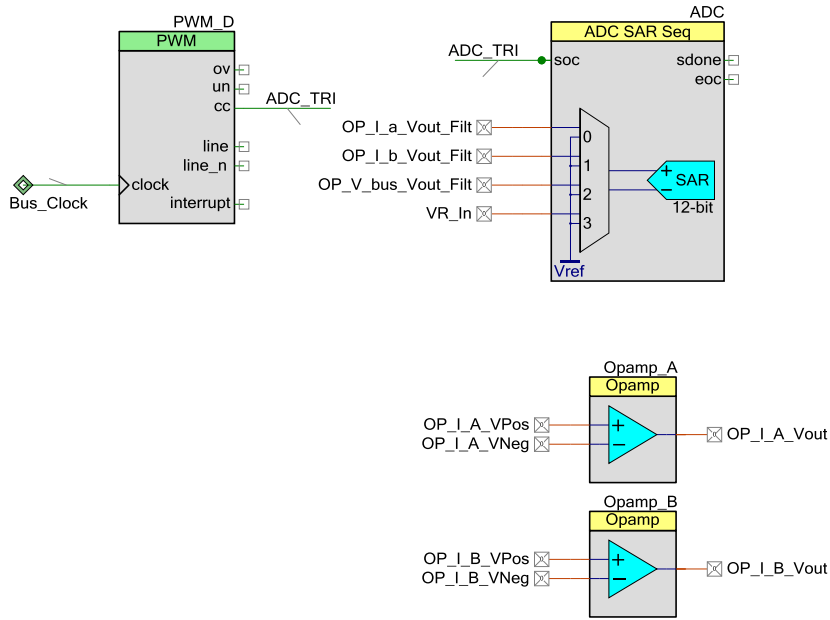
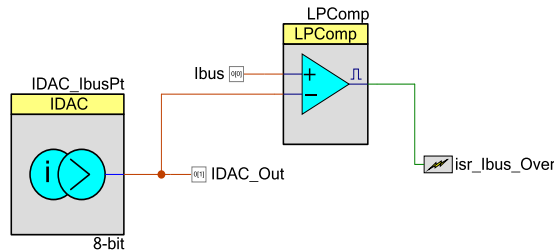


Figure 5-41 shows the Over Current Protection design in the Sensorless FOC Motor Control code example schematic. IDAC_IbusPt connects to a resistance and generates a reference voltage to LPComp.

Figure 5-41. Over Current Protection Schematic

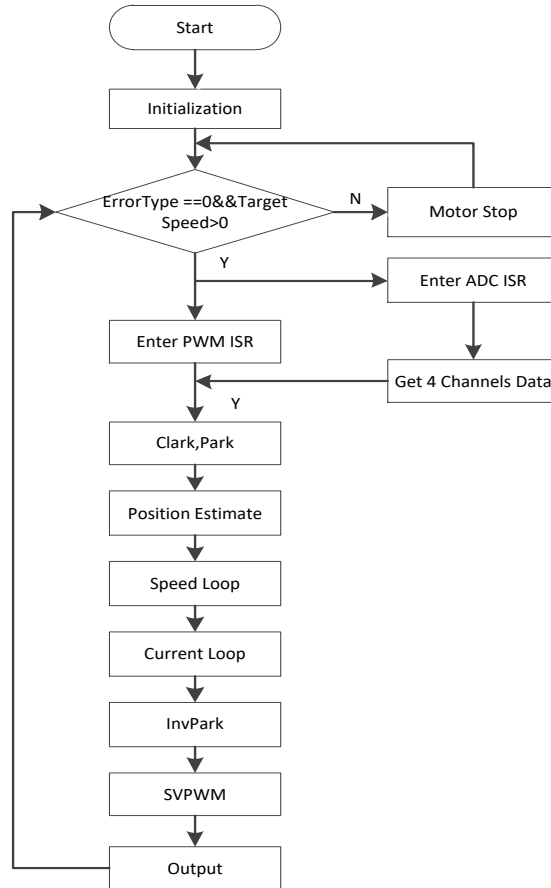


5.5.4 Firmware Introduction

Figure 5-42 gives the firmware execution flow. Most of the FOC calculation is done in an ISR triggered by the PWM UN event. As there is only one UN for each PWM period, the ISR is executed per 100µs (10-kHz PWM on the inverter), which is also the control cycle period for the PSoC 4 FOC implementation.

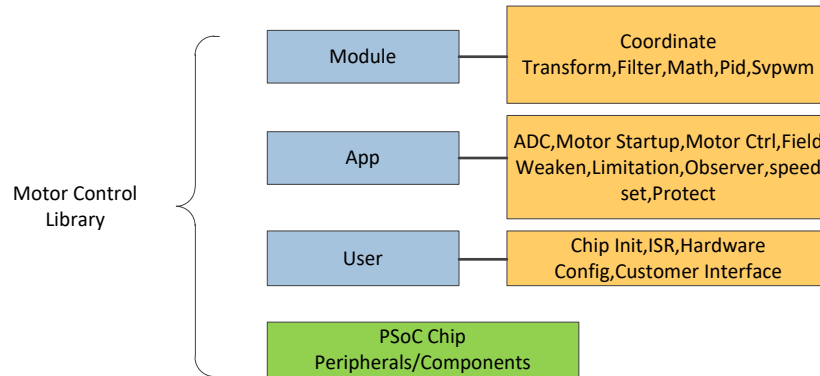
The FOC algorithm requires that the PWM duty cycle be updated for each control period, while other functions, such as communication with other systems, set speed, LED, and so on do not require real-time processing. Therefore, these functions are executed in the main routine.

Figure 5-42. Firmware Execution Flow Chart



To create a multilayer and extensible library architecture for the Cypress motor control solution, the Sensorless FOC Motor Control code example utilizes a multilayer structure. Figure 5-43 shows the layer diagram of the code example.

Figure 5-43. Sensorless FOC Motor Control Code Example Layer Diagram



A description of each layer follows.

Module Layer

- Fundamental control modules (Clarke/Park transformation and so on)
- General filters (1-order LPF, 2-order BWF, 2-order filter)
- Trigonometric and math function (sin, cos, arctan, sqrt, abs, and so on)
- Standard PID calculation
- SVPWM

App Layer

- ADC sample, offset detect.
- Motor startup flow
- Motor control flow
- Field weaken
- Limitation calculation
- Position estimate
- Basic fault detection (over current, power over range, motor lock)
- Speed set and running direction

User Layer

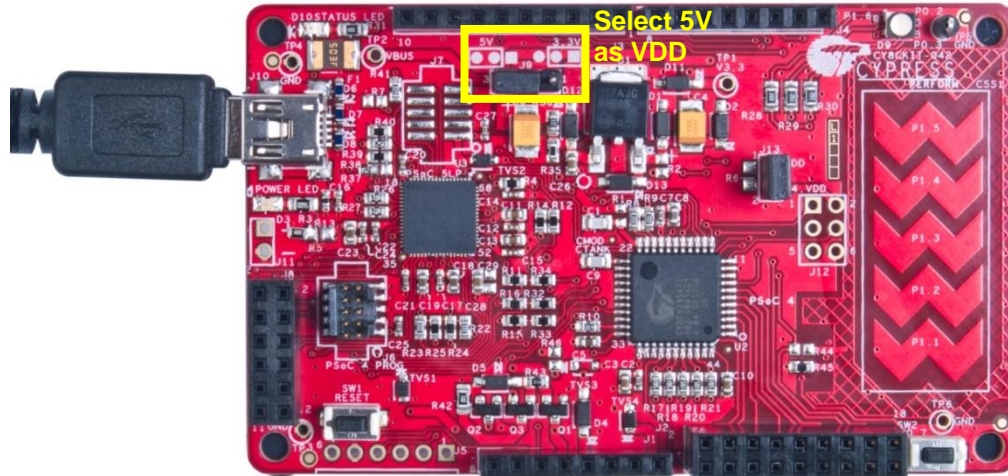
- Initialization for PWM start and stop, PWM configuration
- Interrupt entrance
- Motor running parameters set and so on.
- Definition of sample channel, ADC reference and clock frequency

5.5.5 Running the Sensorless FOC Motor Control Code Example

5.5.5.1 Step 1 – Configure CY8CKIT-042

Select 5 V as the VDD power at jumper J9 on the Pioneer Kit. Ensure to keep the USB cable connected to the Pioneer Kit because the Pioneer Kit does not provide a 5 V converter. It gets 5 V power directly from the USB port of the PC. Figure 5-44 shows the configuration.

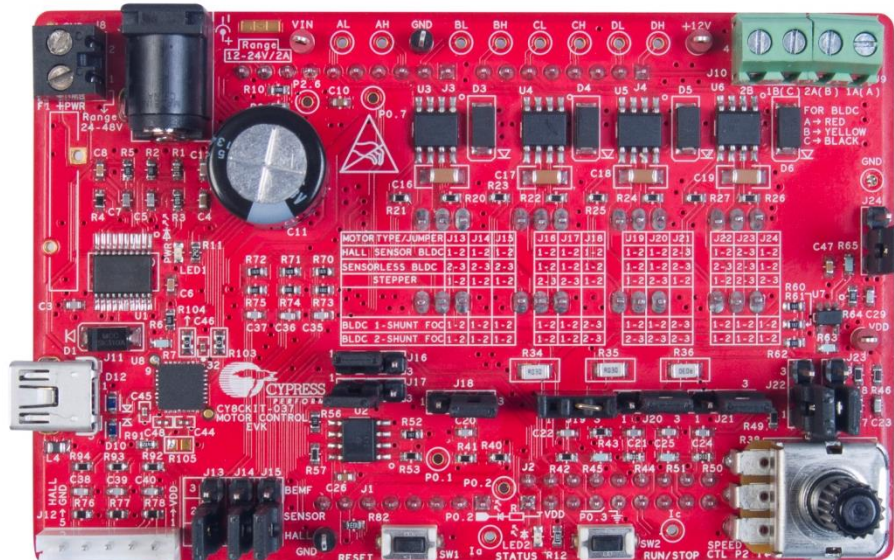
Figure 5-44. CY8CKIT-042 Configuration for Sensorless FOC Motor Control Code Example



5.5.5.2 Step 2 – Configure CY8CKIT-037

Configure the board via jumpers J13–J24 for the Sensorless FOC Motor Control code example (“BLDC 2-SHUNT FOC” row in Figure 5-6). Figure 5-45 shows the EVK board configured for the Sensorless FOC Motor Control code example.

Figure 5-45. CY8CKIT-037 Configuration for Sensorless FOC Motor Control Code Example



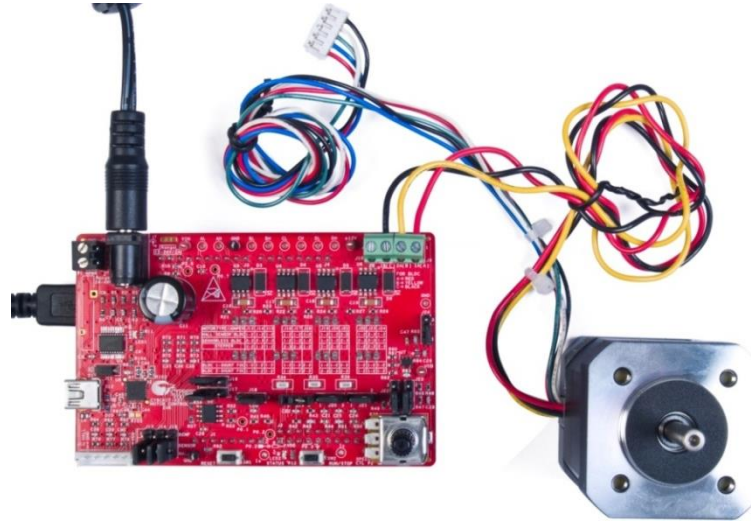
5.5.5.3 Step 3 – Plug CY8CKIT-037 into CY8CKIT-042

Plug the EVK board into the Pioneer Kit via connectors J1–J4, as shown in Figure 5-16.

5.5.5.4 Step 4 – Connect the Power Supply and Motor

Connect the BLDC motor windings to J9 and J10 on the EVK board, and then connect the 24 V power adapter to J7, as shown in [Figure 5-46](#). LED1 glows, indicating that the kit is powered. If it is not glowing, the fuse F2 may be broken. Replace F2 with the backup fuse provided with the kit.

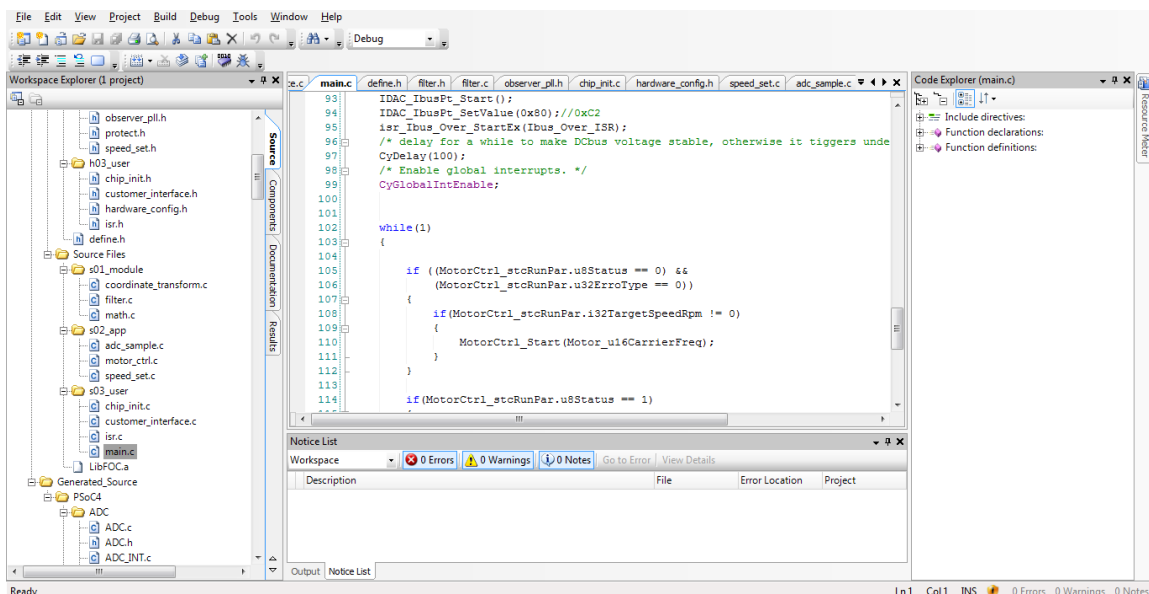
Figure 5-46. Connect Motor and Power Supply



5.5.5.5 Step 5 – Build the Project and Program PSoC 4

Open the Sensorless FOC Motor Control code example in PSoC Creator 4.2 or later version, as shown in [Figure 5-47](#). Choose **Build > Build Sensorless FOC Motor Control** to start building the project, and then choose **Debug > Program** to program the chip.

Figure 5-47. Open the Sensorless FOC Motor Control Code Example



CAUTION During the debugging process, if you modify the code in the firmware, make sure that the changes do not turn on both the high-side and low-side MOSFETs.

5.5.5.6 Step 6 – Rotate the Potentiometer R38 to Start Motor Rotation

Rotate the potentiometer R38 to change the motor rotation speed (see [Figure 5-19](#)). Press the SW2 button to set the motor running direction. If the motor does not rotate, it indicates that an error has occurred. If so, first ensure that step 1 through step 5 have been executed correctly. Then press the **Reset** button and rotate the potentiometer R38 again. If the motor still does not rotate, there must be a problem in the hardware or software. Contact [Cypress](#) for technical support.

CAUTION Do not remove jumpers while the kit is powered.

5.5.6 Adapting the Code Example to another Motor

This code example can be used to control several different types of three-phase PMSM motors, and the motor performance is closely related to the motor parameters. So the motor-related parameters in the project firmware need to be modified according to specific motor characteristics. The motor-related parameters are defined in *customer_interface.c*. [Table 5-3](#) shows the motor-related parameters.

Table 5-3. Motor-Related Parameters for Sensorless FOC Motor Control Code Example

Variable's Name in Project	Variable's Value	Comments
<code>int32_t</code> Motor_i32PolePairs	4	rotor pole pairs
<code>float32_t</code> Motor_f32Ld	0.006	motor Ld
<code>float32_t</code> Motor_f32Lq	0.006	motor Lq
<code>float32_t</code> Motor_f32Res	1.1	motor resistance between two phases
<code>float32_t</code> Motor_f32Ke	2.83	inductive voltage constant between two phases

You can modify these parameters in *customer_interface.c*.

[Table 5-4](#) shows the hardware-related parameters, PID parameters and motor startup parameters.

