Document Number: BLDCDEMOUG
Rev. 1, 06/2016

BLDC Control Demo User's Guide

1. Introduction

This document provides instructions for running and controlling the Brushless DC (BLDC) sensorless application with the Freedom and Tower System development boards shown in Table 1.

The required software, hardware setup, jumper settings, project arrangement, and user interface are described in the following sections. For more information, see Section 8, “References”.

Contents

2. Supported Development Boards

There are three supported development boards with two Kinetis KV series motor-control MCUs for motor-control applications. The development boards and the supported MCUs are shown in the following table. The Tower System modular development platform and the Freedom development platform are targeted for low-voltage and low-power applications with BLDC control type.

Table 1. Supported development boards

—		Platform	
		Freedom	**Tower System**
Power stage		FRDM-MC-LVBLDC	TWR-LV3PH
MCU	**KV10**	FRDM-KV10Z	TWR-KV10Z32
	KV11	—	TWR-KV11Z75M
	KV31	FRDM-KV31F	TWR-KV31F120
	KV46	—	TWR-KV46F150M
	KV58	—	TWR-KV58F220M

3. Hardware Setup

The BLDC sensorless application runs on Tower System and Freedom development platforms with a default 24 V Linix motor.

3.1. Linix 45ZWN24-40 motor

The BLDC sensorless application uses the Linix 45ZWN24-40 motor described in the following table. The motor can be used with the Tower System or Freedom development platforms.

Table 2. Linix 45ZWN24-40 motor parameters

Characteristic	Symbol	Value	Units
Rated Voltage	V_t	24	V
Rated Speed @ V_t	—	4000	RPM
Rated Torque	T	0.0924	Nm
Rated Power	P	40	W
Continuous Current	I_{cs}	2.34	A
Number of Pole Pairs	pp	2	—

Figure 1. Linix motor 45ZWN24-40

The motor has two types of cable connectors. One cable has three wires and it is designated to power the motor. The second cable has five wires and it is designated for the Hall sensors signal sensing. Only the power input wires are needed for the BLDC sensorless application.

3.2. Tower System

To run the BLDC application using the Tower System, these Tower boards are required:

- Kinetis KV10Z Tower System module, (see TWR-KV10Z32), Kinetis KV11F75M Tower System module (see TWR-KV11F75M), or Kinetis KV31F Tower System module (see TWR-KV31F120M).
- 3-Phase Low-Voltage Motor Control module (see TWR-MC-LV3PH) including the Linix motor.
- Tower System elevator modules (see TWR-ELEV).

3.2.1. TWR-MC-LV3PH module

The 3-Phase Low-Voltage Motor Control module (TWR-MC-LV3PH) is a peripheral Tower System module, interchangeable across the Tower System. The phase voltage and current feedback signals are provided. These signals enable a variety of algorithms to control the 3-phase PMSM and BLDC motors. A high level of board protection (over-current, under-voltage, over-temperature) is provided by the MC33937 pre-driver. Before you insert the TWR-MC-LV3PH module into the Tower System, ensure that the jumpers on your TWR-MC-LV3PH module are configured as follows:

Table 3. TWR-MC-LV3PH jumper settings

Jumper	Setting	Function
J2	1-2	Selects the internal analog power supply.
J3	1-2	Selects the internal analog power reference (GND).
J10	2-3	Selects BEMF_SENSE_C.
J11	2-3	Selects BEMF_SENSE_B.
J12	2-3	Selects BEMF_SENSE_A.
J13	2-3	Selects I_SENSE_DCB.
J14	2-3	Selects V_SENSE_DCB_HALF.

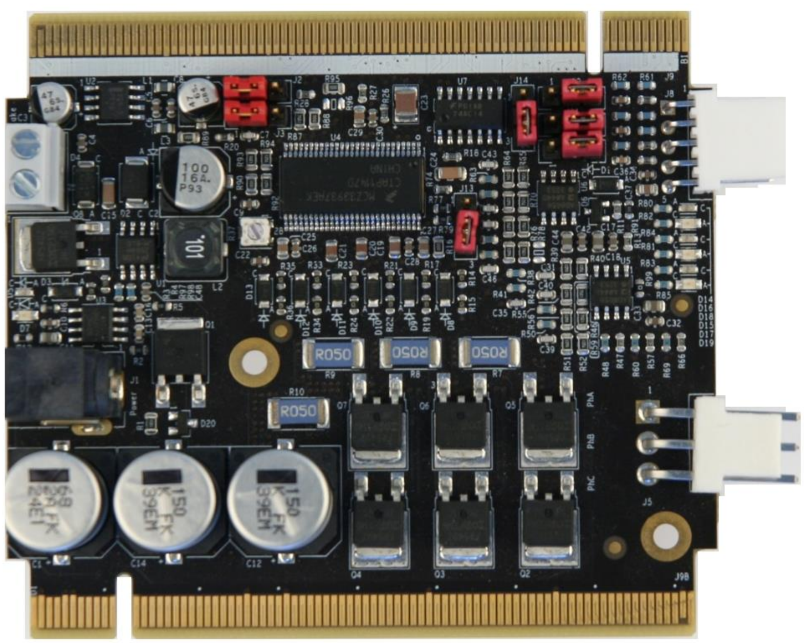

Figure 2. TWR-LV-MC3PH jumper settings

3.2.2. TWR-KV10Z32 MCU module

Configure the jumpers on the TWR-KV10Z32 MCU and TWR-MC-LV3PH modules properly. This table lists the relevant jumpers and their settings for the TWR-KV10Z32 MCU module:

Table 4. TWR-KV10Z32 jumper settings

Jumper	Setting	Function
J22	3-4	OpenSDA terminal port.
J21	3-4	OpenSDA terminal port.
J3	2-3	Connects PTB0 to the on-board push-button SW2.

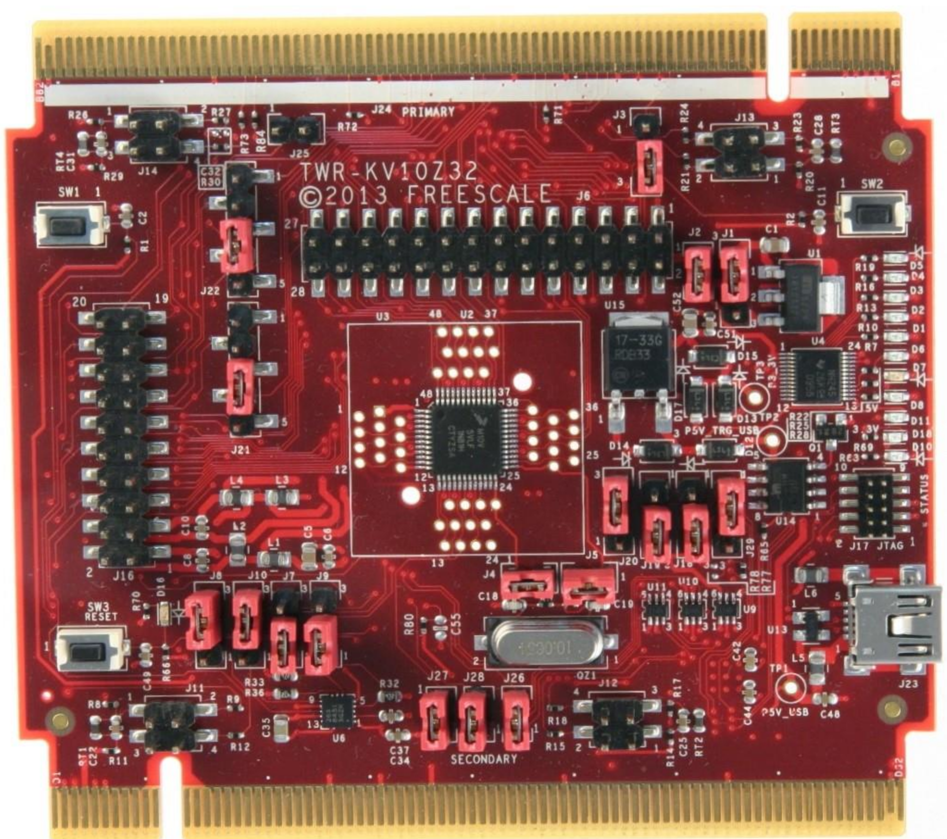

Figure 3. TWR-KV10Z32 MCU module

3.2.3. TWR-KV11Z75M MCU module

Configure the jumpers on the TWR-KV11Z75M MCU and TWR-MC-LV3PH modules properly. This table lists the relevant jumpers and their settings for the TWR-KV11Z75M MCU module:

Table 5. TWR-KV11Z jumper settings

Jumper	Setting	Jumper	Setting	Jumper	Setting
J1	1-2	J9	open	J512	1-2
J2	2-3	J10	1-2	J517, J518	J518-J517(2)
J4	2-3	J11	2-3	J519	1-2
J5	5-6, 7-8, 9-10	J12	1-2	J523	1-2
J6	open	J13	2-3	J524	open
J7	open	J14	1-2	J526	1-2
J8	open	J17	2-3	—	—
J505	2-3	J506	2-3	—	—

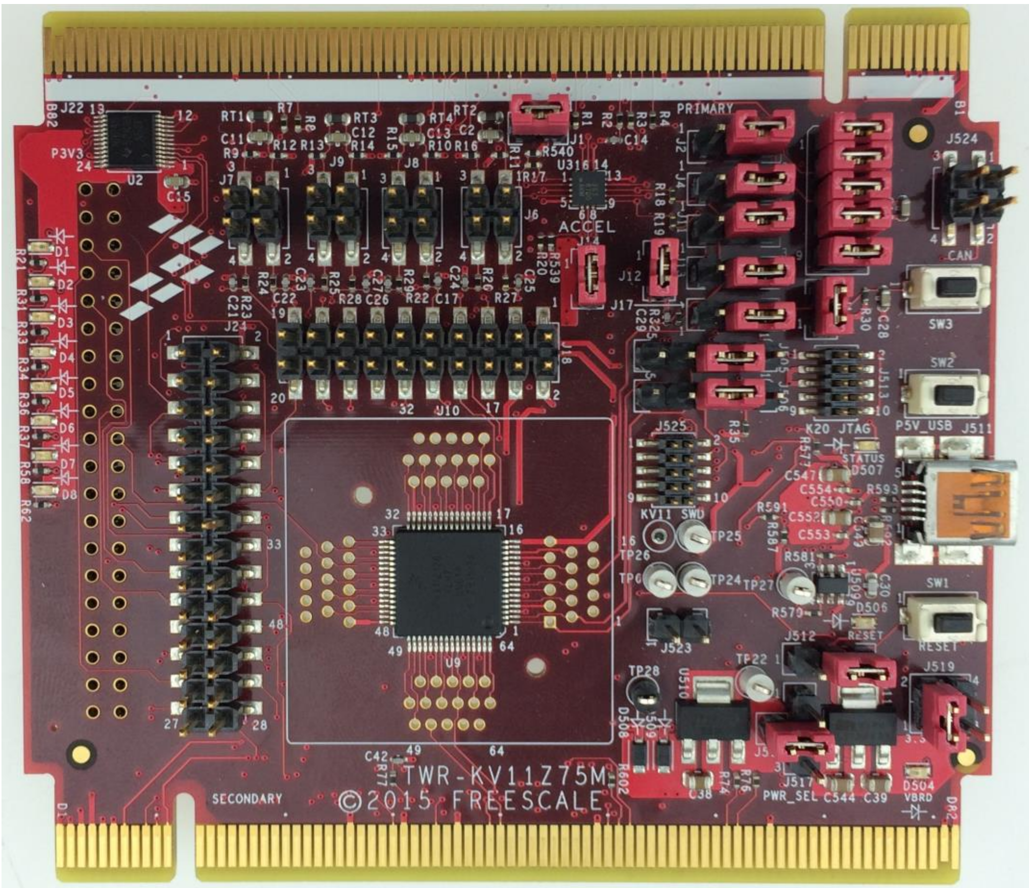

Figure 4. TWR-KV11Z75M MCU module

3.2.4. TWR-KV31F MCU module

Configure the jumpers on the TWR-KV31F120 MCU and TWR-MC-LV3PH modules properly. This table lists the relevant jumpers and their settings for the TWR-KV31F120 MCU module:

Table 6. TWR-KV10Z32 jumper settings

Jumper	Setting	Function
J22	2-3	OpenSDA terminal port.
J23	2-3	OpenSDA terminal port.

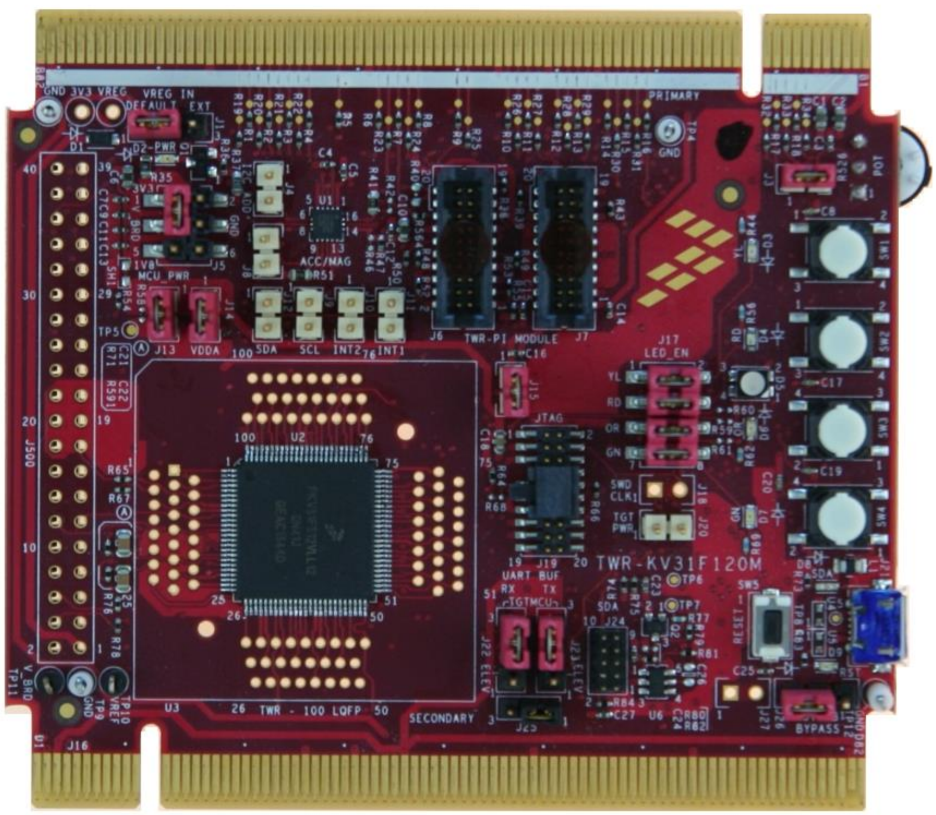

Figure 5. TWR-KV31F120M MCU module

3.2.5. TWR-KV46F150M MCU module

Configure the jumpers on the TWR-KV46F150 and TWR-MC-LV3PH modules properly. This table lists the relevant jumpers and their settings for the TWR-KV46F150 MCU module:

Table 7. TWR-KV10Z32 jumper settings

Jumper	Setting	Function
J505	3-4	OpenSDA terminal port.
J506	3-4	OpenSDA terminal port.

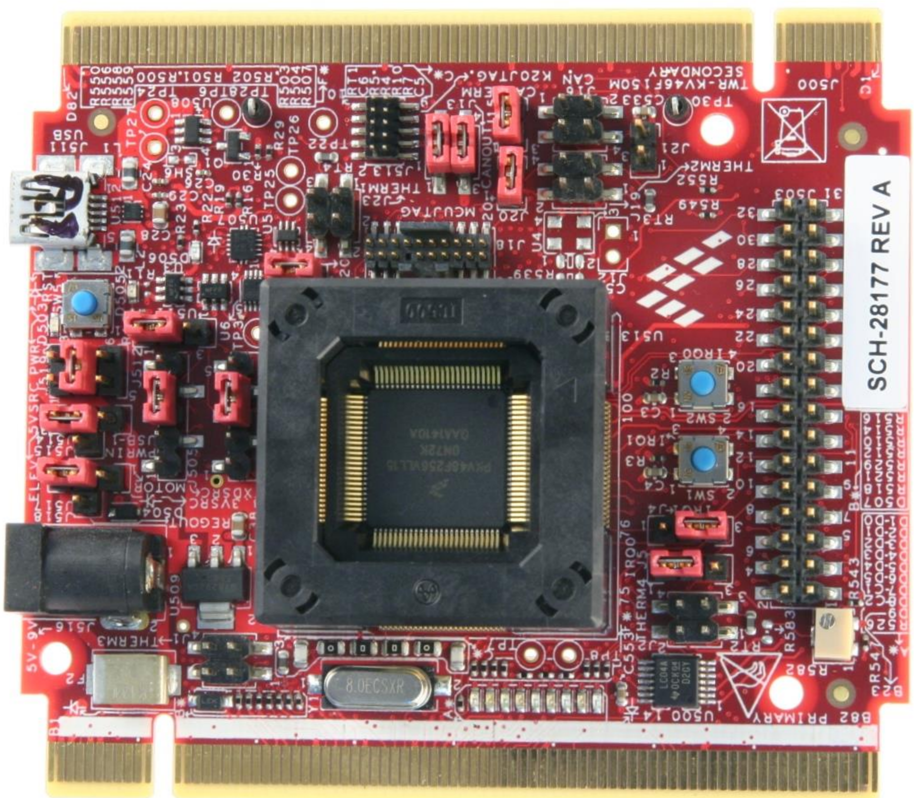

Figure 6. TWR-KV46F150M MCU module

3.2.6. TWR-KV58F Tower System module

The TWR-KV58F220M is a development module for the Kinetis KV5x family of MCUs built around the ARM® Cortex®-M7 core. This MCU has enough power for use in multi-motor control applications (such as the PMSM or BLDC motors with simple or advanced control techniques). The MCU has a wide range of motor-control peripherals, lots of memory (depending on the model used), and a powerful core.