Table 5-4. Control-Related Parameters

Variable's Name in Project	Variable's Value	Comments
<code>float32_t</code> Motor_f32IuvwSampleResistor	0.03	Iuvw sample resistor
<code>float32_t</code> Motor_i32IuvwAmplifierFactor	4.16	Iuvw calculation factor
<code>int32_t</code> Motor_i32IuvwOffsetNormal	2048	the middle value of 12-bits ADC
<code>int32_t</code> Motor_i32IuvwOffsetRange	150	ADC offset range of Iuvw sampling
<code>int32_t</code> Motor_i32IuvwOffsetCheckTimes	64	Iuvw ADC sample offset check times
<code>float32_t</code> Motor_f32DeadTimeMicroSec	0.8f	Dead timer us
<code>uint16_t</code> Motor_u16CarrierFreq	10000	motor carry frequency
<code>float32_t</code> Motor_f32Dki	0.04f	d axis current PI regulator integral constant
<code>float32_t</code> Motor_f32Dkp	0.50f	d axis current PI regulator proportion constant
<code>float32_t</code> Motor_f32Qki	0.04f	q axis current PI regulator integral constant
<code>float32_t</code> Motor_f32Qkp	0.50f	q axis current PI regulator proportion constant
<code>float32_t</code> Motor_f32LowSpdKi	0.005f	Low speed PI regulator integral constant
<code>float32_t</code> Motor_f32LowSpdKp	0.25f	Low speed PI regulator proportion constant
<code>float32_t</code> Motor_f32Ski	0.005f	High speed PI regulator integral constant

Variable's Name in Project	Variable's Value	Comments
<code>float32_t</code> Motor_f32Skp	0.25f	High speed PI regulator proportion constant
<code>uint16_t</code> Motor_u16ChgPiSpdHz	30	PID parameters change at this speed
<code>uint8_t</code> Motor_u8RunLevel	4	1->orientation, 2->open loop running, 3->closed loop running, 4->change speed enable
<code>int16_t</code> Motor_i16Q8_OrientEndIqRef	Q8 (0.5)	orientation end current
<code>int16_t</code> Motor_i16Q8_OrientInitIqRef	Q8 (0)	orientation initial current
<code>float32_t</code> Motor_f32OrientIqrefIncAPS	1.0f	orientation current added per sec
<code>float32_t</code> Motor_f32OrientTime	0.0f	orientation time,Second(if 0,skip this step)
<code>uint16_t</code> Motor_u16OpenLoopSpdInitHz	3	force start initial speed
<code>uint16_t</code> Motor_u16OpenLoopSpdEndHz	3	force start end speed(if init= end, skip this step)
<code>uint16_t</code> Motor_u16OpenLoopSpdIncHz	3	force start speed added per sec
<code>int16_t</code> Motor_i16Q8_OpenLoopInitIqRef	Q8 (1.0)	force start q axis initial current reference
<code>int16_t</code> Motor_i16Q8_OpenLoopEndIqRef	Q8 (2.0)	force start q axis current reference
<code>float32_t</code> Motor_f32OpenLoopIqrefIncAPS	2.0f	force start current added per sec

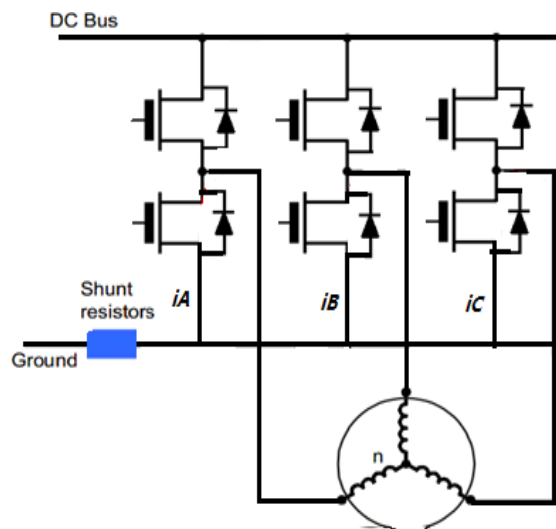
5.6 Single Shunt FOC Motor Control Code Example

5.6.1 Single Shunt FOC Background

The two-shunt sensorless FOC in the [Sensorless FOC Motor Control Code Example](#) gets the motor winding current and estimates the rotor position from two sensing resistors directly in series with two coil current paths. The single-shunt FOC technique uses only one sensing resistor to get the bus current and reconstruct the two-phase motor winding current. It then estimates the rotor position according to the reconstructed winding current. Compared with the two-shunt FOC technique, it saves a sensing resistor and a differential amplifier and so reduces the cost.

Figure 5-48 shows the current measurement by the single shunt.

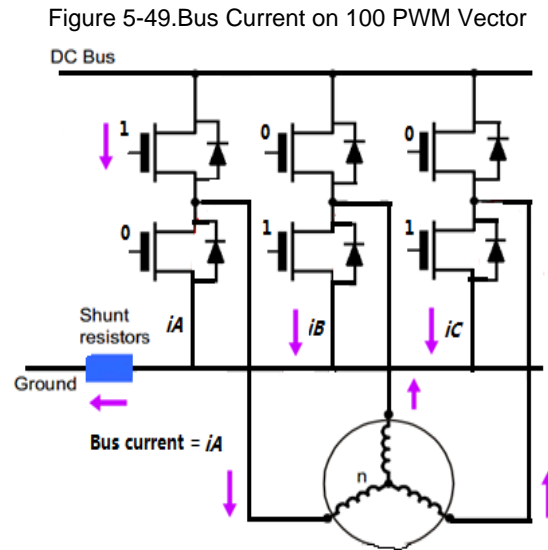
Figure 5-48. Single-Shunt FOC Current Measurement



The disadvantages of single-shunt FOC are obvious. You need to reconstruct the three-phase motor winding current from the bus current, so the sinusoidal modulation pattern needs to be modified to allow the bus current to be measured. This makes the PWM signal generation algorithm more complicated than in the two-shunt Sensorless FOC Motor Control code example.

As Figure 5-49 shows, there are eight possible combinations of ON and OFF states of PWM. Each state has a different bus current. When the PWM state is zero vector (000 or 111), the bus current is zero. For the other six states, the following discussion uses 100 vector to explain the winding current reconstruction.

When PWMA is ON, PWMB and PWMC are OFF (100 state), and current flows from phase A into phases B and C. At this time, the bus current is i_A , as Figure 5-49 shows.



In the same way, you can get the bus current of other PWM states. Table 5-5 shows the relationship between the bus current and the PWM state.

Table 5-5. Relationship Between Bus Current and PWM State

1H	1L	2H	2L	3H	3L	iBus
1	0	0	1	0	1	i_A
0	1	1	0	0	1	i_B
0	1	0	1	1	0	i_C
0	1	1	0	1	0	$-i_A$
1	0	0	1	1	0	$-i_B$
1	0	1	0	0	1	$-i_C$

So, if you sample bus current twice in one PWM period in a different PWM state, you can get two-phase currents. Then you can reconstruct the third one.

5.6.2 Single Shunt FOC Motor Control Code Example Overview

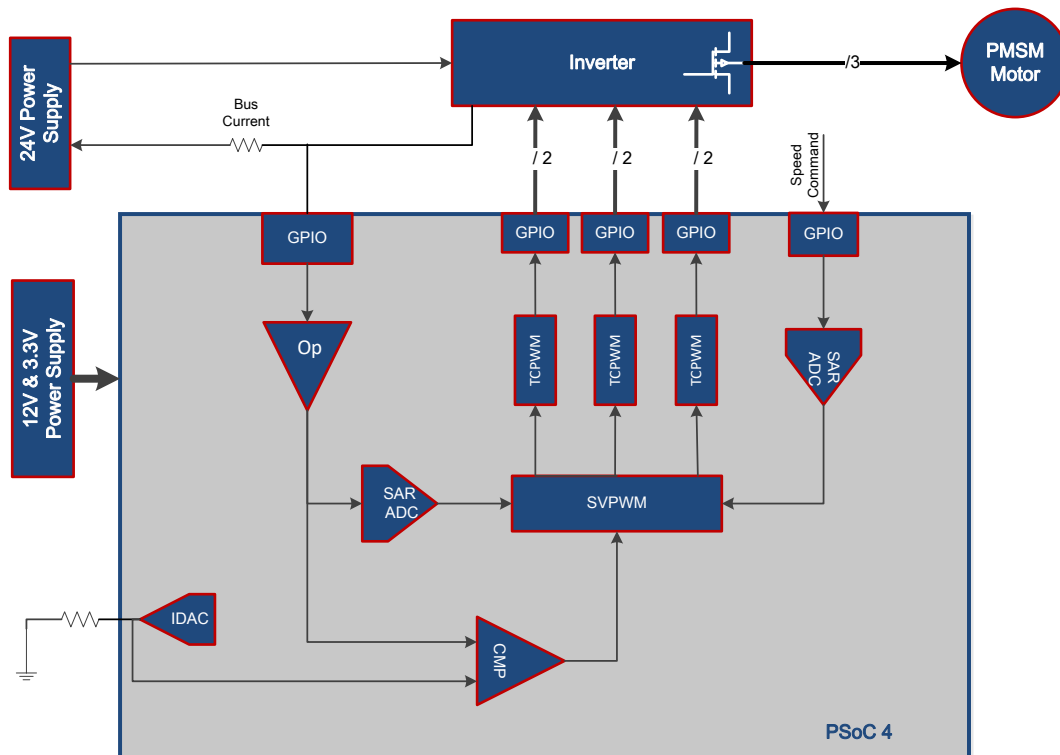
Figure 5-50 illustrates the block diagram of the Single Shunt FOC Motor Control code example based on PSoC 4. The input control signals to the PSoC 4 device are the following.

- **Speed command:** Analog input pin that measures the voltage across a potentiometer to set the desired speed of rotation (one analog input pin)
- **Motor current detection:** Analog input pin to detect and cut off power to the device driver to protect the motor when an overcurrent condition is detected (see [Control Schematic Overview](#))
- **Vbus:** Analog input pin that routes bus voltage to the SAR ADC to monitor the bus voltage
- **Start/stop control:** Digital input connected to a switch to start and stop the motor rotation (one digital input pin)

Outputs from the PSoC 4 device are power device driver signals.

- PWM signals to the high side of the MOSFET driver (three digital output pins)
- PWM signals to the low side of the MOSFET driver (three digital output pins)
- Overcurrent limit voltage set by the internal IDAC and an outside resistor

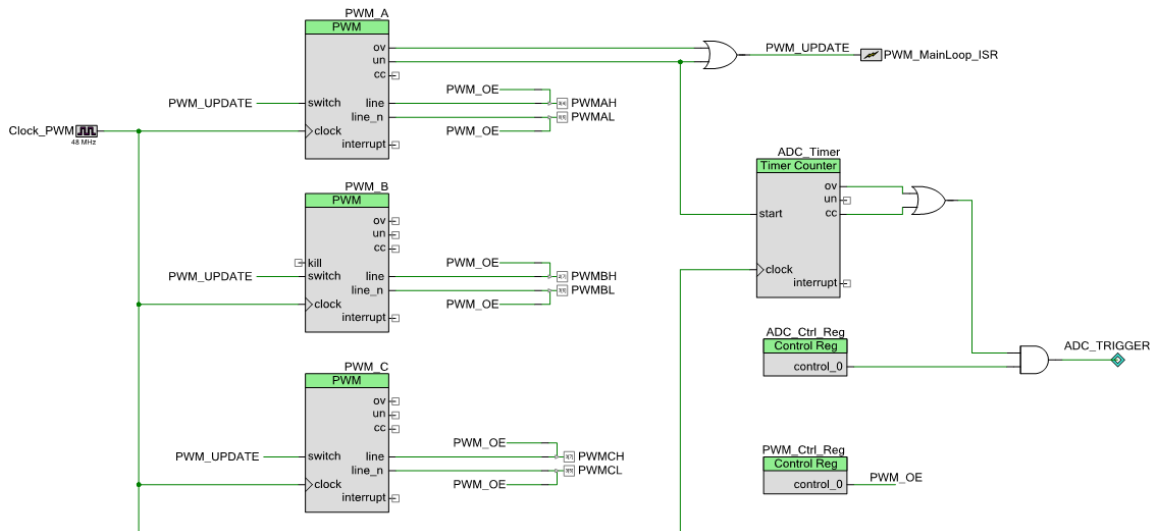
Figure 5-50. SingleShunt Sensorless FOC Motor Control Code Example Block Diagram



5.6.3 Control Schematic Overview

The Single Shunt Foc Motor Control code example firmware is developed in PSoC Creator 4.2. Its schematic is separated into two pages according to function. Figure 5-51 shows the “PWM Drive” schematic in the “FOC” page. The three TCPWM Components generate three couples of the center-aligned and complemented SVPWM signal according to the FOC algorithm.

Figure 5-51. PWM Drive Schematic



By comparing Figure 5-52 with Figure 5-39, you can see that the PWM duty cycle update and ADC trigger mechanism are different from the Sensorless FOC Motor Control code example because the current sensing and reconstructing method has been changed in the Single Shunt FOC Motor Control code example.

Figure 5-52 shows the ADC trigger timing. To avoid impacting the current pulse when the PWM state changes, ADC sampling is triggered in the middle of two PWM states. So the schematic shows the use of a PWM underflow signal (“un”) as a trigger source and then the use of a timer (“ADC_Timer”) to create a delay to ensure that the ADC is triggered in the middle of T_1 and T_2 . The compare value of “ADC_Timer” is updated in real time while the motor is running. In the PWM ov interrupt ISR, the ADC trigger delay should be set as $T_1/2$. After the ADC is triggered, the ADC trigger delay should be set as $T_2/2$ in the ADC eoc interrupt ISR.

So, to change the compare value twice in one PWM period, you need two TC events. Then the PWM switch signal is triggered by “PWM_UPDATE” on both the OV and UN events. The PWM compare value will be updated twice in one period. The PWM ISR should also be triggered twice on the OV and UN events.

Figure 5-52. ADC Trigger Timing

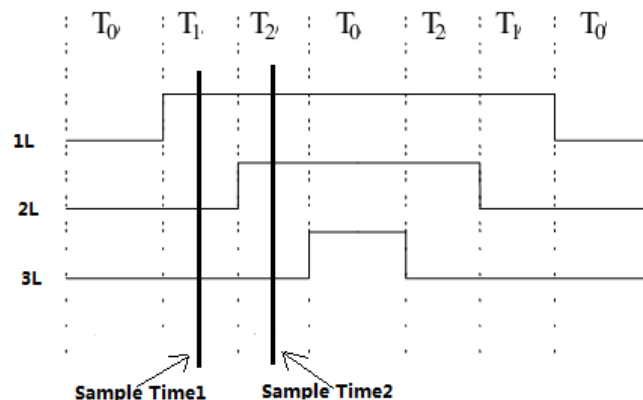


Figure 5-53 shows the “Phase Current Sampling” schematic in the “FOC” page. SAR ADC is triggered twice in every PWM cycle by the “ov” and “cc” signals, and it samples three signals: bus current, bus voltage, and potentiometer divider voltage. “Opamp_A” is the internal opamp to amplify and filter the bus current combining the external resistors and capacitors network.

Figure 5-53. Phase Current Sampling Schematic

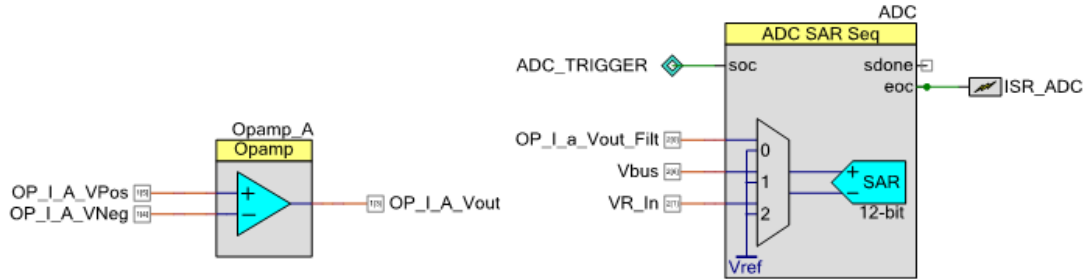
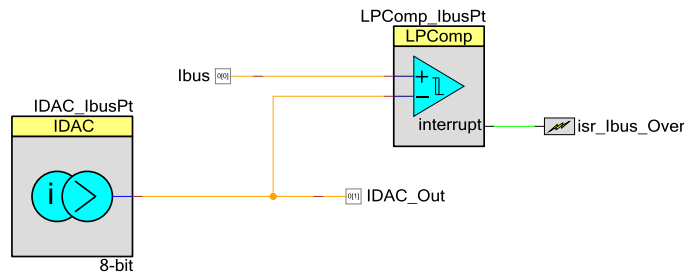


Figure 5-54 shows the “Overcurrent Protection” schematic. The bus current is measured by an external opamp and routed to the positive terminal of the internal low-power comparator “LPComp_IbusPt”. The overcurrent threshold is set by an internal IDAC, “IDAC_IbusPt”, and routed to the negative terminal of “LPComp_IbusPt”. When an overcurrent condition occurs, “LPComp_IbusPt” will generate an interrupt to disable the PWM output.

Figure 5-54. Overcurrent Protection Schematic



5.6.4 Firmware Introduction

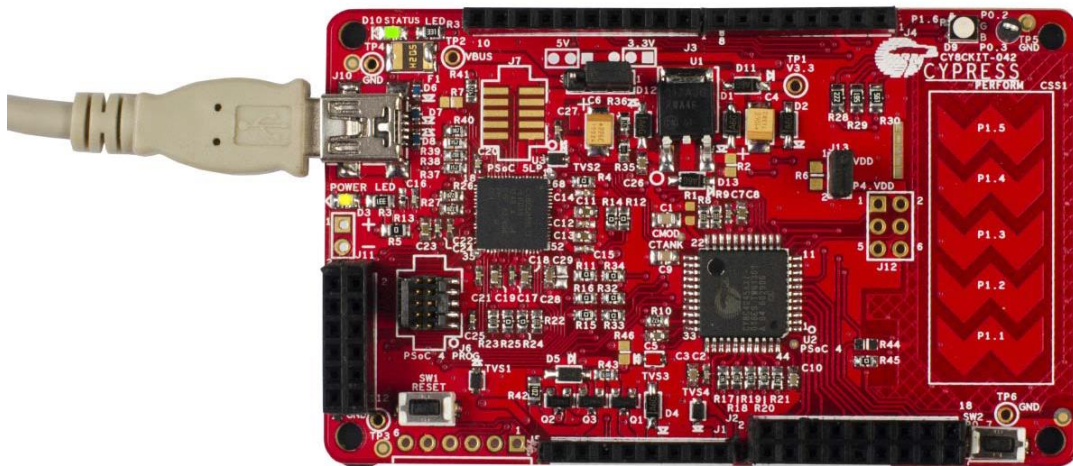
Except for the motor current sensing and reconstruction mechanism introduced in [Control Schematic Overview](#), the control firmware is similar to the Sensorless FOC Motor Control code example. Refer to [Firmware Introduction](#) to learn about the firmware of the Single Shunt FOC Motor Control code example.

5.6.5 Running the Single Shunt FOC Motor Control Code Example

5.6.5.1 Step 1 – Configure CY8CKIT-042

Select 3.3 V as the VDD power at jumper J9 on the Pioneer Kit. In this application, the USB cable is used only for burning firmware, so it can be removed from CY8CKIT-042 after programming. [Figure 5-55](#) shows the CY8CKIT-042 configuration.

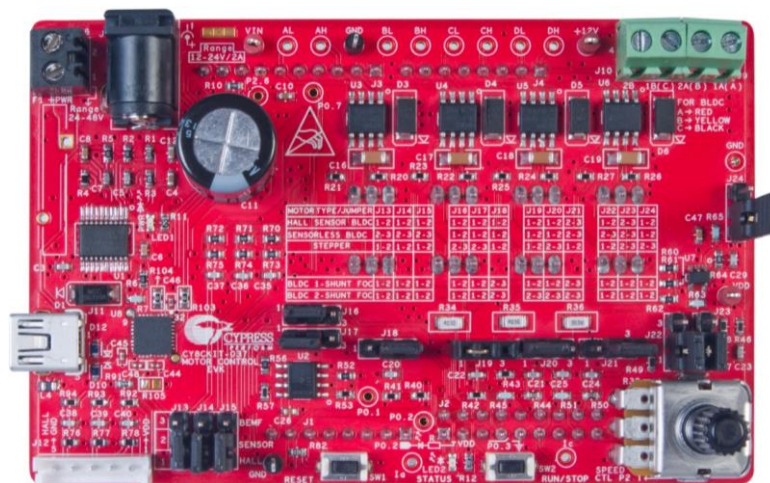
Figure 5-55. CY8CKIT-042 Configuration



5.6.5.2 Step 2 – Configure CY8CKIT-037

Configure the board via jumpers J13–J24 for the Single Shunt FOC Motor Control code example (“BLDC 1-SHUNT FOC” row in [Figure 5-6](#)). [Figure 5-56](#) shows the EVK board configured for the SingleShunt FOC Motor Control code example.