To begin, configure the jumpers on the TWR-KV58F220M and TWR-MC-LV3PH Tower System modules properly. This table lists the specific jumpers and their settings for the TWR-KV58F220M Tower System module:

Table 8. TWR-KV58F jumper settings

Jumper	Setting	Jumper	Setting	Jumper	Setting
J1	1-2	J11	1-2	J23	2-3
J2	open	J12	1-2	J24	2-3
J3	1-2	J14	open	J25	2-3
J4	1-2	J17	open	J26	2-3
J7	1-2	J18	open	J28	1-2
J8	1-2	J19	open	J29	open
J9	open	J20	open	J30	1-2, 3-4, 5-6, 7-8
J10	1-2	J21	1-2	—	—

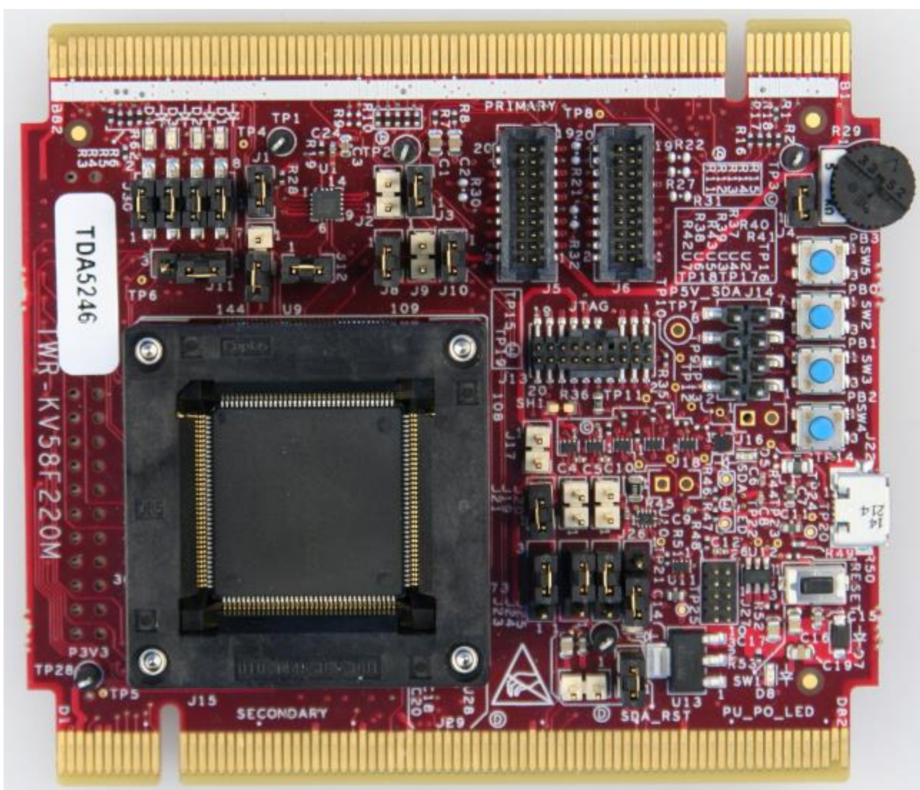

Figure 7. TWR-KV58F Tower System module

3.3. Tower System assembly

1. Insert the TWR-KVxxXxx MCU module and the TWR-MC-LV3PH module into the TWR-ELEV card slots. Ensure that the primary sides of the modules (marked by white stripes) are inserted into the primary elevator card (marked by white connectors).
2. After assembling the Tower System, connect the required cables as follows:
   - Connect the power input cable (3-wire connector) of the Linix motor to its corresponding connector (J5) on the TWR-MC-LV3PH motor control driver board.
   - Plug in the power supply cable that is attached to the TWR-MC-LV3PH system kit to the motor control peripheral board TWR-MC-LV3PH.
   - Connect the TWR MCU module to the host PC via a USB cable connected to J23 of the TWR-KV10Z32 MCU module or J21 of the TWR-KV31F120 MCU module and any USB port on the host PC.

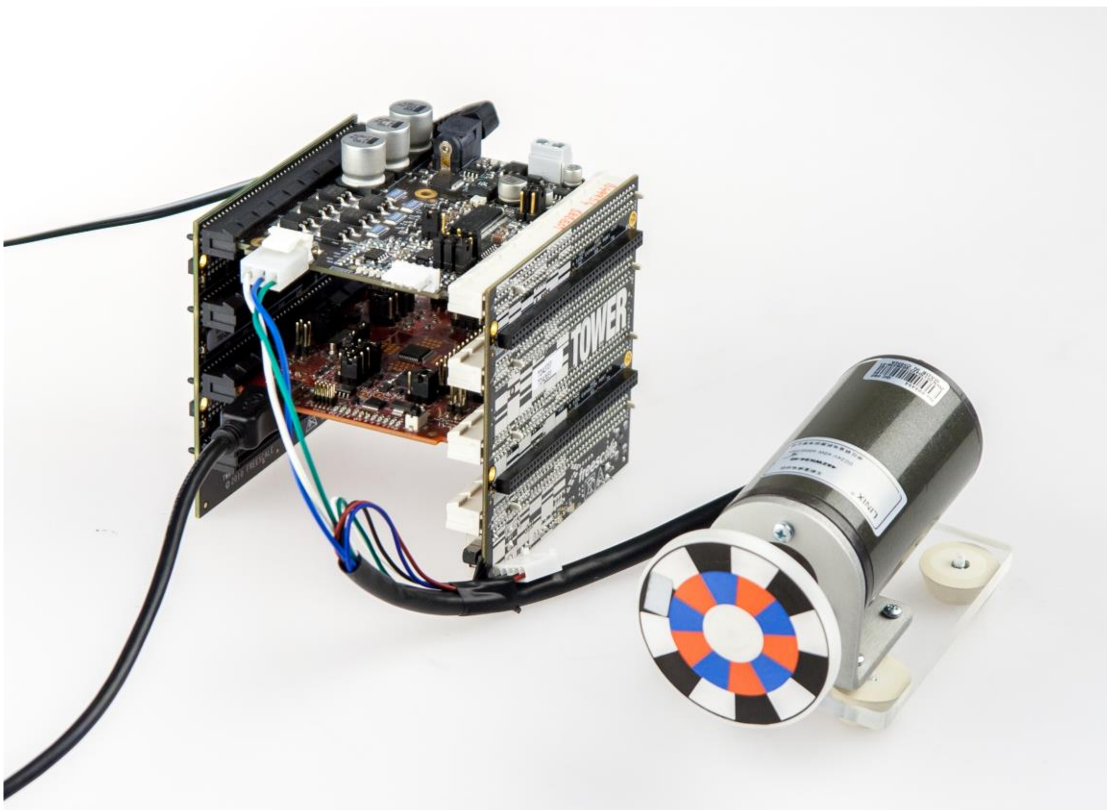

Figure 8. Assembled Tower System

3.4. Freedom development platform

To run the BLDC application using Freedom, you need these Freedom boards:

- Kinetis KV10Z Freedom board (see FRDM-KV10Z) or Kinetis KV31F Freedom board (see FRDM-KV31F).
- 3-Phase Low-Voltage Power Freedom shield (see FRDM-MC-LV3PH) including the Linix motor.

3.4.1. FRDM-MC-LVBLDC low-voltage evaluation board

The FRDM-MC-LVBLDC low-voltage evaluation board (in a shield form factor) effectively turns the Freedom development platform into a complete motor-control reference design compatible with the existing Freedom development platforms (FRDM-KV31F and FRDM-KV10Z).

The FRDM-MC-LVBLDC board does not require any hardware configuration or jumper settings. It contains no jumpers.

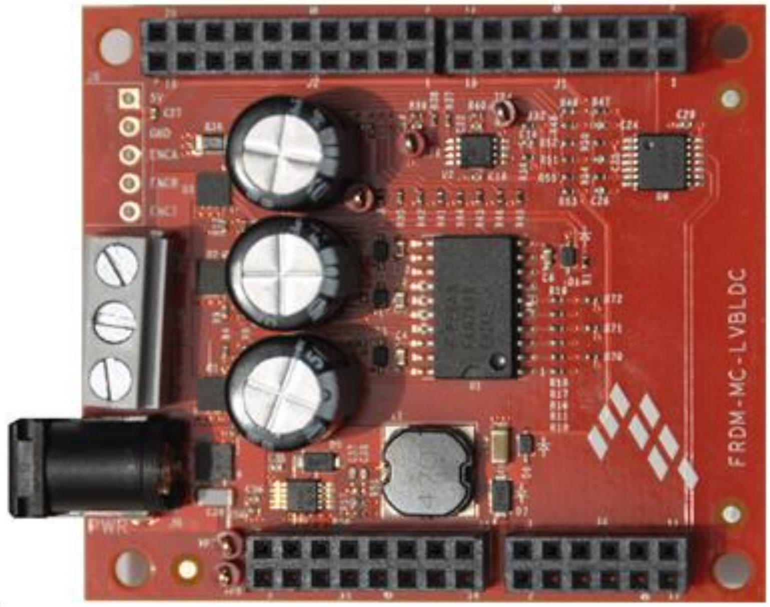

Figure 9. FRDM-MC-LVBLDC

3.4.2. FRDM-KV10Z board

The FRDM-KV10Z board is a low-cost development tool for the Kinetis V series KV1x MCU family built on the ARM Cortex-M0+ processor. The FRDM-KV10Z board's hardware is form-factor compatible with the Arduino™ R3 pin layout, providing a broad range of expansion board options. The FRDM-KV10Z platform features OpenSDA, the Freescale open source hardware embedded serial and debug adapter running an open source bootloader.

The FRDM-KV10Z board does not require any hardware configuration or jumper settings. It contains no jumpers.

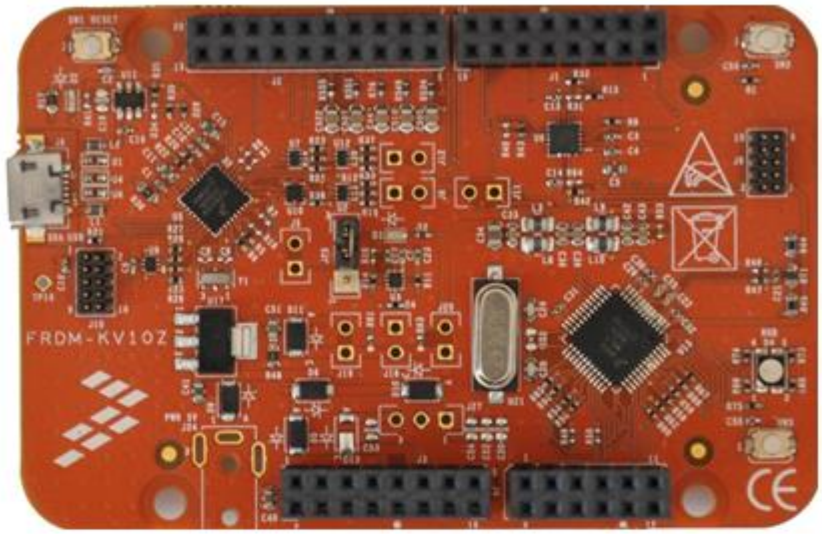

Figure 10. FRDM-KV10Z Freedom development board

3.4.3. FRDM-KV31F board

The FRDM-KV31F board is a low-cost development tool for the Kinetis V series KV3x MCU family built upon the ARM Cortex-M4 processor. The FRDM-KV31F board hardware is form-factor compatible with the Arduino™ R3 pin layout, providing a broad range of expansion board options, including FRDM-MC-LVPMSM and FRDM-MC-LVBLDC for PMSM and BLDC motor control.

The FRDM-KV31F platform features OpenSDA, the open-source hardware embedded serial and debug adapter running an open-source bootloader. This circuit offers several options for serial communication, flash programming, and run-control debugging.

The FRDM-KV31F board does not require any hardware configuration or jumper settings. It contains no jumpers.

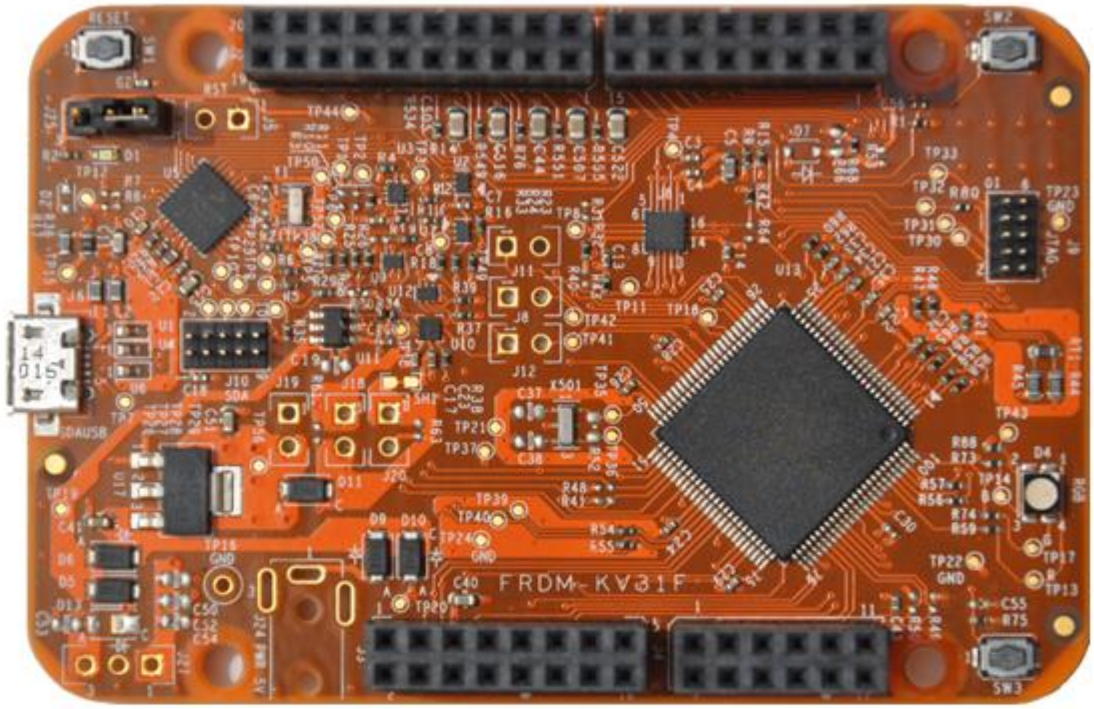

Figure 11. FRDM-KV31F Freedom development board

3.4.4. Freedom development platform assembly

1. Connect the FRDM-MC-LVBLDC shield on top of the FRDM-KVxxx board.
2. Connect the BLDC motor 3-phase wires into the screw terminals on the board.
3. Plug in the USB cable from the USB host to the OpenSDA micro USB connector.
4. Plug in a 12 V DC power supply to the DC power jack.