Figure 5-56. CY8CKIT-037 Configuration for SingleShunt Foc Motor Control Code Example



5.6.5.3 Step 3 – Plug CY8CKIT-037 into CY8CKIT-042

Plug the EVK board into the Pioneer Kit via connectors J1–J4, as shown in [Figure 5-16](#).

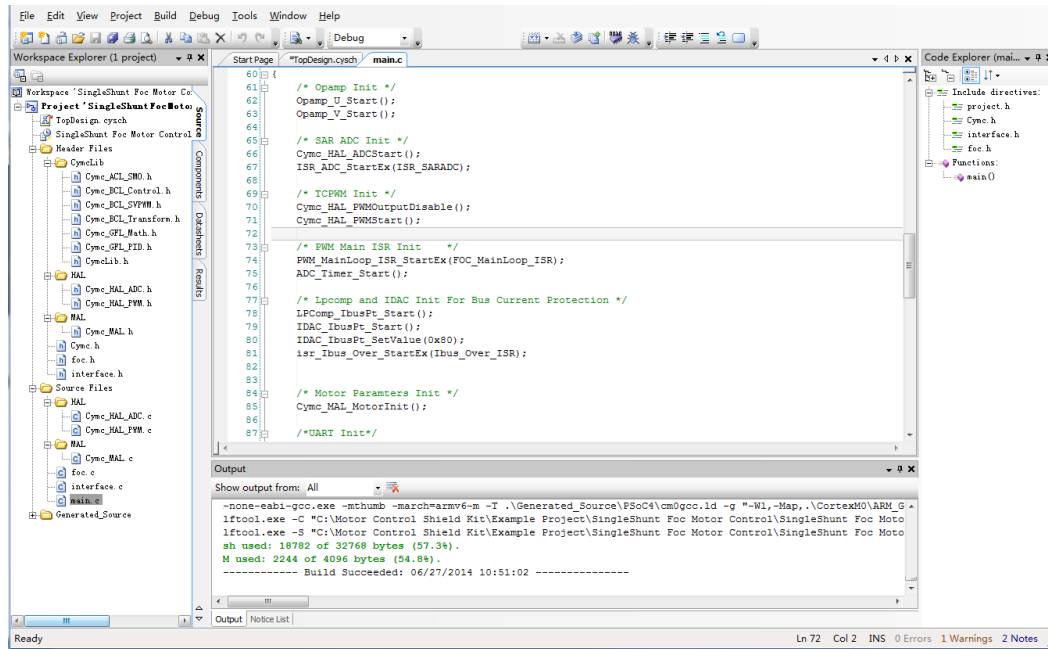
5.6.5.4 Step 4 – Connect the Power Supply and Motor

Connect the BLDC motor windings to J9 and J10 on the EVK board, and then connect the 24 V power adapter to J7, as shown in [Figure 5-46](#). LED1 glows, indicating that the kit is powered. If it is not glowing, the fuse F2 may be broken. Replace F2 with the backup fuse provided with the kit.

5.6.5.5 Step 5 – Build the Project and Program PSoC 4

Open the Single Shunt FOC Motor Control code example in PSoC Creator 4.2 or later version, as shown in [Figure 5-57](#). Choose **Build > Build SingleShunt Foc Motor Control** to build the project, and then choose **Debug > Program** to program the chip.

Figure 5-57. Open the SingleShunt Foc Motor Control Code Example



CAUTION During the debugging process, if you modify the code in the firmware, ensure that the changes do not turn on both the high-side and low-side MOSFETs.

5.6.5.6 Step 6 – Press the SW2 Button to Start Motor Rotation

Press the **SW2** button to start motor rotation (see [Figure 5-19](#)). If the motor does not rotate and you observe LED2 blinking, it indicates that an error has occurred. If so, first ensure that step 1 through step 5 have been executed correctly. Then press the **Reset** button and press the **SW2** button again. If LED2 still blinks, there must be a problem in the hardware or software. Contact [Cypress](#) for technical support.

CAUTION Do not remove jumpers while the kit is powered.

5.6.6 Adapting the Code Example to another Motor

Refer to [Adapting the Code Example to another Motor](#).

5.7 Stepper Motor Control Code Example

CY8CKIT-037 does not provide a stepper motor in the kit package. However, the kit does support a stepper motor controlled with microstep from hardware and firmware. A Stepper Motor Control code example is included in the kit installation directory. To demonstrate this project, you need to connect your stepper motor to the CY8CKIT-037 kit. The only requirement for the motor is that the power range should meet the kit specifications.

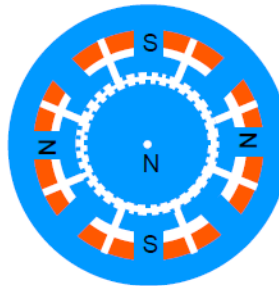
5.7.1 Stepper Motor Background

The stepper motor is an electromechanical device that converts electrical pulses into discrete mechanical movement. It can achieve accurate positioning and potential rotation speed control without a feedback sensor. So it is ideally suited for many measurement and control applications where positional accuracy is important and cost is very low.

The most frequently used stepper motor is the hybrid stepper motor. It has smaller steps, greater torque, and greater maximum speeds than a variable reluctance and permanent magnet stepper motor. Figure 5-58 shows a sectional view of the hybrid stepper motor. Both the rotor and stator have teeth, providing a smaller magnetic circuit resistance in some rotor positions, which improves static and dynamic torque.

Stepper motors have several operational modes, such as full-step mode, half-step mode, and microstep mode. Each mode controls stepper motor phases in different ways. The full-step and half-step modes have obvious torque rippling and resonance. Microstep mode changes the current by small steps that split each step into microsteps. When two phases are turned on and the current on each phase is not equal, the current phase ratio determines the rotor position. Changing the ratio produces a number of microsteps inside each full step.

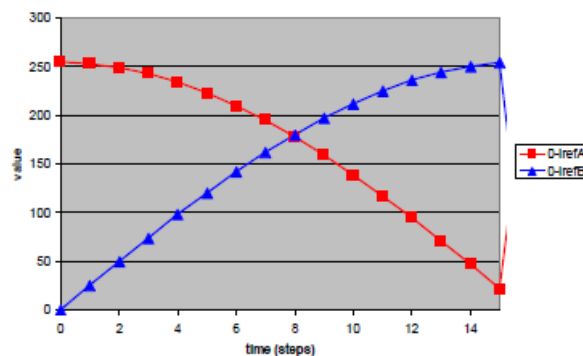
Figure 5-58. Hybrid Stepper Motor



Fractional steps are created by scaling torque contributions between windings. Because torque is proportional to the magnetic flux that is proportional to the current in the winding, the rotor's position can be controlled by controlling the current flowing in each winding. To create smooth microsteps between full steps, the current is varied sinusoidally with a 90° phase shift between the two windings, as shown in Figure 5-59.

The current is scaled by controlling the RMS current using a current-mode buck converter, commonly called a “chopper drive” when used with stepper motors. The phase current is converted into a voltage using a sensing resistor in each phase ground path. This voltage is routed to a comparator, which disables the output whenever the phase current rises above a reference. The comparator reference is provided by an IDAC. By changing the IDAC-supplied current limit for each microstep, the total motor torque remains approximately constant for each step of the sinusoidal current waveform.

Figure 5-59. VDAC Current Limit for Microstep Mode



5.7.2 Stepper Motor Control Code Example Overview

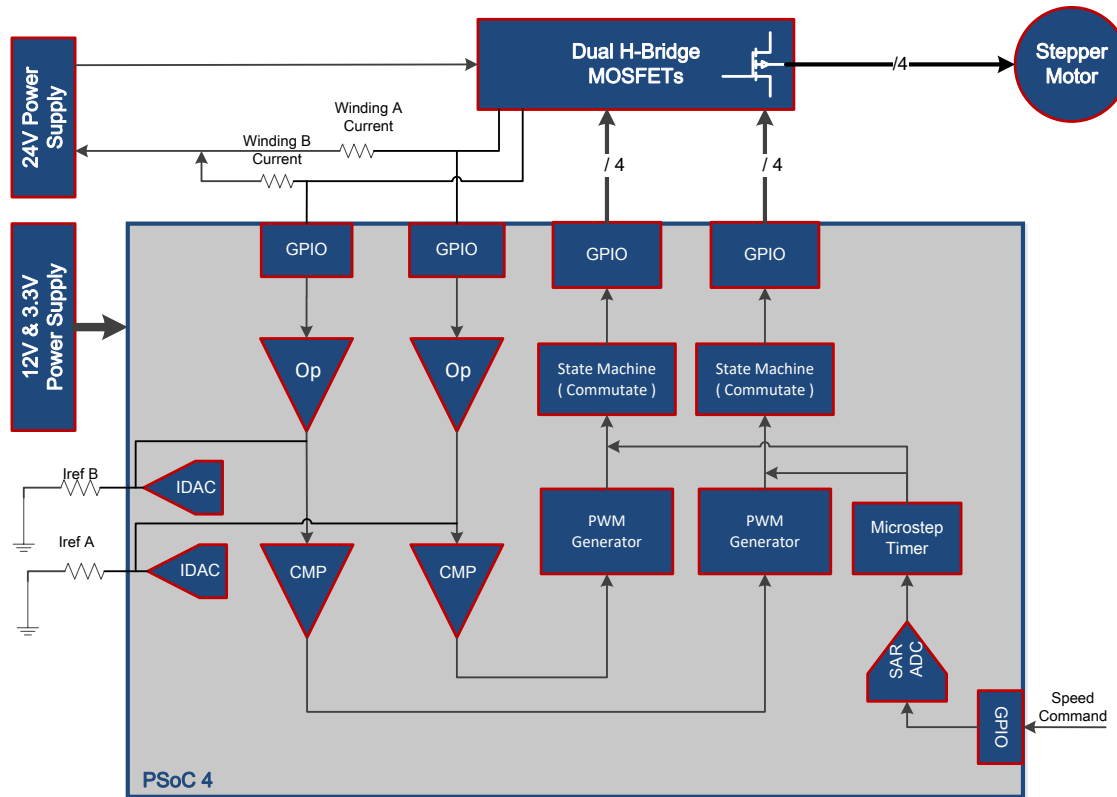
Figure 5-60 illustrates the block diagram of the Stepper Motor Control code example based on PSoC 4. The input control signals to the PSoC 4 device are as follows:

- **Speed command:** An analog input pin that measures the voltage across a potentiometer to set the desired speed of rotation (one analog input pin).
- **Motor current detection:** Two analog input pins to detect the motor winding current by sensing resistors. Then the signal is routed to internal opamps to be amplified.
- **PWM kill:** Two digital input pins that are the output of external comparators. A high level of this signal will stop internal PWM output and force the winding current to decrease to the current reference set by the IDAC.
- **Start/lock control:** A digital input connected to a switch to start and lock rotation of the motor (one digital input pin).

Outputs from the PSoC 4 device are power device driver signals.

- PWM signals to the high side of the MOSFET driver (four digital output pins).
- PWM signals to the low side of the MOSFET driver (four digital output pins).
- A winding current signal amplified by internal opamps. The signal is routed to positive external comparators and is compared with the current reference set by the IDAC to construct the winding current chopper circuit.
- The sinusoidal current reference voltage set by the internal IDAC and outside resistors.

Figure 5-60. Stepper Motor Control Code Example Block Diagram



5.7.3 Control Schematic Overview

The Stepper Motor Control code example firmware is developed in PSoC Creator 4.2. Its schematic is separated into four pages according to function. Figure 5-61 shows the “PWM and Commutation” page. When the motor phase current is more than the sinusoidal reference current level, the chopper circuit will output a high level. This high level closes the PWM output and synchronizes the real motor phase current with the sinusoidal reference current level. “Control_Reg_1” indicates the current stage of the stepper motor. It is dynamically modified by firmware to commutate two phase windings independently through “LUT_A” and “LUT_B.”

Figure 5-61. PWM and Commutation Schematic

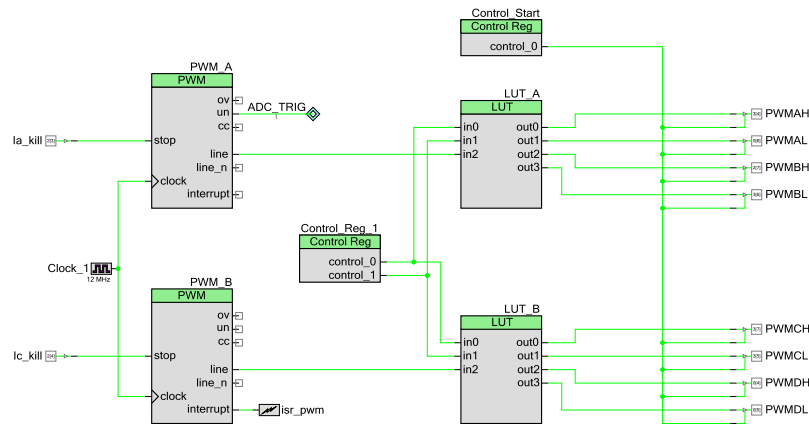


Figure 5-62 shows the “Current Chopping” page. Internal IDACs generate the sinusoidal reference, employing the internal opamps circuit sensing, and amplify the winding current. Then the output of the opamps circuit is compared with the sinusoidal reference current level for each phase in every microstep by external comparators. When the motor phase current is more than the sinusoidal reference current level, the comparator will output a high level.

Figure 5-62. Current Chopping Schematic

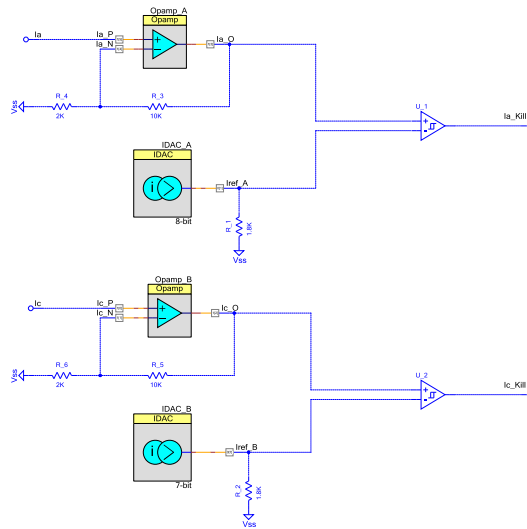
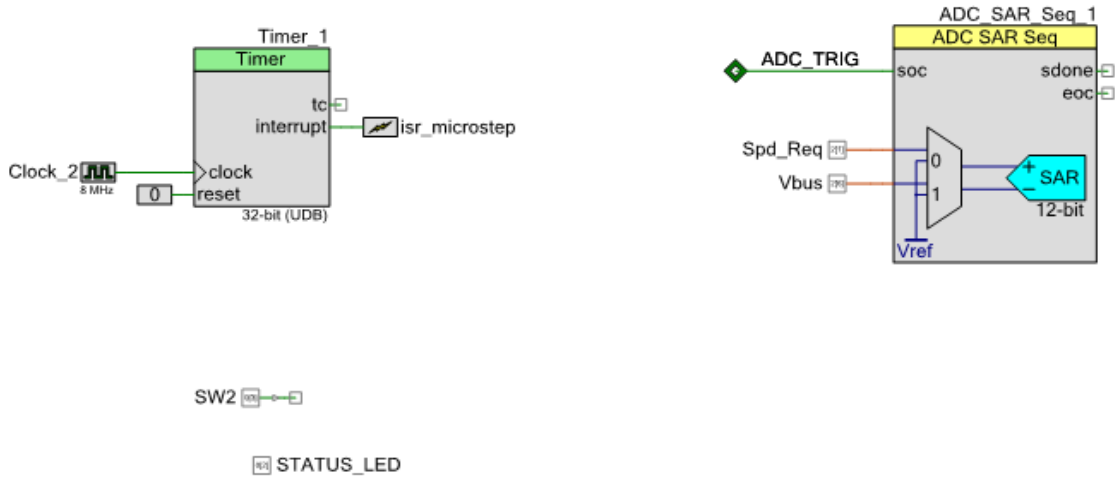


Figure 5-63 shows the “ADC, Button and Microstep Timer” page. “Timer_1” sets the period of the microsteps. Its overflow event triggers an interrupt, and winding commutating and sinusoidal current reference value updating are executed in its ISR. “ADC_SAR_Seq_1” is the SAR ADC to detect and measure bus voltage and potentiometer voltage.

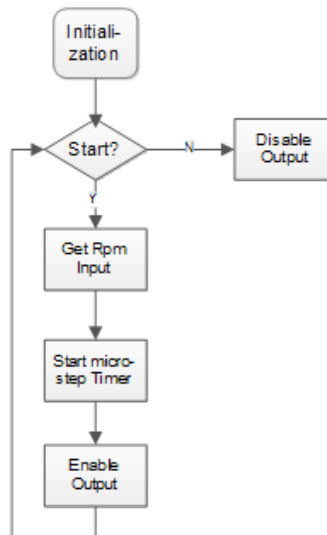
Figure 5-63. ADC, Button, and Microstep Timer Schematic



5.7.4 Firmware Introduction

Figure 5-64 shows the flow chart of the main loop function. The program first initializes and configures the internal resources of PSoC 4, and then the main loop detects the user’s start-stop command and RPM reference.

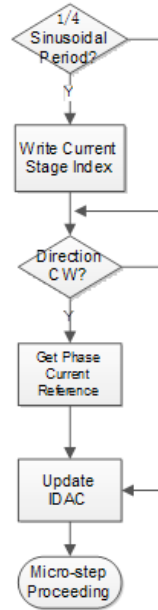
Figure 5-64. Main Loop Function Flow Chart



The microstep drive algorithm is processed in the timer overflow ISR in which the microstep length was set.

Figure 5-65 shows the flow chart of the timer overflow ISR.