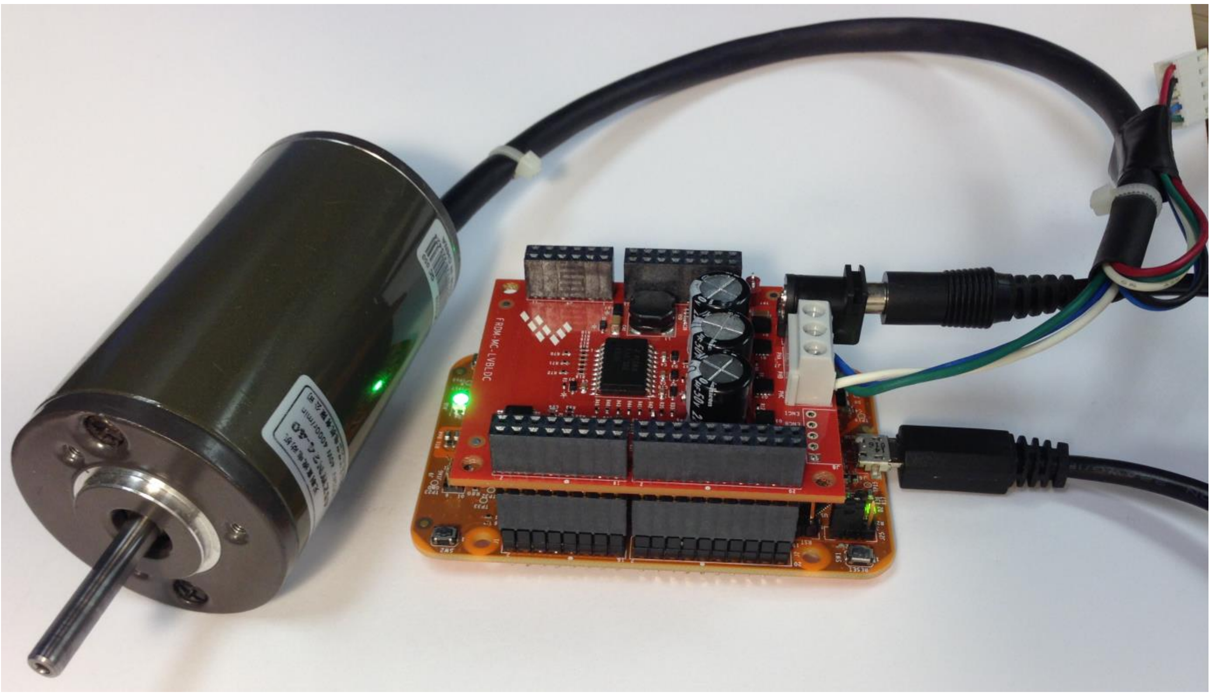

Figure 12. Assembled Freedom system

4. Project File Structure

The demo project folder (for example *boards\frdmkv10z\demo_apps\mc_bldc*) contains these folders and files:

- *IAR* folder—contains the configuration files for IAR Embedded Workbench® IDE. If IAR Embedded Workbench for ARM is installed on your computer, open the project using IAR IDE.
- *KDS* folder—contains the configuration files for KDS IDE and the launch configuration for debuggers. If KDS is installed on your computer, open the project using KDS IDE.
- *MDK* folder—contains the configuration files for μVision® Keil® IDE. If Keil IDE is installed on your computer, open the project using Keil IDE.
- *Project files*—contains the device-specific files. They specify the peripheral initialization routines, motor definitions, and state machines. The source code contains a lot of comments. The functions of the particular files are explained in this list:
    - *m1_bldc_appconfig.h*—contains definitions of constants for the application control processes (parameters of the motor and regulators, and the constants for BLDC sensorless control-related algorithms).
    - *main.c*—contains basic application initialization (enabling interrupts), subroutines for accessing the MCU peripherals, and interrupt service routines.
    - *main.h*—header file for *main.c*.
    - *app_init.c*—contains the application initialization and initialization functions for buttons and LEDs.

- *app_init.h*—header file for *app_init.h*; contains LED macros.
- *mcdrv_< board&MCU >.c*—contains motor-control driver peripherals initialization functions, specific for the board and MCU used.
- *mcdrv_< board&MCU >.h*—header file for *mcdrv_<board&MCU>.c*. This file contains macros for changing the PWM period and ADC channels assigned to the phase currents and board voltage.

The motor-control folder *<KSDK_install_folder>\middleware\motor_control\bldc* contains these common source and header files used in all motor-control projects. The folder contains the subfolders common to the entire project in this package:

- *mc_algorithms*—contains the control algorithms used to control the BLDC motor.
- *mc_drivers*—contains the source and header files used to initialize and run motor-control applications.
- *mc_state_machine*—contains the software routines that are executed when the application is in a particular state or state transition.
- *state_machine*—contains the state machine functions for the Fault, Initialization, Stop, and Run states.

Each motor-control project is based on RTCESL (Real-Time Control Embedded Software Library) placed in the *<KSDK_install_folder>\middleware* folder. The library contains the mathematical functions used in the project. It contains the library subfolders for specific cores ("*cm0*", "*cm4*", and "*cm7*"). This subfolder includes the required header files and library files used in the project. The *RTCESL* folder is taken from the RTCESL release 4.3, and it is fully compatible with the official release. The library names are changed for easier use with the available IDEs. See nxp.com/fslesl for more information about RTCESL.

5. Tools

The following list shows the required software to be installed on your PC to run and control the BLDC sensorless application properly.

- IAR Embedded Workbench IDE v7.60 or higher
- Kinetis Design Studio IDE v3.2 or higher
- ARM-MDK Keil µVision version 5.20

6. Application Building and Debugging

The package contains projects for Kinetis Design Studio, IAR Embedded Workbench, and µVision Keil IDEs. Both of them are targeted for motor-control applications. The release configuration is the default one, and there are no special requirements needed to run and debug the demonstration applications.

6.1. IAR Embedded Workbench IDE

Use IAR Embedded Workbench IDE to compile and run the demonstration projects. The first step is to choose the demonstration, development board, and MCU. For example, to run a demonstration project for the Freedom development platform and Kinetis KV10 MCU, the project is located in the *<KSDK_install_folder>\boards\frdmkv10z\demo_apps\mc_bldc\iar* folder, which contains all necessary files. Double-click "*mc_bldc.eww*" to run this project. This figure shows the IAR workspace:

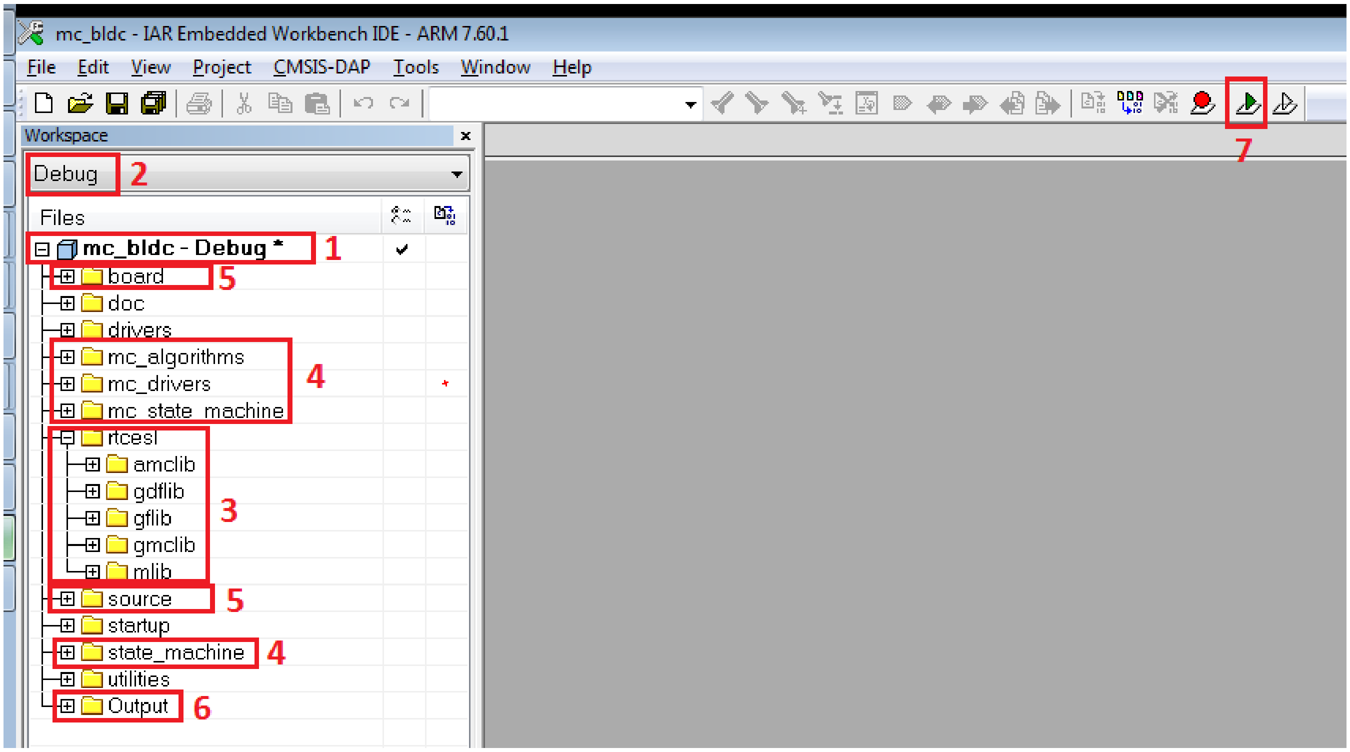

Figure 13. IAR Embedded Workbench IDE

The project opened in IAR Embedded Workbench is fully configured and includes all source and header files required by the application (such as the startup code, clock configuration, and peripherals' configuration). You can choose from two compiling conditions ("debug" or "release") shown in Figure 13 point 2. Each of the two conditions has its own setting:

- "debug"—used for debugging, optimization has the "None – turned off" flag.
- "release"—used for releasing, optimization has the "High – highest optimization for speed" flag.