Figure 5-65. Timer Overflow ISR Flow Chart



5.7.5 Running the Stepper Motor Control Code Example

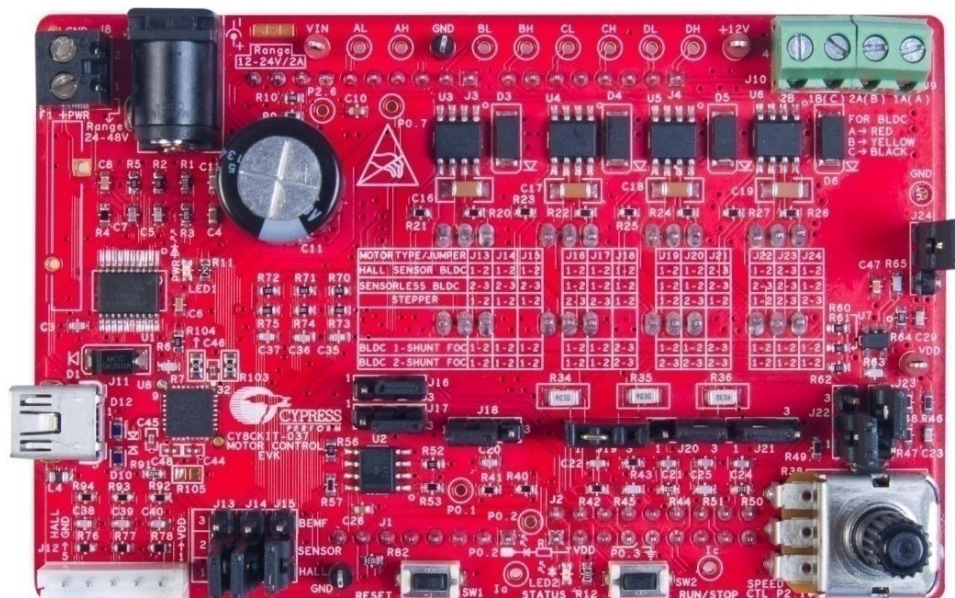
5.7.5.1 Step 1 – Configure CY8CKIT-042

Select 3.3V as VDD at jumper J9 on the Pioneer Kit. In this application, the USB cable is used only for burning firmware, so it can be removed from CY8CKIT-042 after programming. Figure 5-55 shows the CY8CKIT-042 configuration.

5.7.5.2 Step 2 – Configure CY8CKIT-037

Configure the board via jumpers J13–J24 for the Stepper Motor Control code example (“STEPPER” row in Figure 5-6). Figure 5-66 shows the EVK board configured for the Stepper Motor Control code example.

Figure 5-66. CY8CKIT-037 Configuration for Stepper Motor Control Code Example



5.7.5.3 Step 3 – Plug CY8CKIT-037 into CY8CKIT-042

Plug the EVK board into the Pioneer Kit via connectors J1–J4, as shown in [Figure 5-16](#).

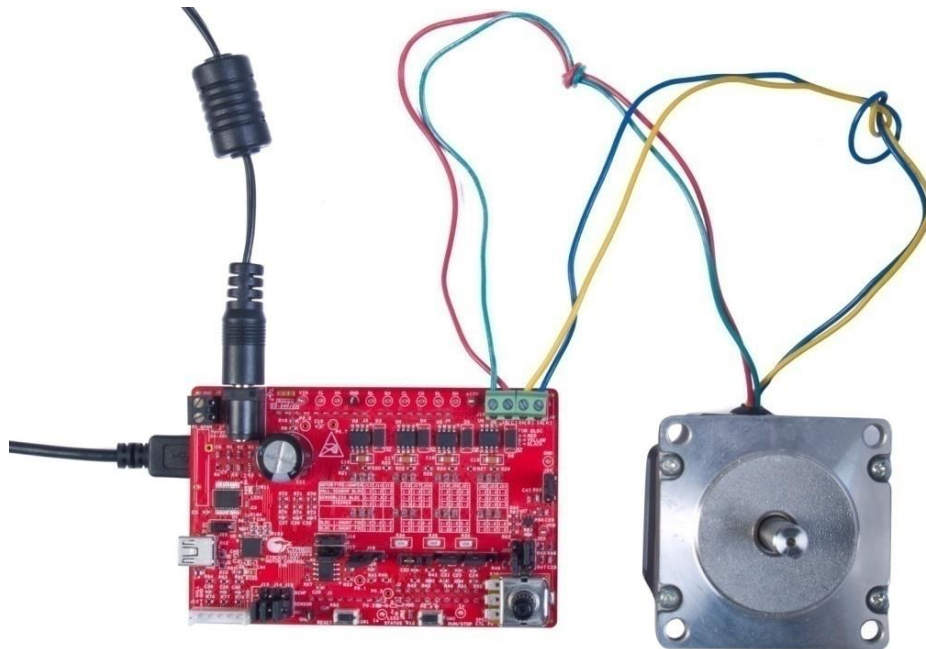
5.7.5.4 Step 4 – Connect the Power Supply and Motor

Connect the stepper motor windings to J9 and J10 on the EVK board, as shown in [Figure 5-67](#). The wire sequence and color for the motor windings should be exactly as shown in [Figure 5-67](#). Refer to [Motor Winding Connectors](#) for instructions. Then connect the 24-V adapter to J7. LED1 glows indicating that the kit is powered. If it is not glowing, the fuse F2 may be broken. Replace F2 with the backup fuse provided with the kit.

CAUTION

Note: The CY8CKIT-037 PSoC 4 Motor Control EVK package does not include a stepper motor. Cypress recommends that you use a 42-mm (diameter or side length) stepper motor: 42BYGH403AA (with 1.8-degree step angle and 1.65-A phase current). You can also find a stepper motor from other manufacturers with the same size and electrical parameters.

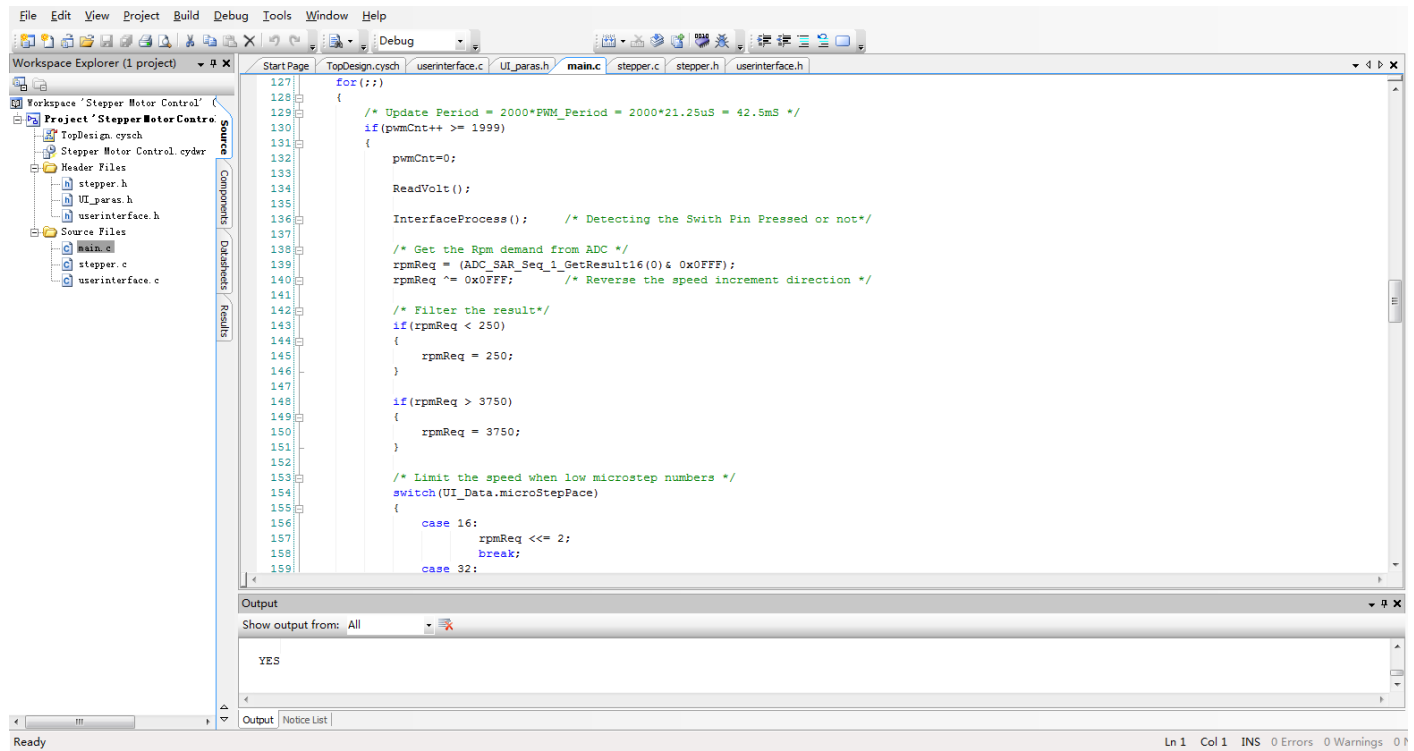
Figure 5-67. Connect Motor and Power Supply



5.7.5.5 Step 5 – Build the Project and Program PSoC 4

Open the Stepper Motor Control code example in PSoC Creator 4.2 or later version, as shown in [Figure 5-68](#). Choose **Build > Build Stepper Motor Control** to build the project, and then choose **Debug > Program** to program the chip.

Figure 5-68. Open the Stepper Motor Control Code Example



CAUTION During the debugging process, if you modify the code in the firmware, ensure that the changes do not turn on both the high-side and low-side MOSFETs.

5.7.5.6 Step 6 – Press the SW2 Button to Start Motor Rotation

Press the **SW2** button to start motor rotation (see [Figure 5-19](#)). If the motor does not rotate and you observe LED2 blinking, it indicates that an error has occurred. If so, first ensure that step 1 through step 5 have been executed correctly. Then, press the **Reset** button and press the **SW2** button again. If LED2 still blinks, there must be a problem in the hardware or software. Contact [Cypress](#) for technical support.

CAUTION Do not remove jumpers while the kit is powered.

5.7.6 Adapting the Code Example to another Motor

This code example can be used to control two phases of the HB (hybrid) stepper motor with microstep algorithm. You need to modify the motor-related parameters in the project firmware according to specific motor characteristics. You can also modify parameters related to function, like microstep numbers, RPM, and current level, according to your requirements.

The main parameters are defined by a `struct` in `UI paras.h` as follows.

```
typedef struct
{
    uint8    dir;                /*Direction*/
    uint8    microStepPace;     /*For Stepper Motor*/
}UI_DATA;
```

The real `struct` variable is defined in `main.c`.

```
UI_DATA UI_Data;
```

This `struct` variable is initialized in `main.c` by a function as follows:

```
void MotorInit(void)
{
    Control_Start_Write(0);
    UI_Cmd.run = FALSE;
    UI_Data.microStepPace = 128; /* Set the number of micro-step pace: microstep number =
                                128 / UI_Data.microStepPace */
    UI_Data.dir = CCW;          /* Set rotating direction of motor */
}
```

This function initializes the parameters before the motor begins running. The initializing value is dedicated for the 42-mm (diameter or side length) stepper motor (42BYGH403AA). If you want to use your own motor and have different functional requirements like microstep numbers or direction, change the initializing value in this function.

Note: The number of microsteps is equal to “128/UI_Data.microStepPace.” So, if you set `UI_Data.microStepPace = 128`, there is only one microstep, which means no microstep pace. So the stepper motor generates high mechanical vibration, torque ripple, and noise and may stop abruptly at some mechanical resonance points when the potentiometer is varied. So it is recommended that you set “UI_Data.microStepPace = 1, 2, 4, 8, 16, or 32” to enable the microstep algorithm to make the motor run more stably and smoothly.

5.8 Bridge Control Panel Monitor Tool Guide

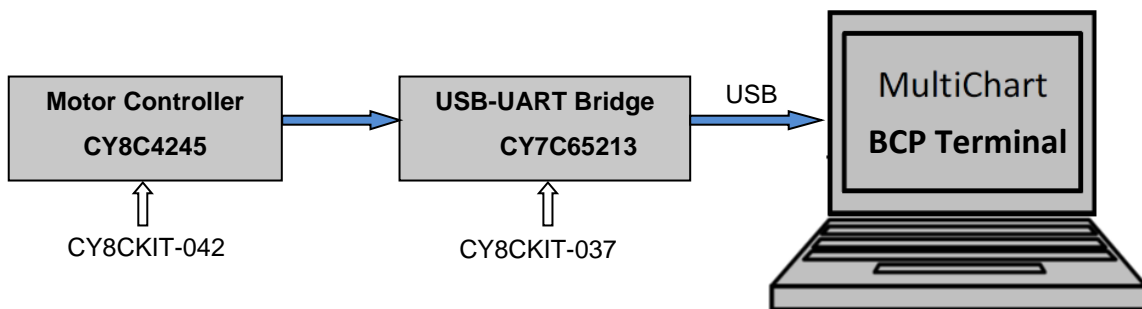
During the motor control debugging process, you cannot use the step debug because it prevents the CPU from running at breakpoint, which leads to blockage of the motor rotation and generates a very high current that burns the motor windings. Therefore, you need to use the real-time debugging tool to monitor the status of the running motor.

The BCP is a very useful and convenient real-time debugging tool provided by Cypress to monitor the motor state in the debugging process. This section introduces how to configure and use the BCP in real-time debugging for the motor control application.

5.8.1 BCP Monitoring Overview

Figure 5-69 shows the system structure of real-time debugging for the motor control application. The controller board (CY8CKIT-042) works as a lower terminal, and the BCP terminal on the PC works as an upper terminal. CY8CKIT-042 sends real-time motor state data through PSoC 4 dedicated UART pins (P4.0 and P4.1), and then data is pushed into the USB-to-UART bridge controller (CY7C65213) on the driver board (CY8CKIT-037). CY7C65213 repackages the data and sends it to the upper terminal through the USB cable, so you can see the curve or chart of the real-time motor state data.

Figure 5-69. Real-Time Debugging Structure



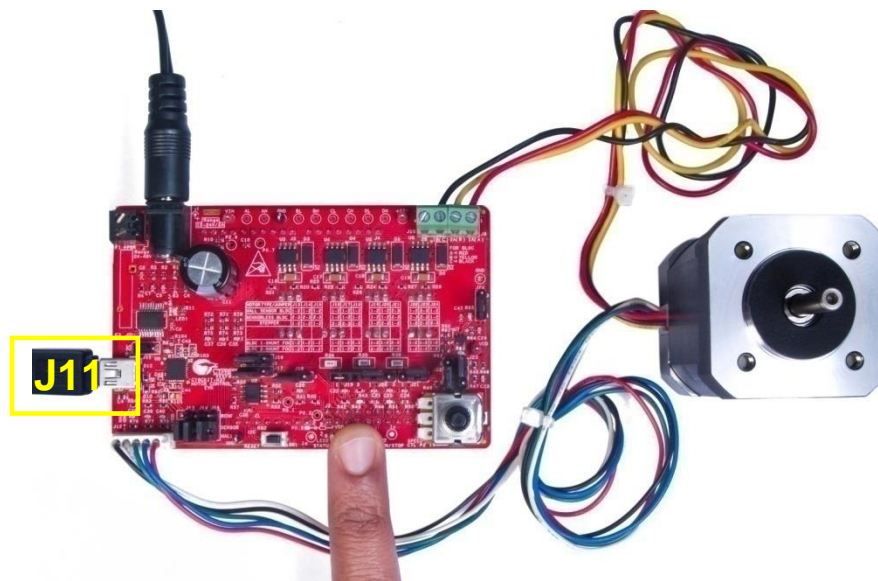
5.8.2 Installing the Driver for CY7C65213 USB-to-UART Bridge Controller

The first step in monitoring data on the BCP is to set up the hardware and install the driver program for the USB-to-UART bridge controller (CY7C65213) on CY8CKIT-037.

Connect the USB cable to CY8CKIT-037 at connector J11. Then, configure the hardware for the Sensored BLDC Motor Control code example following step1 to step 5 in [Running the Sensored BLDC Motor Control Code Example](#).

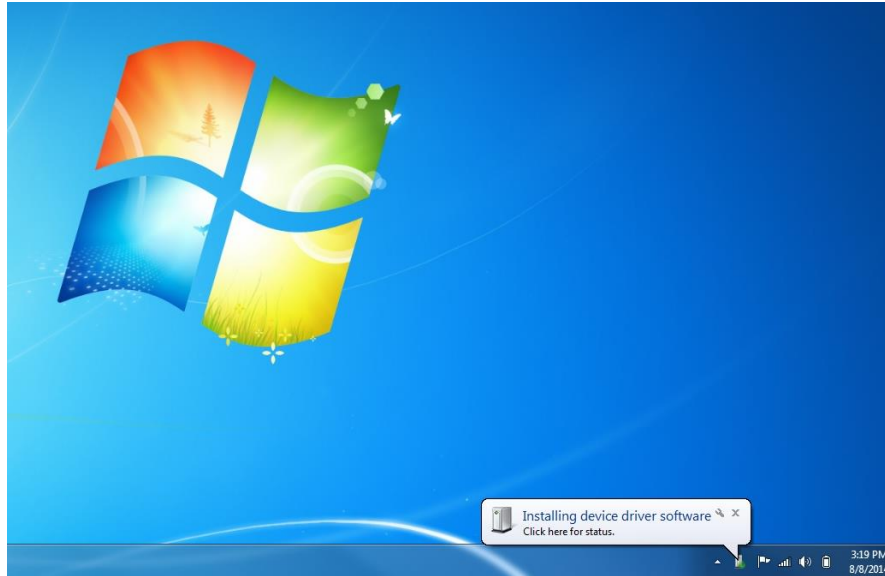
Figure 5-70 shows the hardware setup.

Figure 5-70. Hardware Setup for BCP



Connect the other end of J11 to your PC. The PC operating system (Windows 7 in this document) will automatically recognize the CY7C65213 USB-to-UART bridge controller and search the driver program for it, as shown in [Figure 5-71](#).

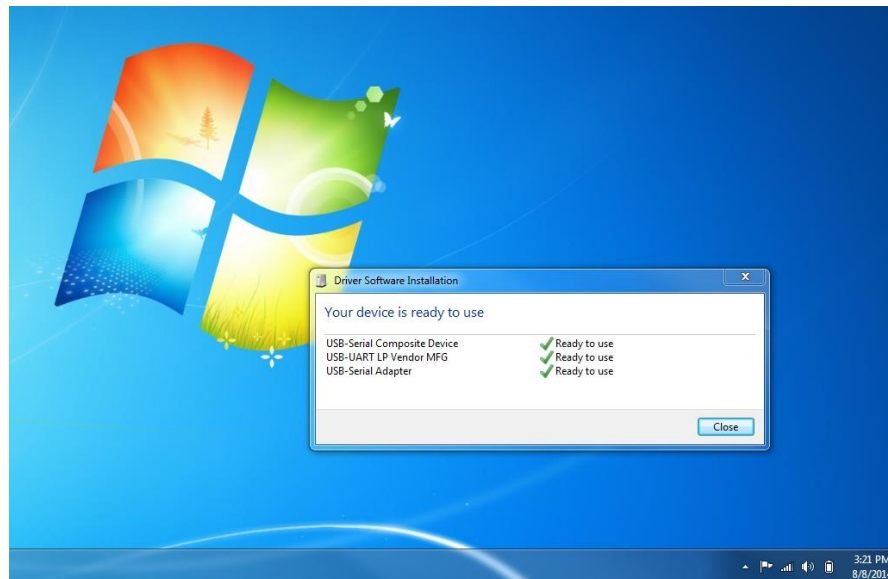
Figure 5-71. CY7C65213 USB-to-UART Bridge Controller Recognition



When the operating system gets the driver program, it will automatically install it for CY7C65213. [Figure 5-72](#) shows the information window that indicates that the driver program installation for the CY7C65213 USB-to-UART Bridge Controller is finished.