NOTE

The "debug" condition has the optimization turned off, and the output file may not fit into MCUs with a smaller flash (for example KV10Z32).

The source code shown in Figure 13 includes these source files and folders:

- Point 3—the RTCESL library source folder contains header files for the mathematical and control functions used in this project. The theory about using and applying these functions is described in the user's guides specific for each library. Find the user's guides at nxp.com/fslesl.

- Point 4—the board independent source files contain the application source code. These files are placed in the *<KSDK_install_folder>\middleware\motor_control\bldc* folder.
- Point 5—the device-specific files contain the application source code. These files are placed in the *<KSDK_install_folder>\boards\<board&MCU>\demo_apps\mc_bldc* folder.
- Point 6—shows the output file generated by the compiler, and is ready to use with the default debugger (P&E Micro OpenSDA). This debugger is set as default for the Tower System boards, and can be changed in the project options by right-clicking Point 1, selecting “Options”, and clicking “Debugger”. Start the project debugging by clicking Point 7 (Figure 13).

The installation package contains only the required project files. All cached files are deleted by default. From IAR IDE version 7.60 onwards, the updated P&E Micro OpenSDA driver requires selecting a valid device (see the following figure) during the first code build and loading to the MCU.

Figure 14. IAR IDE select device alert

After you click the “OK” button, the “P&E Connection Manager” opens. Click the “Select New Device” button and select the particular MCU according to Table 9.

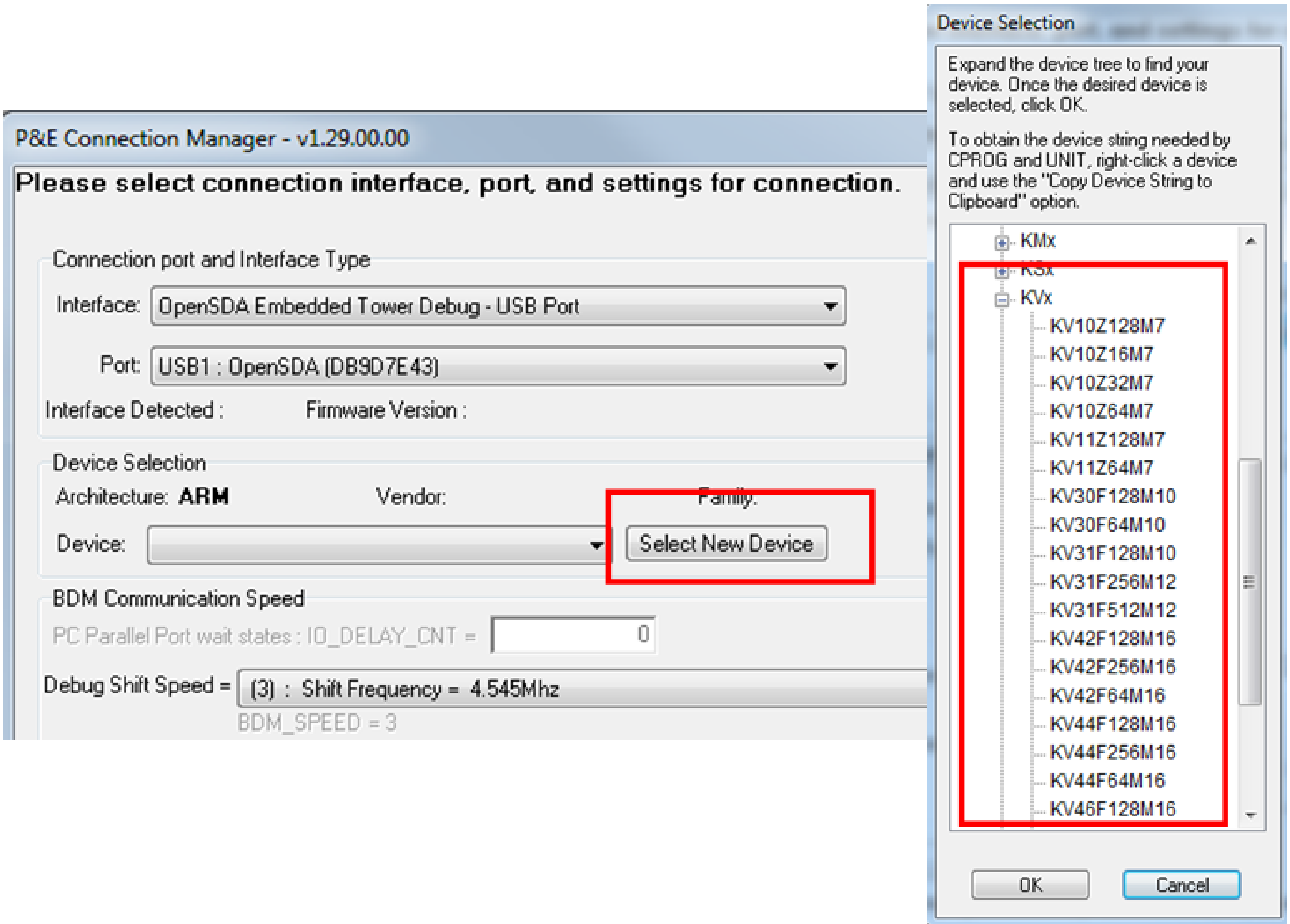

Figure 15. P&E Connection Manager

Table 9. Supported development platforms

—		Platform	
		Tower	**Freedom**
MCU	**KV10Z**	KV10Z32M7	KV10Z32M7
	KV31F	KV31F512M12	KV31F512M12
	KV11Z	KV11Z128M7	—
	KV46F	KV46F256M16	—
	KV58F	KV58F512M22	—

6.2. Kinetis Design Studio (KDS)

Kinetis Design Studio (KDS) is an IDE tool that you can use to develop and test software for NXP MCUs. It supports a wide range of Kinetis devices, such as the powerful K series, low-power KL series, and KV series targeted for motor control. KDS includes tools for compiling, linking, and debugging of source code. KDS supports a wide range of debuggers, such as P&E Micro or J-Link™

(and others). Download the latest release of KDS from the official NXP website (nxp.com/kds). For installation and configuration, see *Kinetis Design Studio V3.2.0 User's Guide* (document KDSUG).

To open a demonstration, choose the development board and MCU. For example, if you want to open a demonstration project for the Freedom development platform and Kinetis KV10 MCU, locate the project in *<KSDK_install_folder>\boards\frdmkv10z\demo_apps\mc_bldc\kds*, run KDS IDE from the default installation path or from the installed programs, and perform these steps:

- Click the "File" menu in the top-left corner of the IDE, and select "Import…".
- A window opens. Highlight "Existing Projects into Workspace" in the "General" folder, and click the "Next" button:

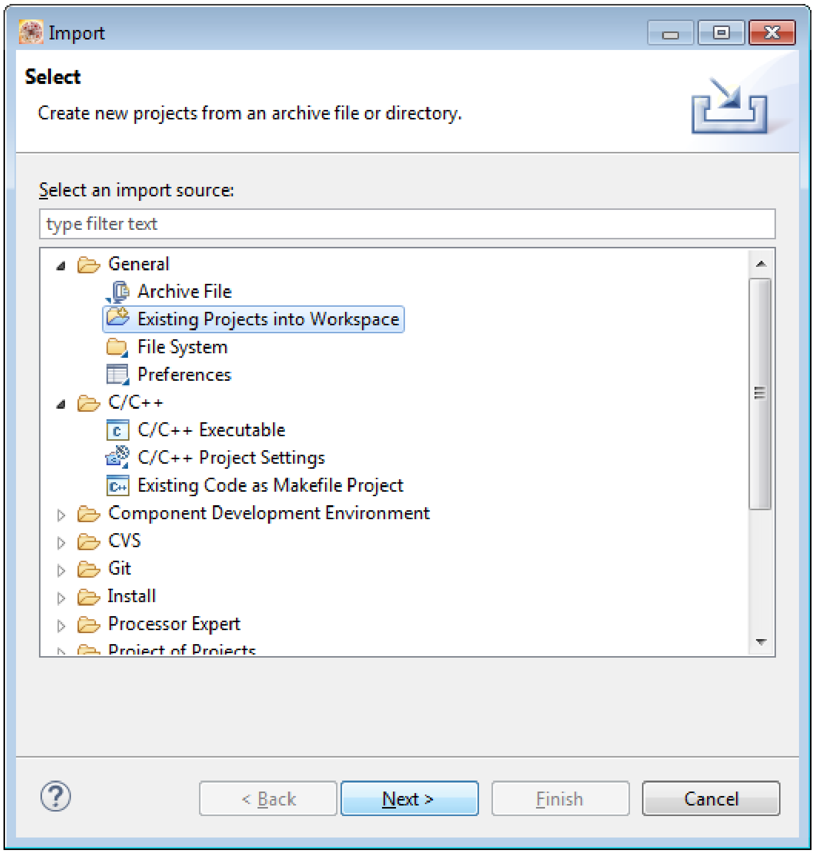

Figure 16. KDS—importing project

- The "Import" window opens. Click the "Browse" button, and then locate the project in the *<KSDK_install_folder>\boards\frdmkv10z\demo_apps\mc_bldc\kds* folder. Click the "OK" button.