CAUTION Do not press switch SW2 on the CY8CKIT-037 board during the installation process because the controller board (CY8CKIT-042) is unable to send any data to the USB-to-UART Bridge Controller (CY7C65213) during the installation process. Otherwise, the BCP will not work after the installation. All the code examples included in the kit send data only after the motor is started by pressing the start/stop switch, SW2.

Figure 5-72. Driver Program Installation Completion



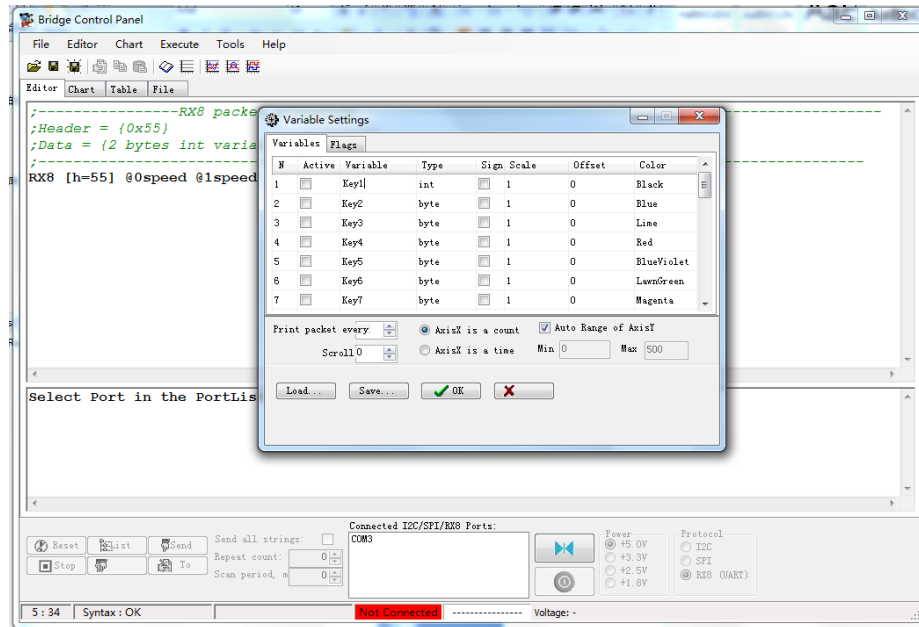
5.8.3 Upper Terminal Configuration Guide

Each code example contains the lower terminal firmware to send real-time data and attaches two upper terminal configuration files, *.ini* and *.iic*.

This section uses the Sensored BLDC Motor Control code example as an example. Other code examples are similar.

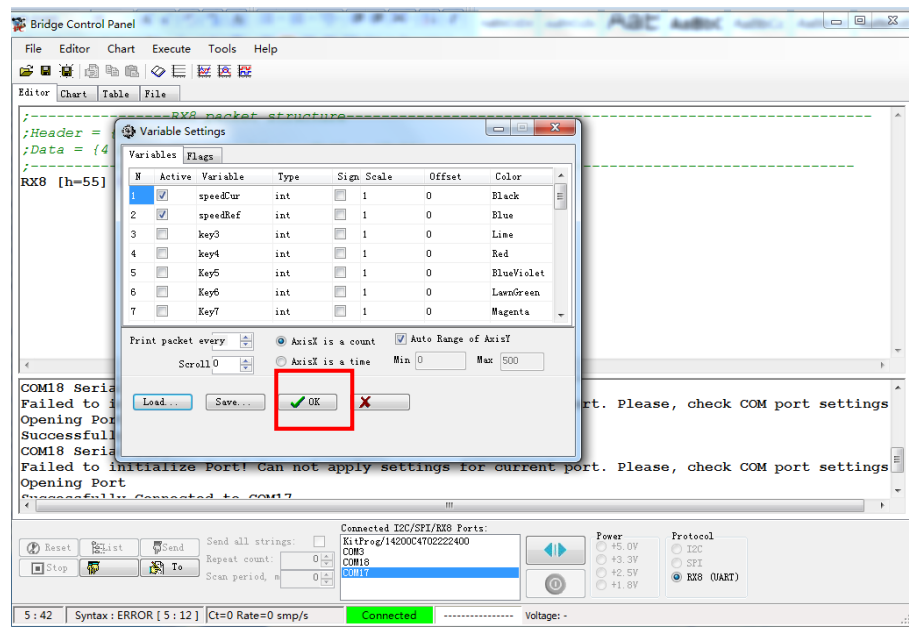
Open Bridge Control Panel (BCP) and select the COM port, and choose **Chart > Variable Settings** to open the **Variable Settings** window, as [Figure 5-73](#) shows.

Figure 5-73. Variable Settings Window



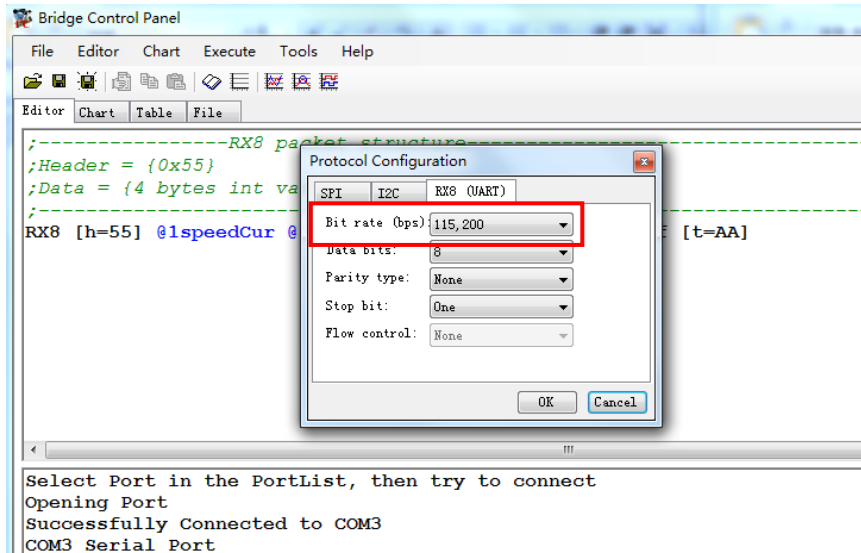
Then, click the **Load** button to import the *variables.ini* file into the Sensored BLDC Motor Control code example, as [Figure 5-74](#) shows. Click the **OK** button to finish.

Figure 5-74. Import variables .ini File



You also need to set the baud of BCP communication. Enter the BCP software interface, and choose **Tools > Protocol Configuration** to open the **Protocol Configuration** window, as [Figure 5-75](#) shows. The recommended baud setting is 115,200 bps.

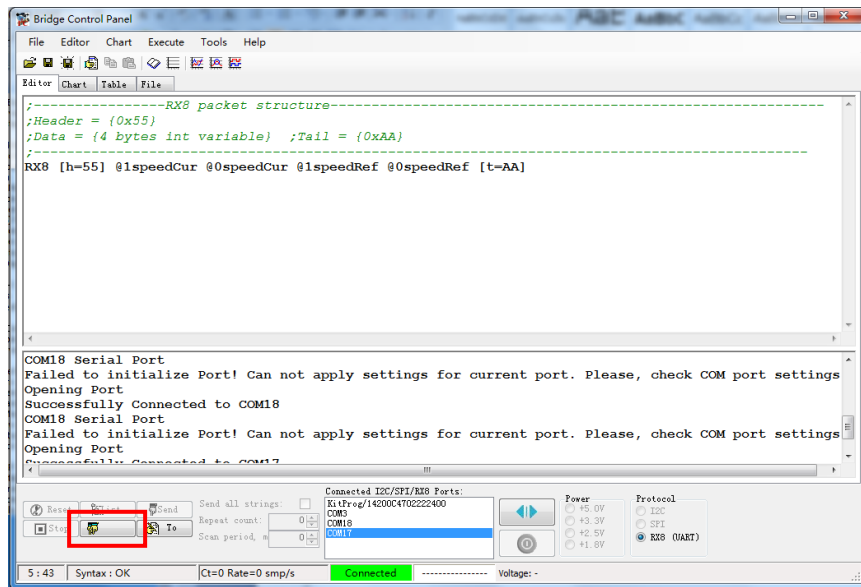
Figure 5-75. Baud (bps) Settings Window



Choose **File > Open File** to import the *Sensored BLDC Control.iic* configuration file into the Sensored BLDC Motor Control code example, as [Figure 5-76](#) shows. Click the highlighted button to start monitoring data defined by the command in the **Editor** and **Variable Settings** windows.

The **Editor** window shows the BCP monitor 2 variables as a default. “SpeedCur” shows the real-time motor speed, and “SpeedRef” shows the reference motor speed. You can change or add monitoring variables by changing the command statement in the **Editor** window and keeping the **Variable Settings** window updated synchronously.

Figure 5-76. Bridge Control Panel Editor



The command structure consists of four parts: start symbol “Rx8”, header, body, and tail as follows:

Rx8 [H=<byte0><byte1> ... <byteN>] <byte0><byte1> ... <byteK> [T=<byte0><byte1> ... <byteM>]

Where “Rx8” is a start symbol that indicates the beginning of the RX8 command.

[H=<byte0><byte1> ... <byteN>] is a header with a list of bytes. It is intended for synchronization of the BCP to an asynchronous input stream. The header part is optional. It can be present or absent in the script. But if it is absent, then the tail also should be absent. Otherwise, the ScriptEngine syntax analyzer will inform you about a syntax error and mark in red the incorrect part of the command.

<byte0><byte1> ... <byteK> is a body list of bytes. It is a useful packet load that is received by the host and shown in the Log and Chart subsystems, depending on the variables configuration in the command. The body is a mandatory part of the command.

[T=<byte0><byte1> ... <byteM>] is a tail with a list of bytes. It indicates the end of the RX8 command. The tail is an optional part of the command. The RX8 command can be present in the BCP script editor analyzer in three ways:

1) Without the header and tail, with only the body. In this scenario, the host does not synchronize with a stream of bytes. It simply prints bytes received from the USB-Serial device to the Log and on the Chart if variables were configured in the script, as shown in the following example.

RX8 X X @1Var1 @0Var1 X @3Var2 @2Var2 @1Var2 @0Var2 X X

2) With the header and body, but without the tail. In this scenario, the host synchronizes with a stream of bytes, finding header bytes in the stream, as shown in the following example.

RX8 [H = 0A 0E 0C] X X @1Var1 @0Var1 X @3Var2 @2Var2 @1Var2 @0Var2 X X

3) With the header, body, and tail. In this scenario, the host synchronizes with a stream of bytes, finding header, body, and tail bytes in the stream, as shown in the following example.

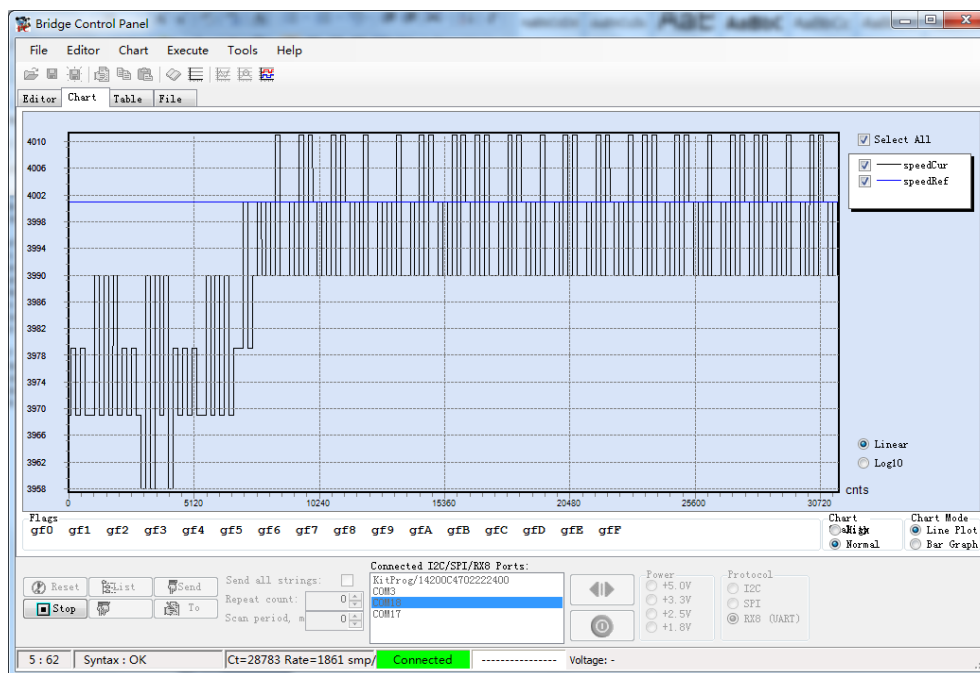
RX8 [H = 0A 0E 0C] X X @1Var1 @0Var1 X @3Var2 @2Var2 @1Var2 @0Var2 X X [T = 00 FF FF]

All five code examples use this command format: H = 0x55 and T = 0XAA. Variables are 16 bits and always in “MSB-LSB” format.

You can define your own commands by following these rules:

Keep the default configuration, Press SW2 on CY8CKIT-037 to start the motor. You can see the real-time data in the **Chart** window, as [Figure 5-77](#) shows. You can use a tachometer to measure the motor speed and check it on the BCP display.

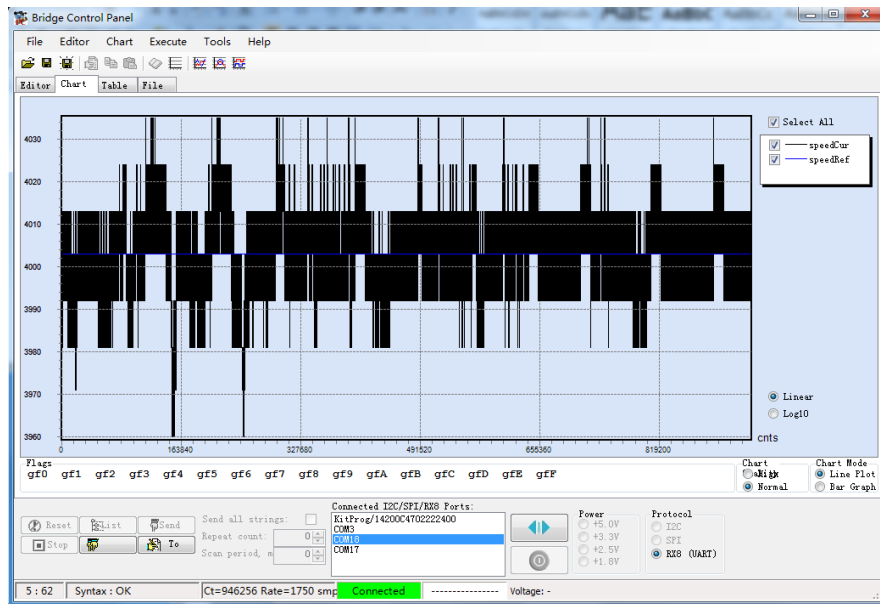
Figure 5-77. BCP Data Display for Sensored BLDC Motor Control Code Example



Note: Before you complete all the configuration steps in this section (from Figure 5-73 to Figure 5-76), ensure that switch SW2 on CY8CKIT-037 is not pressed. If you press it, you will not see data display on the BCP **Chart** window. However, if you do press SW2, press SW1 on CY8CKIT-037 to reset the motor, and then complete all the configuration steps shown in Figure 5-73 to Figure 5-76. You can then press SW2 to start the motor and observe the data on the BCP **Chart** window.

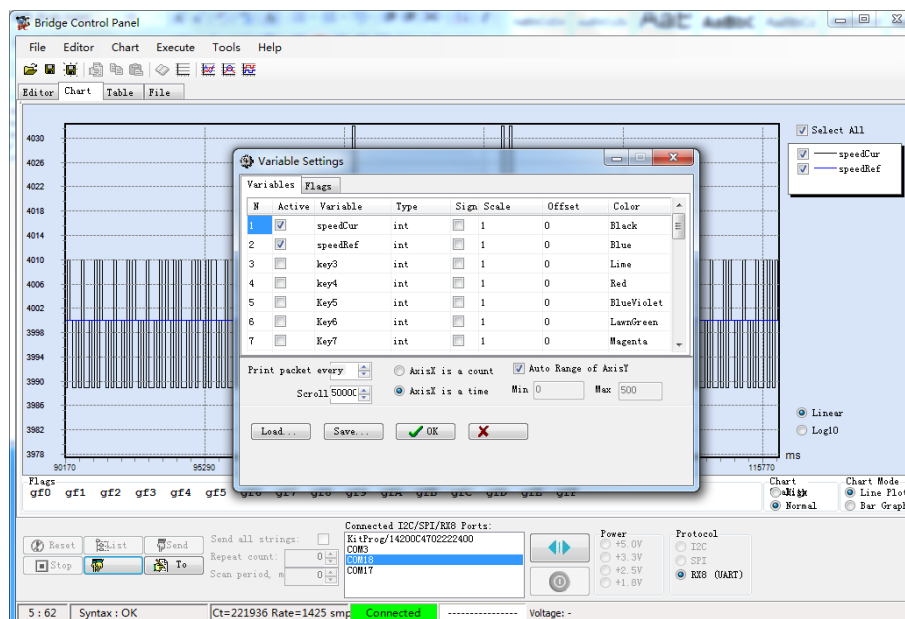
If you want to run the motor and monitor data for long time (more than 8 minutes), the BCP can display only the original 81920 counts (about 7 minutes at 115,200 bps), as Figure 5-78 shows. The later data no longer appears because the BCP puts all the received data for display in limited stack memory. So if the stack memory is full, the BCP cannot receive and display new data. If you want to monitor the real-time data for a long time, reconfigure the BCP to make it display only the latest data, not all the data from the motor or BCP start.

Figure 5-78. Maximum Data Counts on BCP Monitoring



Choose **Chart > Variable Settings** to open the **Variable Settings** window. Then select the “AxisX is a time” option and write a number (minimum: 1, maximum:500000, unit:ms) into the **Scroll** text box to define the latest time length to make the BCP display data, as shown in Figure 5-79. Then the stack memory will never be full, and the BCP can always show the latest data of the defined time length.

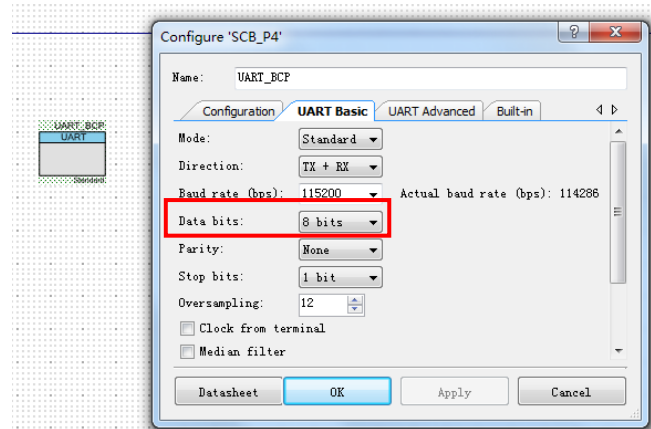
Figure 5-79. BCP Reconfiguration



5.8.4 Lower Terminal Configuration Guide

While the motor is running, MCU CY8C4245 on the controller board gets all motor state parameters and is responsible for sending them to the BCP upper terminal. [Figure 5-80](#) shows the Serial Communication Block (SCB) Component and configuration window. It sends motor state data through two dedicated UART pins (P4.0 and P4.1). It is configured as "UART_BCP".

Figure 5-80. SCB Component Configuration



Data is sent by function `BCPPoll()`, located in the main control loop of the firmware. The function code follows:

```
void BCPPoll()
{
    uint8 index = 0;

    if(UART_BCP_SpiUartGetTxBufferSize())
        return;

    /* package header */
    bcpTxBuffer[index++] = 0x55;

    /* construct BCP data package with speed value, MSB first */

    /* current measured speed */
    bcpTxBuffer[index++] = (uint8)((UI_Data.speedRpm & 0xFF00) >> 8);
    bcpTxBuffer[index++] = (uint8)(UI_Data.speedRpm & 0x00FF);
    /* speed reference */
    bcpTxBuffer[index++] = (uint8)((UI_Data.speedRpmRef & 0xFF00) >> 8);
    bcpTxBuffer[index++] = (uint8)(UI_Data.speedRpmRef & 0x00FF);

    /* package tail */
    bcpTxBuffer[index++] = 0xAA;

    UART_BCP_SpiUartPutArray(bcpTxBuffer, index);
}
```

This function sends two motor state variables: `UI_Data.speedRpm` and `UI_Data.speedRpmRef`. You can modify the variable name and add variables by copying the highlighted code and then changing the variable name from `UI_Data.speedRpm` to whatever you want.

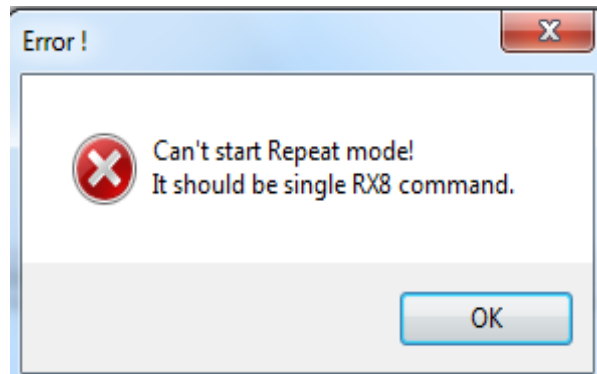
5.8.5 Reading the Motor Speed Using BCP Commands

The BCP commands are not functional if the tool is invoked after the motor starts rotating.

To view the proper motor rotation status or RPM readings, follow these steps:

1. Connect and configure CY8CKIT-037 on the CY8CKIT-042 Pioneer Kit per the [Sensored BLDC Motor Control Code Example](#).
2. Program the HEX file.
3. Connect a USB cable at the J11 port of CY8CKIT-037 to enable the motor speed to be viewed in the BCP. (Wait until driver enumeration is complete.)
4. Open the BCP and import the UART commands related to the Sensored BLDC project. Refer the [Upper Terminal Configuration Guide](#).
5. Select the appropriate COM port related to USB-to-UART on CY8CKIT-037 (for example: COM17 in [Figure 5-76](#)).
6. Move the cursor to the end of the BCP command "RX8 [h=55] @1speedCur @0speedCur @1speedRef @0speedRef [t=AA]" in the **Command** window, and press **Enter** to execute the BCP command once. Then click the **Repeat** button to repeat the command execution (If this operation is missing, an error message window is displayed as shown in [Figure 5-81](#)).

Figure 5-81. Error Message that appears when Step 6 is missed while configuring BCP



7. Press the **SW2** button on CY8CKIT-037 to start the BLDC motor rotation.
8. Observe the motor speed on the BCP Editor window (see [Figure 5-82](#)) and in Chart Window (see [Figure 5-83](#)).

Figure 5-82. Motor Speed in Editor Window

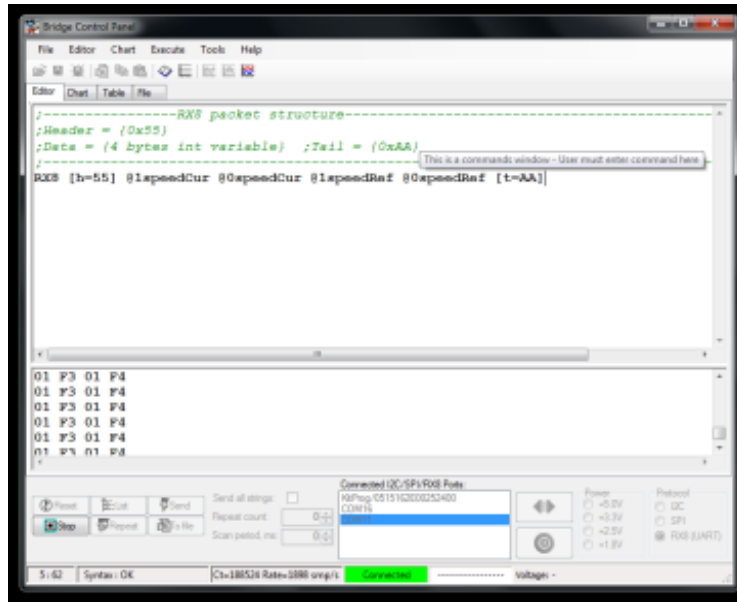


Figure 5-83. Motor Speed in Chart Window

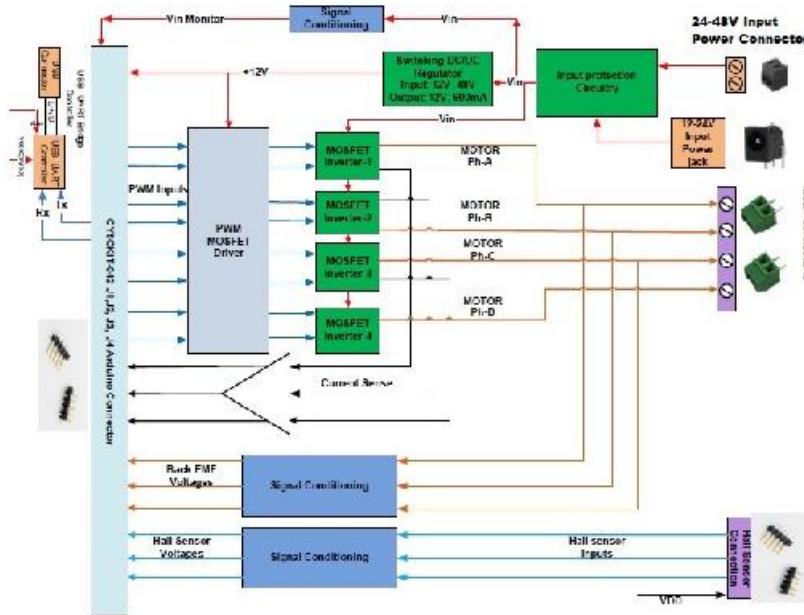


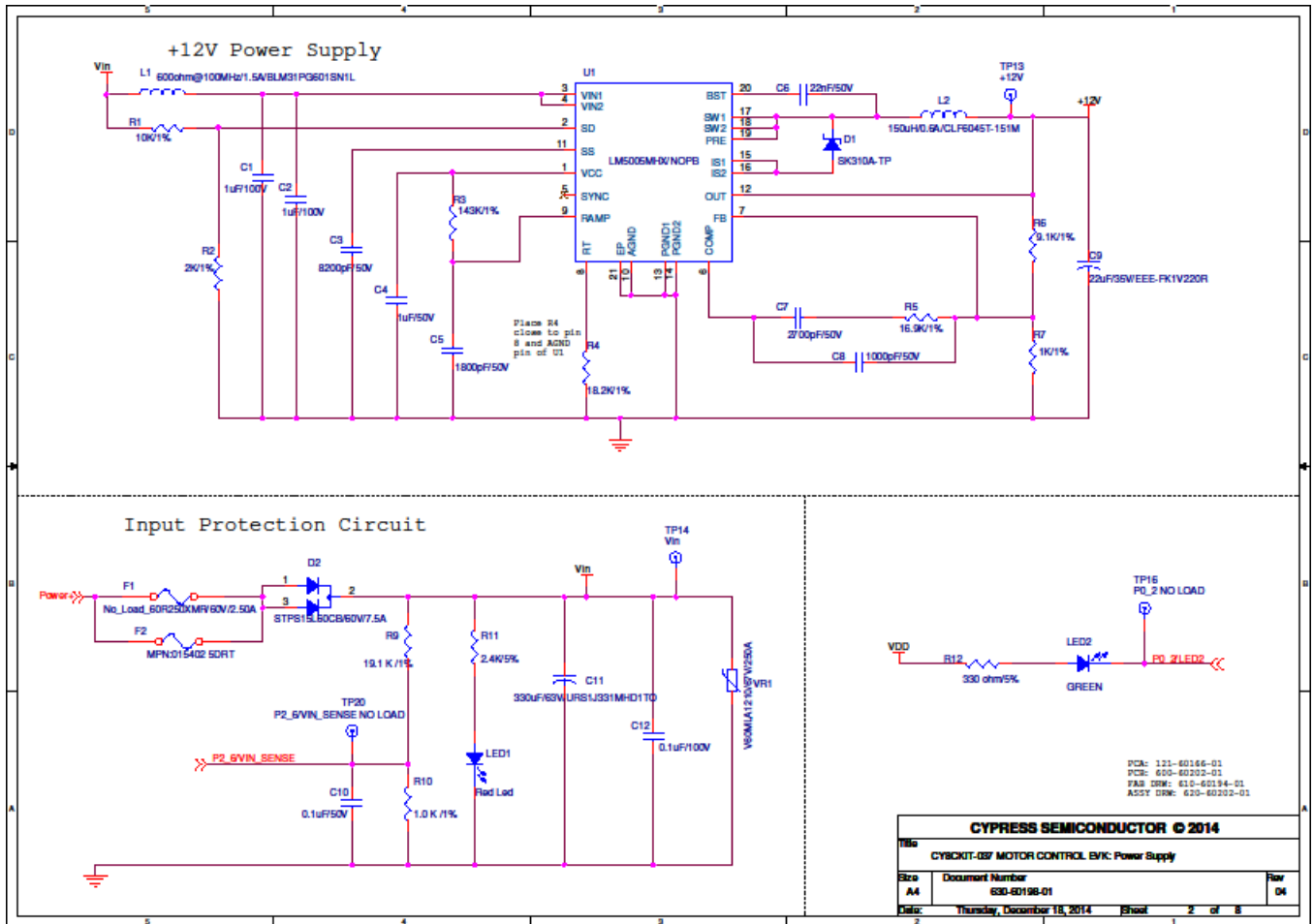
A. Appendix

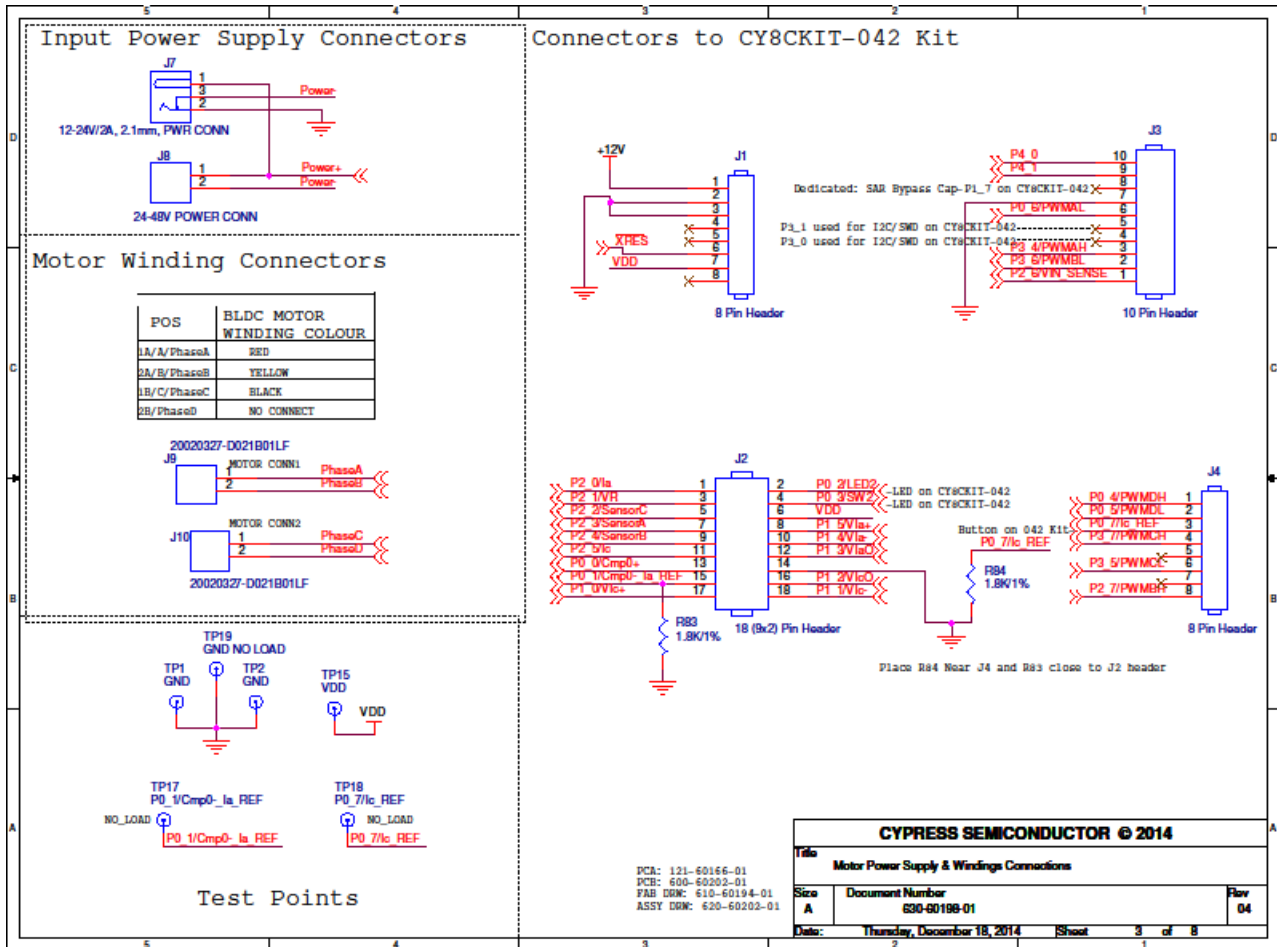


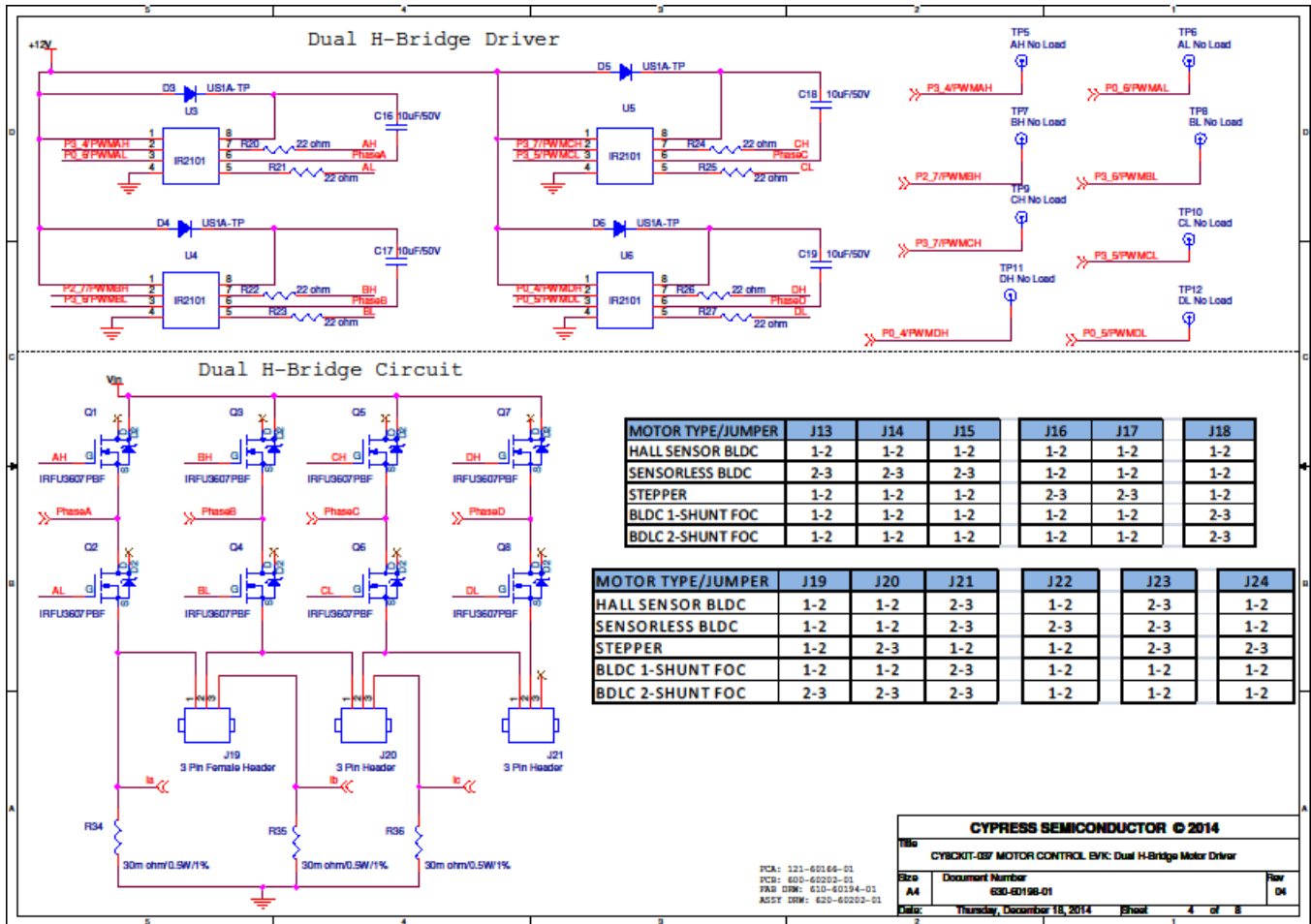
A.1. Board Schematics, Board Layout, and BOM