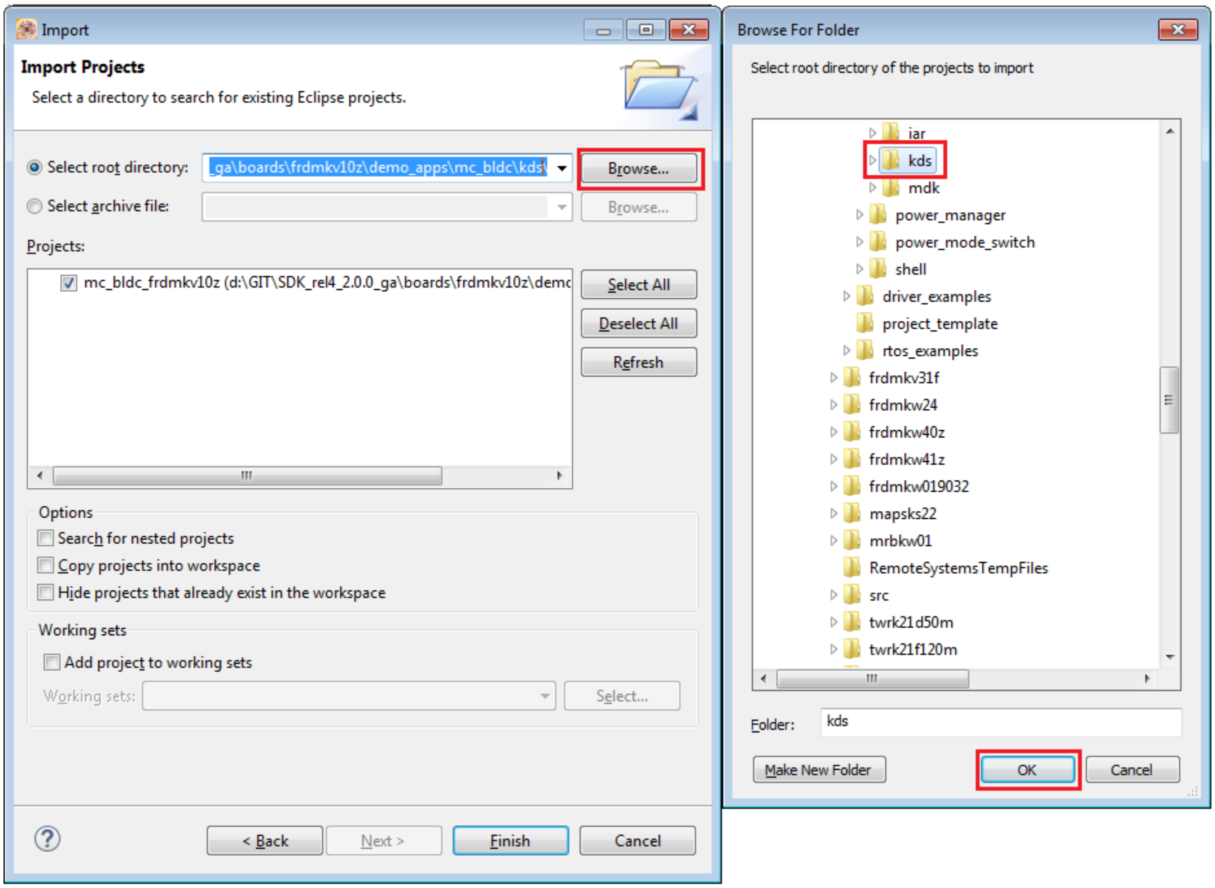

Figure 17. KDS import project 3

- Confirm the project by clicking the “Finish” button.

The project is now imported to Kinetis Design Studio (Figure 18). Point 1 shows the imported project in “Project Explorer”, and Point 2 shows the source code of this project. Build the project by clicking the “build” icon (Point 3) where the “release” configuration is set as default. You can change the configuration to “debug”. Each of these two conditions has its own setting:

- “debug”—used for debugging, optimization has the “None – turned off” flag.
- “release”—used for releasing, optimization has the “High – Highest optimization for speed” flag.

NOTE

The “debug” condition has the optimization turned off, and the output file may not fit into MCUs with a smaller flash (for example KV10Z32).

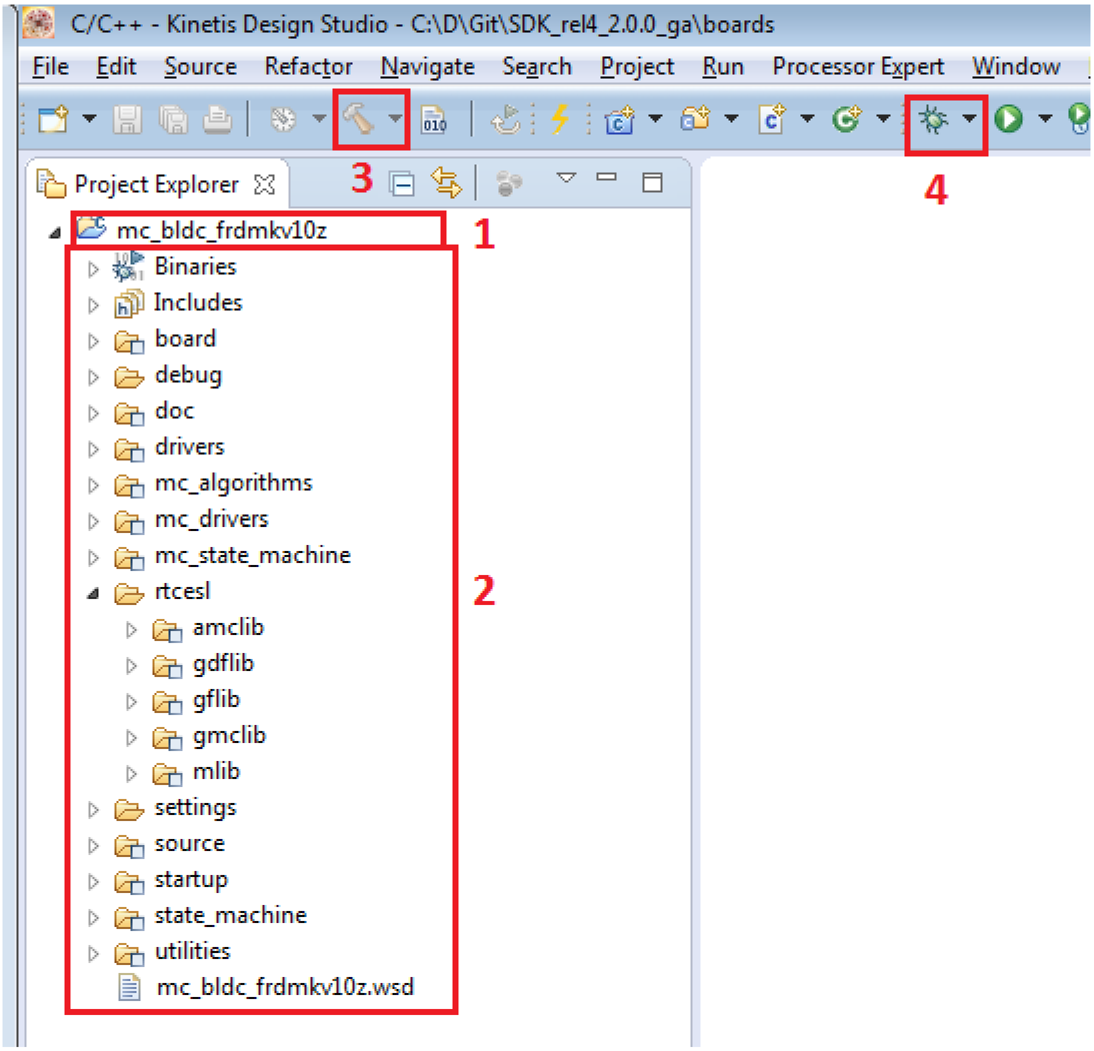

Figure 18. KDS BLDC project

After the project is compiled, either the *debug* or *release* folder is created, depending on the build condition selected. An **.elf* binary file is created in one or both of those folders. Now use the debugger (Point 4 in Figure 18). You can use the predefined debugger (P&E Micro OpenSDA), or choose a different debugger from the menu. In the top list menu, select "Run-> Debug Configuration" to define a different type of debugger, or change the conditions of debugging (such as the optimization level).