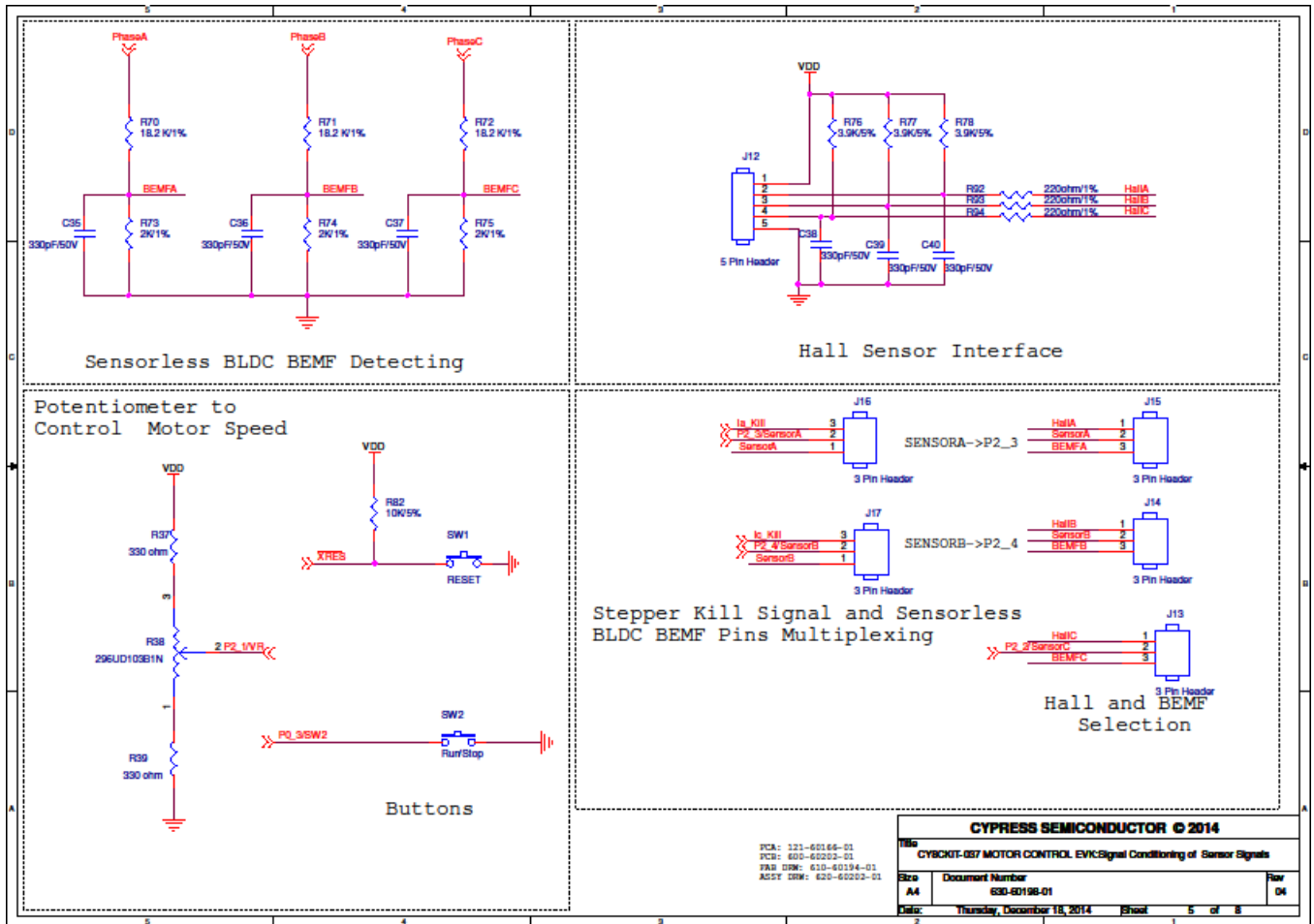
Figure A-1. Board Schematics

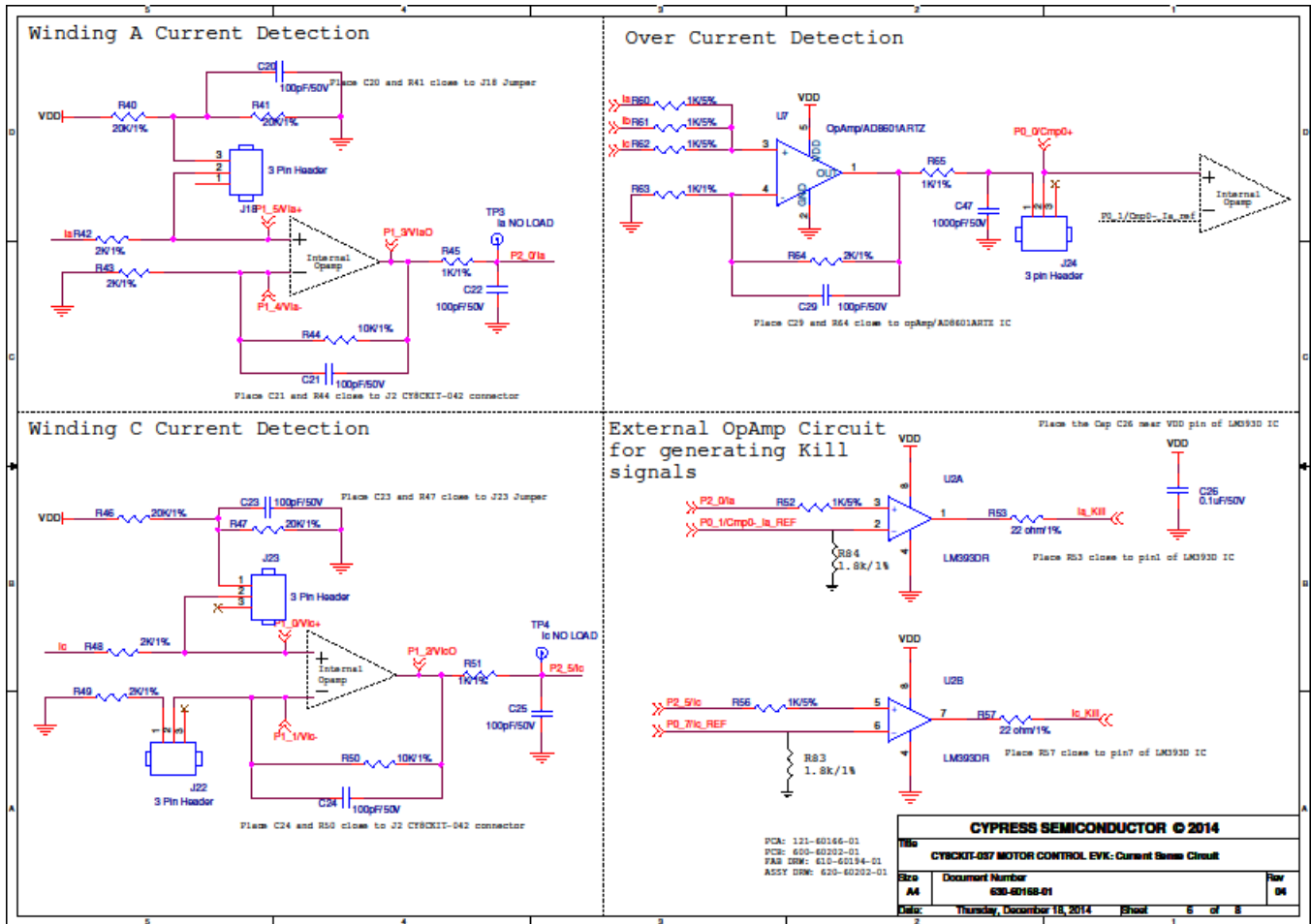












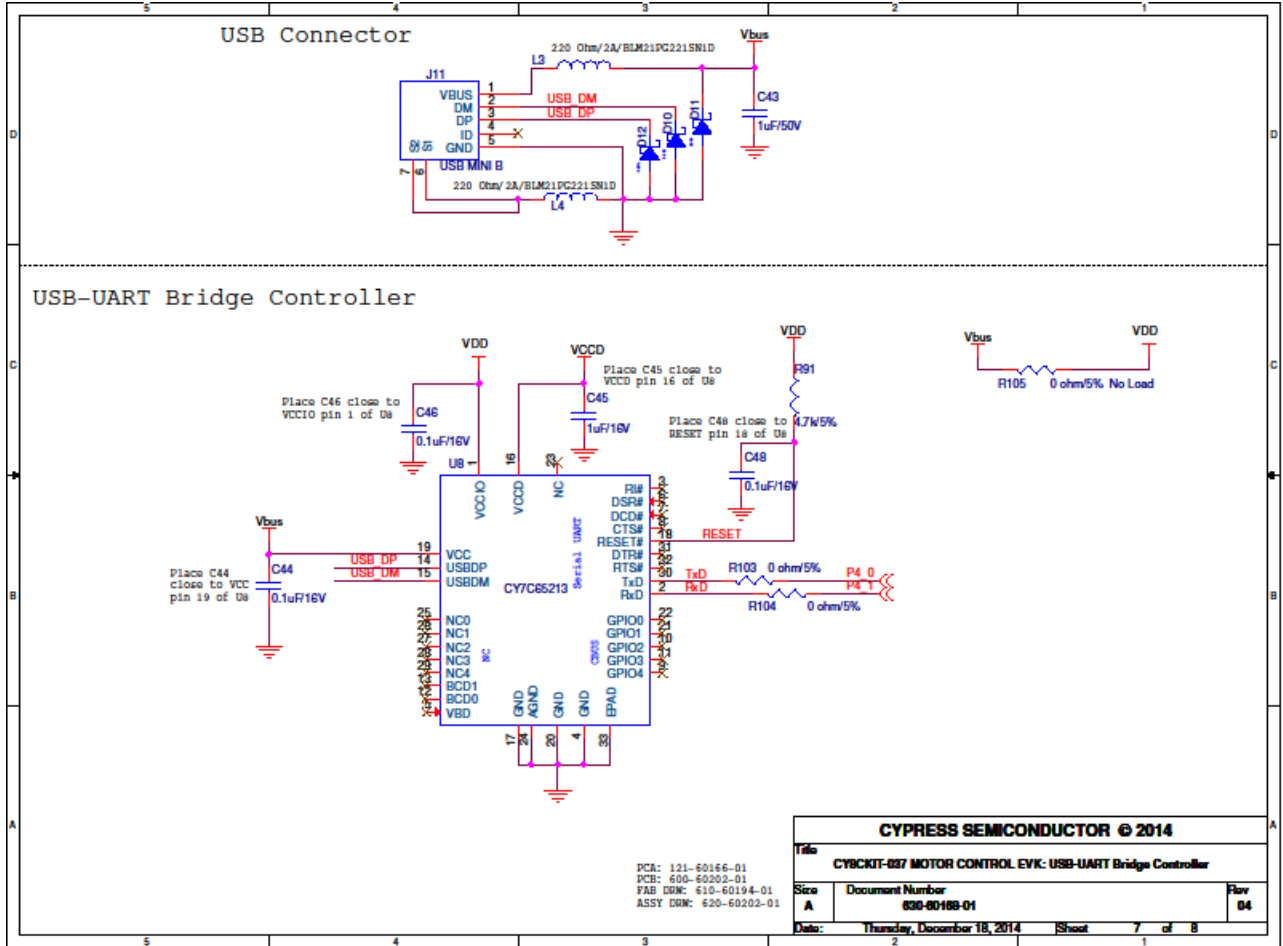


Figure A-2. Board Layout (Top Layer)

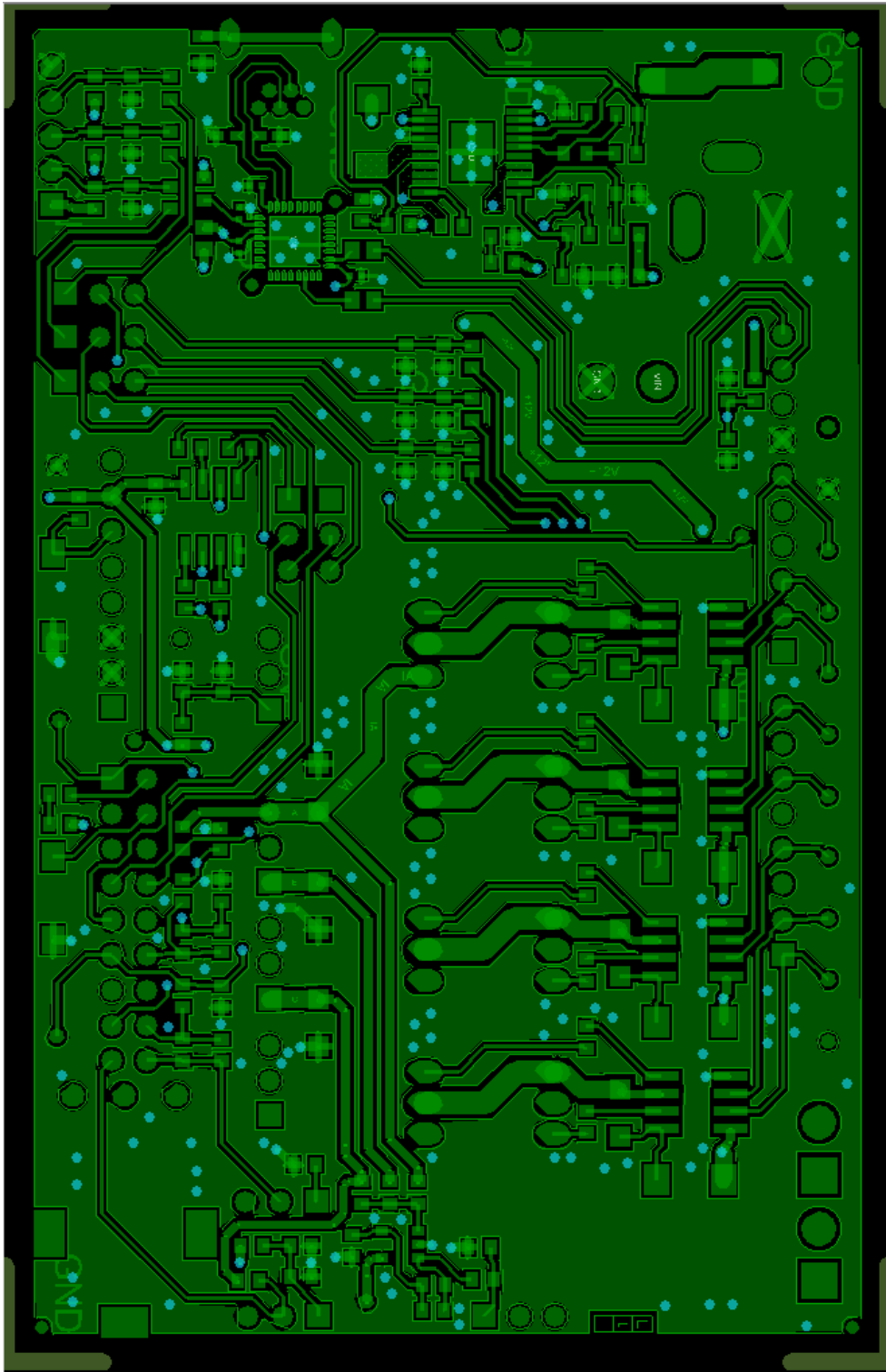
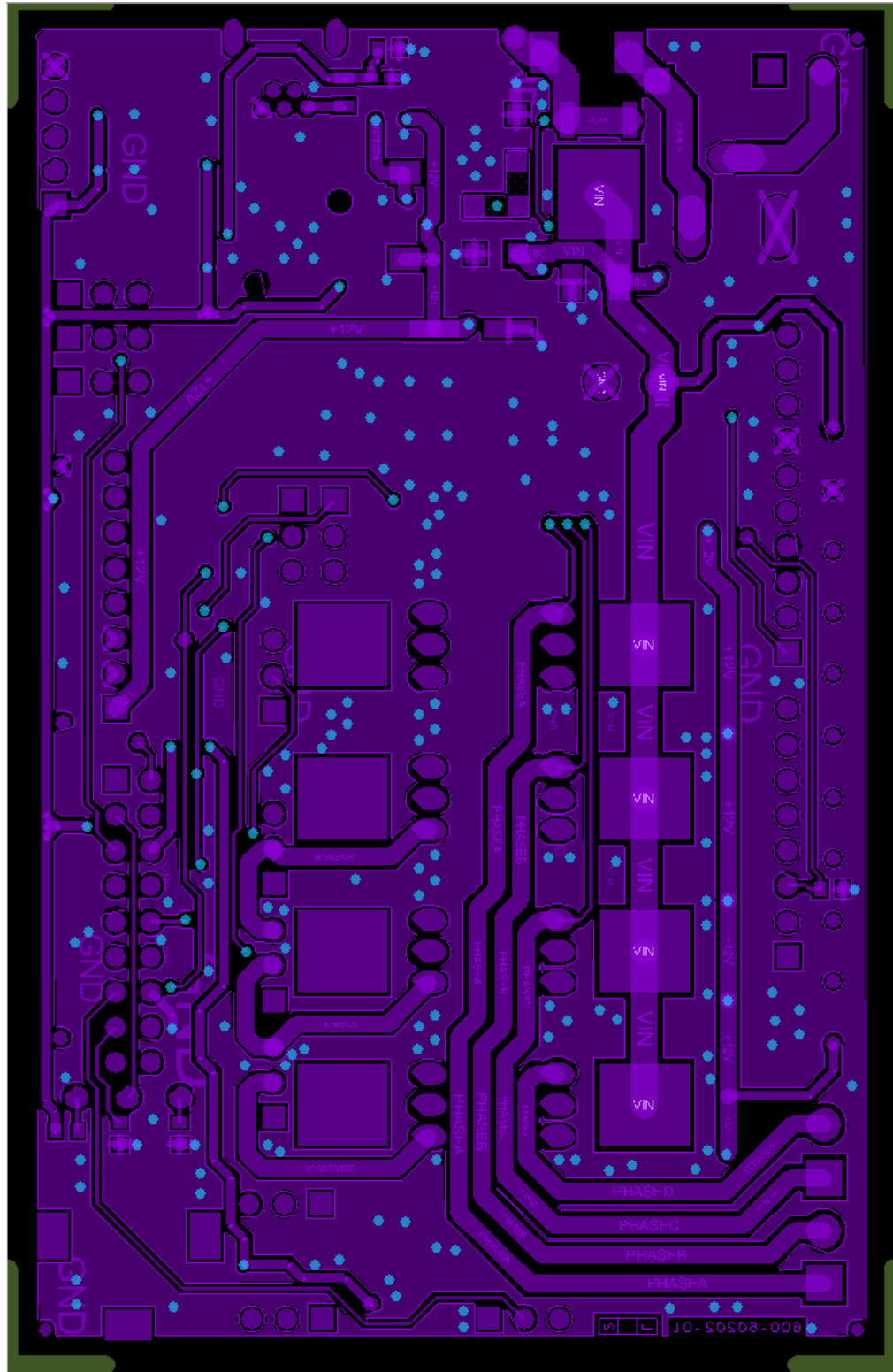