6.3. ARM-MDK Keil µVision

The ARM-MDK Keil µVision IDE (Keil) is a software development and testing tool for various MCUs. It supports a wide range of Kinetis devices, such as the powerful K series, low-power KL series, and KV series targeted for motor control. Keil includes tools for compiling, linking, and debugging of source code. Keil supports a wide range of debuggers, such as P&E Micro or J-Link (and others). Download the latest release of Keil from the official Keil website www2.keil.com/mdk5/uvision.

To open the demonstration, choose the development board and MCU. For example, to open a reference project for the Freedom development platform and Kinetis KV10 MCU, locate the project in the *<KSDK_install_folder>\boards\frdmkv10z\demo_apps\mc_bldc\mdk* folder:

- Double click the *mc_bldc.uvprojx* project file.
- After the project is opened, click the “Build” button (Point 1) to compile the project. Click the “Download” button (Point 2) to download the code to the target. Then click the “Debug” button to enter the debug session (Point 3):

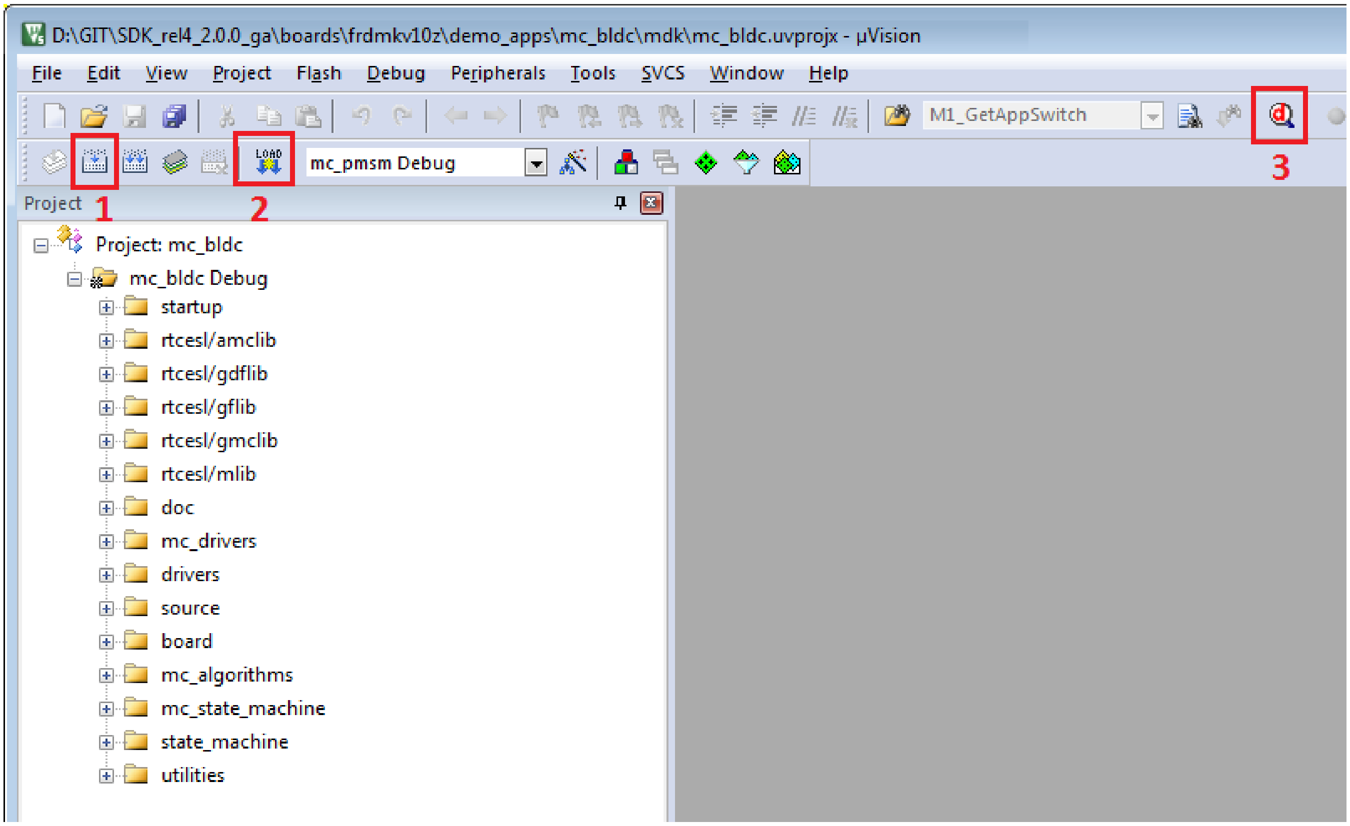

Figure 19. Keil BLDC project

- Before downloading the code to the target, select the proper target device (if using the P&E Micro OpenSDA debugger).

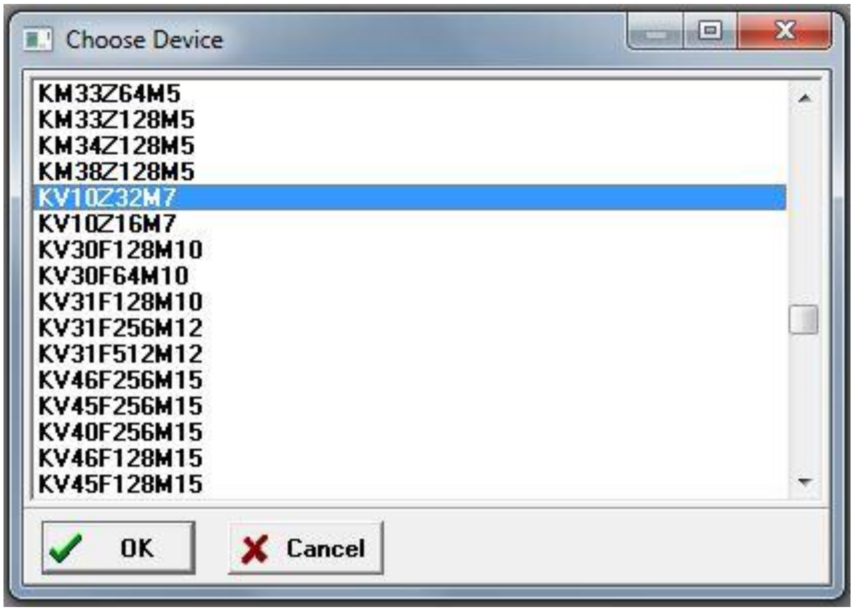

Figure 20. P&E Micro debugger—target selection

- There are two project configurations:
  - "debug"—used for debugging, the optimization has the "None – turned off" flag.
  - "release"—used for releasing, the optimization has the "High – Highest optimization for speed" flag.
- Use the predefined debugger (such as P&E Micro OpenSDA), or choose a different debugger from the menu. In the top list menu, select "Run-> Debug Configuration" to define a different type of debugger or change the conditions of debugging (such as the optimization level).

7. User Interface

The application contains the demo application mode to demonstrate the motor rotation. Operate it using the user button. The Tower System and Freedom boards include a user button associated with the port interrupt (generated whenever one of the buttons is pressed). At the beginning of the ISR, a simple logic executes, and the interrupt flag clears. When you press the SW2 button, the demo mode starts; when you press the same button again, the application stops and transitions back to the STOP state. There is also an LED indication of the current application state. The green continuous LED indicates that the application is in the RUN state, the flashing LED indicates the FAULT state, and the LED off (or red LED) indicates the STOP state.

Control the application using the buttons on the NXP Kinetis V Tower System and Freedom development boards.

7.1. Control button

When you press the SW2 button, the demonstration mode switches on (or off, if it is already switched on).

8. References

See these documents at nxp.com:

1. *Kinetis Design Studio v3.2.0 User's Guide* (document KDSUG)
2. Embedded Software Libraries User's Guides
3. *3-Phase BLDC Sensorless Motor Control Application* (document DRM144)
4. *Sensorless BLDC Control on Kinetis KV* (document AN5263)

9. Revision History

Table 10. Revision history

Revision number	Date	Substantive changes
0	06/2016	Initial release.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: BLDCDEMOUG
Rev. 1
06/2016