Figure A-3. Board Layout (Bottom Layer)



A.2. Bill of Materials

Table A-1. Bill of Materials

Item	Qty	Reference	Description	Manufacturer	Mfr Part Number
1	2	C1, C2	CAP CER 1UF 100V 10% X7R 1206	TDK Corporation	C3216X7R2A105K160AA
2	1	C3	CAP CER 8200PF 50V 10% NP0 0603	TDK Corporation	C1608C0G1H822K080AA
3	2	C4, C43	CAP CER 1UF 50V 10% X7R 0603	Taiyo Yuden	UMK107AB7105KA-T
4	1	C5	CAP CER 1800PF 50V 10% NP0 0603	TDK Corporation	C1608C0G1H182K080AA
5	1	C6	CAP CER 0.022UF 50V 10% X8R 0603	TDK Corporation	C1608X8R1H223K080AA
6	1	C7	CAP CER 2700PF 50V 10% NP0 0603	TDK Corporation	C1608C0G1H272K080AA
7	2	C8, C47	CAP CER 1000PF 50V 10% X7R 0603	TDK Corporation	CGA3E2X7R1H102K080AA
8	1	C9	CAP ALUM 22UF 35V 20% SMD	Panasonic Electronic Components	EEE-FK1V220R
9	2	C10, C26	CAP CER 0.1UF 50V 10% X7R 0603	TDK Corporation	C1608X7R1H104K080AA
10	1	C11	CAP ALUM 330UF 63V 20% RADIAL	Nichicon	UVZ1J331MPD
11	1	C12	CAP CER 0.1UF 100V 10% X7S 0603	TDK Corporation	CGA3E3X7S2A104K080AB
12	4	C16, C17, C18, C19	CAP CER 10UF 50V 10% X5R 1206	TDK Corporation	C3216X5R1H106K160AB
13	7	C20, C21, C22, C23, C24, C25, C29	CAP CER 100PF 50V 10% CH 0603	TDK Corporation	C1608CH1H101K080AA
14	6	C35, C36, C37, C38, C39, C40	CAP CER 330PF 50V 10% NP0 0603	TDK Corporation	C1608C0G1H331K080AA
15	3	C44, C46, C48	CAP CER 0.1UF 16V 10% X7R 0402	TDK Corporation	C1005X7R1C104K050BC
16	1	C45	CAP CER 1UF 16V 10% X5R 0402	TDK Corporation	C1005X5R1C105K050BC
17	1	D1	DIODE SCHOTTKY 100V 3A SMA	Micro Commercial Co	SK310A-TP
18	1	D2	DIODE SCHOTTKY 60V 7.5A DPAK	STMicroelectronics	STPS15L60CB-TR
19	4	D3, D4, D5, D6	DIODE FAST REC 50V 1A SMA	Micro Commercial Co	US1A-TP
20	3	D10, D11, D12	DIO, SUPPRESSOR ESD 5VDC 0603 SMD	Bourns Inc	CG0603MLC-05LE
21	1	F1	PTC RESETTABLE 60V 2.50A KINKAMO	Littelfuse Inc	60R250XMR
22	2	J1, J4	2.54mm PITCH SINGLE ROW 8POS HEADER, 13mm MATING LENGTH	PROTECTOR	P9101-08-D32-1
23	1	J2	2.54mm PITCH DUAL ROW 18POS HEADER, 13mm MATING LENGTH	PROTECTOR	P9103-18-D32-1

Item	Qty	Reference	Description	Manufacturer	Mfr Part Number
24	1	J3	2.54mm PITCH SINGLE ROW 10POS HEADER, 13mm MATING LENGTH	PROTECTRON	P9101-10-D32-1
25	1	J7	CONN, JACK, POWER, 2.1mm, MALE, 3PIN, PCB RA, TH	CUI Inc	PJ-102A
26	1	J8	BLACK COLOUR, TERM BLOCK 2POS SIDE ENT 3.81MM	TE Connectivity	1776113-2
27	2	J9, J10	GREEN COLOUR, TERM BLOCK 2POS 3.81MM PCB HORIZ	FCI	20020327-D021B01LF
28	1	J11	CONN, MINI USB, RCPT R/A, DIP, B TYPE	TE Connectivity	1734510-1
29	1	J12	ConWTB Wire to Board Connector 2.50mm pitch, "J" Type Vertical Male Shrouded Header, 5 pin	Wurth Elektronik	688 005 116 22
30	12	J13, J14, J15, J16, J17, J18, J19, J20, J21, J22, J23, J24	CONN, HEADER, VERT, SGL, 3POS, GOLD	PROTECTRON	P9101-03-12-1
31	1	LED1	LED RED CLEAR 0603 SMD	Lite-On Inc	LTST-C190CKT
32	1	LED2	LED GREEN CLEAR THIN 0603 SMD	Lite-On Inc	LTST-C190GKT
33	1	L1	FERRITE CHIP 600 OHM 1500MA 1206	Murata Electronics North America	BLM31PG601SN1L
34	1	L2	INDUCTOR POWER 150UH 20% SMD	TDK Corporation	CLF6045T-151M-H
35	2	L3, L4	IND, FERRITE, 220 OHM, 2A, 25%, 0805	Murata Electronics North America	BLM21PG221SN1D
36	8	Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8	MOSFET N-CH 75V 56A I-PAK	International Rectifier	IRFU3607PBF
37	3	R1, R44, R50	RES 10.0K OHM 1/16W 1% 0603	TE Connectivity	5-1879337-9
38	5	R2, R64, R73, R74, R75	RES 2.00K OHM 1/10W 1% 0603 SMD	Vishay Dale	CRCW06032K00FKEA
39	1	R3	RES 143K OHM 1/10W 1% 0603 SMD	Yageo	RC0603FR-07143KL
40	1	R4	RES 18.2K OHM 1/10W 1% 0603 SMD	Yageo	RC0603FR-0718K2L
41	3	R70, R71, R72	RES 8.20K OHM 1/10W 1% 0603 SMD	Vishay Dale	CRCW06038K20FKEA
42	1	R5	RES 16.9K OHM 1/10W 1% 0603 SMD	Vishay Dale	CRCW060316K9FKEA
43	1	R6	RES 9.10K OHM 1/10W 1% 0603 SMD	Vishay Dale	CRCW06039K10FKEA
44	5	R7, R45, R51, R63, R65	RES 1.00K OHM 1/8W 1% SMD 0603	Vishay Beyschlag	MCT06030C1001FP500
45	1	R10	RES 590 OHM 1/10W 1% 0603 SMD	Vishay Dale	CRCW0603590RFKEA
46	1	R9	RES, 5.49K OHM, 1%, 1/10W, 0603, SMD	Panasonic-ECG	ERJ-3EKF5491V

Item	Qty	Reference	Description	Manufacturer	Mfr Part Number
47	1	R11	RES 2.4K OHM 1/10W 5% 0603 SMD	Yageo	RC0603JR-072K4L
48	3	R12, R37, R39	RES, 330 OHM, 5%, 200PPM, 1/10W, 0603	Panasonic-ECG	ERJ-3GEYJ331V
49	10	R20, R21, R22, R23, R24, R25, R26, R27, R53, R57	RES, 22 OHM, 1%, 200PPM, 1/10W, 0603, SMD	Panasonic-ECG	ERJ-3EKF22R0V
50	3	R34, R35, R36	RES 0.03 OHM 1/2W 1% 1206 SMD	Rohm Semiconductor	UCR18EVHFSR030
51	1	R38	POT 10K OHM 1/8W CARB VERTICAL	CTS Corporation	296UE103B1C
52	4	R40, R41, R46, R47	RES, 20K OHM, 1%, 100PPM, 1/10W, 0603	Panasonic Electronic Components	ERJ-3EKF2002V
53	4	R42, R43, R48, R49	RES 2K OHM 1/10W 1% 0603 SMD	Panasonic Electronic Components	ERJ-3EKF2001V
54	5	R52, R56, R60, R61, R62	RES 1.0K OHM 1/10W 5% 0603 SMD	Vishay Dale	CRCW06031K00JNTA
55	3	R76, R77, R78	RES 3.9K OHM 1/10W 5% 0603 SMD	Yageo	RC0603JR-073K9L
56	1	R82	RES 10.0K OHM 1/10W 5% 0603	Yageo	RC0603JR-0710KL
57	2	R83, R84	RES 1.80K OHM 1/10W 1% 0603 SMD	Vishay Dale	CRCW06031K80FKEA
58	1	R91	RES 4.7K OHM 1/10W 5% 0603 SMD	Panasonic Electronic Components	ERJ-3GEYJ472V
59	3	R92, R93, R94	RES 220 OHM 1/10W 1% 0603 SMD	Panasonic Electronic Components	ERJ-3EKF2200V
60	2	R103, R104	RES, 0.0 OHM, 1/10W, 5%, 0603, SMD	Panasonic Electronic Components	ERJ-3GEY0R00V
61	2	SW1, SW2	SWITCH TACTILE SPST-NO 0.05A 12V	Panasonic Electronic Components	EVQ-PE105K
62	2	TP1(GND), TP2(GND) ,	TP, TEST POINT, 0.063"D, BLACK, TH	Keystone Electronics	5006
63	3	TP13(+12V), TP14(Vin), TP15(VDD)	TP, TEST POINT, RED, 0.063"D, TH	Keystone Electronics	5005
64	1	U1	IC REG BUCK ADJ 2.5A 20TSSOP	TI	LM5005MHX/NOPB
65	1	U2	IC COMPARATOR DUAL 0.8MA 8-SOIC	Rohm Semiconductor	LM393DR
66	4	U3, U4, U5, U6	IC DRIVER HIGH/LOW SIDE 8SOIC	International Rectifier	IR2101STRPBF
67	1	U7	IC OPAMP GP 8.4MHZ RRO SOT23-5	Analog Devices	AD8601ARTZ-R2

Item	Qty	Reference	Description	Manufacturer	Mfr Part Number
68	1	U8	IC, CYPRESS, USB-UART LP, 512-BYTE FLASH, QFN-32	Cypress Semiconductor	CY7C65213-32LTXI
69	1	VR1	VARISTOR 67V 250A 1210	Littelfuse Inc	V60MLA1210H
70	1	PCB	CY8CKIT-037 MOTOR CONTROL SHIELD PCB (Size: 3.8 inch x 2.44 inch No of Layers: 2 Surface Finish: ENIG Solder Mask color: Red Board thickness: 1.6mm)		

A.3. No Load Components

Table A-2. No Load Components

Item	Qty	Reference	Description	Manufacturer	Manufacturer Part Number
1	1	R105	RES, 0.0 OHM, 1/10W, 5%, 0603, SMD	Panasonic Electronic Components	ERJ-3GEY0R00V
2	15	TP3(Ia), TP4(Ic), TP5(AH), TP6(AL), TP7(BH), TP8(BL), TP9(CH), TP10(CL), TP11(DH), TP12(DL), TP16(P0_2), TP17(P0_1), TP18(P0_7), TP19(GND), TP20(P2_6/VIN_SENSE)	TP, TEST POINT, RED, 0.063"D, TH	Keystone Electronics	5005
3	1	F2	FUSE SLOW 125V 2.5A SMD	Littelfuse Inc	045202.5MRL

A.4. Special Installation Instructions

Table A-3. Special Installation Instructions

Item	Qty	Reference	Description	Manufacturer	Manufacturer Part Number
1	12	J13, J14, J15, J16, J17, J18, J19, J20, J21, J22, J23, J24	MINI JUMPER GF 13.5 CLOSE TYPE BLACK WITH HANDLE	Kobiconn	151-8030-E
2	1	J12	ConWTB Wire To board Connector 2.50mm pitch, "J" type Female Terminal Housing, 5 Pins	Würth Elektronik	688 005 113 322
3	5	J12 Hall sensor wire Crimping	2.50mm "J" Type Female Crimp Contact WR-WTB	Würth Elektronik	688 001 137 22

A.5. Label

Table A-4. Label

Item	Qty	Reference	Description	Manufacturer
1	1	N/A	LBL, Kit Product Identification Label, Vendor Code, Datecode, Serial Number CY8CKIT-037 Motor Control EVK Rev**(YYWWVVXXXXX)	Cypress Semiconductor
2	1	N/A	LBL, Assembly Adhesive Label, Manufacturing ID	Cypress Semiconductor
3	1	N/A	LBL, QR Code, CY8CKIT-037 Printed Circuit Assembly, 13mm X 13mm	Cypress Semiconductor

Revision History



Document History

Document Title: CY8CKIT-037 PSoC® 4 Motor Control Evaluation Kit Guide				
Document Number: 001-92562				
Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	4784695	ROWA	06/02/2015	New document.
*A	5761068	AESATMP9	06/02/2017	Updated logo and copyright.
*B	5926483	HUAL	11//22/2017	Updated to new template.
*C	6145250	SAGA	04/19/2018	<p>Updated hyperlinks across the document. Removed unwanted hyperlinks across the document. Replaced PSoC 4A with PSoC 4 in all instances across the document. Replaced "Example Projects" with "Code Examples" in all instances across the document.</p> <p>Updated Introduction: Updated Kit Contents: Updated description. Updated Figure 1-1. Updated Additional Learning Resources: Updated description. Updated Technical Support: Replaced "Ext. 2" with "Ext. 3".</p> <p>Updated Software Installation: Updated Before You Begin: Updated description. Updated Install Kit Software: Updated description. Updated Figure 2-2. Updated Figure 2-3. Updated Uninstall Software: Updated description.</p> <p>Updated Kit Operation: Updated CY8CKIT-037 Evaluation Kit Overview: Updated Figure 3-1. Updated Kit Operation and Configuration Guide:</p>

				<p>Updated Hall Sensors Interface Connector: Replaced “Hall Sensors Connector” with “Hall Sensors Interface Connector” in heading.</p> <p>Updated USB Connector: Updated description.</p> <p>Updated Hardware: Updated DC/DC Switching Regulator: Updated description.</p> <p>Updated Phase Current Detecting and Processing Circuit: Updated description.</p> <p>Updated Connectors: Updated description.</p> <p>Updated Code Examples: Added Using the Kit Code Examples. Updated Sensored BLDC Motor Control Code Example: Updated Sensored BLDC Background: Updated description.</p> <p>Updated Control Schematic Overview: Updated description.</p> <p>Updated Flow Chart: Replaced “Firmware Introduction” with “Flowchart” in heading.</p> <p>Updated Running the Sensored BLDC Motor Control Code Example: Updated Step 4 – Connect the Power Supply and Motor: Updated description.</p> <p>Updated Step 5 – Build the Project and Program PSoC 4: Updated description.</p> <p>Updated Step 6 – Press the SW2 Button to Start Motor Rotation: Updated description.</p> <p>Updated Adapting the Code Example to another Motor: Updated description.</p> <p>Updated Sensorless BLDC Motor Control : Updated Sensorless BLDC Motor Control Code Example Overview: Updated description.</p> <p>Updated Control Schematic Overview: Updated description.</p> <p>Updated Running the Sensorless BLDC Motor Control Code Example: Updated Step 4 – Connect the Power Supply and Motor: Updated description.</p> <p>Updated Step 5 – Build the Project and Program PSoC 4: Updated description.</p> <p>Updated Step 6 – Press the SW2 Button to Start Motor Rotation: Updated description.</p>
--	--	--	--	--

			<p>Updated Adapting the Code Example to another Motor: Updated Tuning Startup Procedure: Updated description. Updated Tuning Parameters Overview: Updated Table 5-1. Updated Sensorless FOC Motor Control Code Example: Updated Control Schematic Overview: Updated description. Updated Running the Sensorless FOC Motor Control : Updated Step 1 – Configure CY8CKIT-042: Updated description. Updated Figure 5-44. Updated Step 5 – Build the Project and Program PSoC 4: Updated description. Updated Step 4 – Connect the Power Supply and Motor: Updated description. Updated Step 6 – Rotate the Potentiometer R38 to Start Motor Rotation: Updated description. Updated Adapting the Code Example to another Motor: Updated Table 5-3. Updated Table 5-4. Updated Single Shunt FOC Motor Control Code Example: Updated Single Shunt FOC Background: Updated description. Updated Single Shunt FOC Motor Control Code Example Overview: Updated description. Updated Control Schematic Overview: Updated description. Updated Running the Single Shunt FOC Motor Control Code Example: Updated Step 1 – Configure CY8CKIT-042: Updated description. Added Figure 5-55. Updated Step 4 – Connect the Power Supply and Motor: Updated description. Updated Step 5 – Build the Project and Program PSoC 4: Updated description. Updated Step 6 – Press the SW2 Button to Start Motor Rotation: Updated description. Updated Stepper Motor Control Code Example: Updated Control Schematic Overview: Updated description. Updated Figure 5-63. Updated Running the Stepper Motor Control Code Example: Updated Step 4 – Connect the Power Supply and Motor: Updated description.</p>
--	--	--	--

				<p>Updated Step 5 – Build the Project and Program PSoC 4: Updated description.</p> <p>Updated Step 6 – Press the SW2 Button to Start Motor Rotation: Updated description.</p> <p>Updated Adapting the Code Example to another Motor: Updated description.</p> <p>Updated Bridge Control Panel Monitor Tool Guide: Updated Upper Terminal Configuration Guide: Updated description.</p> <p>Updated Lower Terminal Configuration Guide: Updated description.</p> <p>Updated Reading the Motor Speed Using BCP Commands: Updated description.</p> <p>Updated Appendix: Updated Board Schematics, Board Layout, and BOM: Updated Figure A-1. Updated Bill of Materials: Updated Table A-1. Updated No Load Components: Updated Table A-2. Updated Special Installation Instructions: Updated Table A-3. Updated Label: Updated Table A-4.</p> <p>Completing Sunset Review.</p>
--	--	--	--	--