Freescale Semiconductor, Inc.
User's Guide

Document Number: AN5205
Rev. 0, 10/2015

BLDC Sensorless Application Package User's Guide

Contents

1. Introduction

This document provides a step by step guide how to run and control BLDC sensorless application with Freescale Freedom and Tower development boards mentioned in Table 1.

Required software, Hardware setup, jumper settings, project arrangement and user interface description is obtained in following chapters.

Table 1. **Supported Development Boards**

		Supported Development Boards		
Platform		**FRDM**	**TWR**	**HVP**
Power Stage		FRDM-MC-LVBLDC	TWR-LV3PH	HVP-MC3PH
MCU	**KV10**	FRDM-KV10Z	TWR-KV10Z32	
	KV11		TWR-KV11Z75M	
	KV31	FRDM-KV31F	TWR-KV31F120	

2. Reference Documents

Table 1 provides a list of reference documents. All of these documents are available online at freescale.com.

Table 2. **Reference Documents**

Filename	Description
Kinetis Design Studio v3.0.0 – User's Guide	This document provides overview and detailed information about KDS 3.0 IDE.
Freescale Embedded Software Libraries (FSLESL) User's Guides	This set of user's guides provides detailed information about Freescale Embedded Software Libraries (FSLESL)

3. Hardware setup

The BLDC sensorless application runs on Freescale Tower and Freedom development platforms with default 24 Volt Linix Motor.

3.1. Motor Linix 45ZWN24-40

Motor Linix 45ZWN24-40 described in Table 3 is used by the BLDC Sensorless application. The motor is offered to Freescale Tower or Freedom development platform.

Table 3. **Motor Linix 45ZWN24-40 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 (cables) connectors. One cable has 3 wires and is designated to power the motor. The second cable has 5 wires and is designated for Hall sensors signal sensing. For BLDC sensorless application the power input wires are only needed.

3.2. Tower System

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

- Kinetis KV10Z Tower module (TWR-KV10Z32) or Kinetis KV11F75M Tower module (TWR-KV11F75M) or Kinetis KV31F Tower module (TWR-KV31F120M)
- Three-phase low voltage power module (TWR-MC-LV3PH) with included Linix motor
- Tower elevator modules (TWR-ELEV)

All modules of the Tower system are available for order via the Freescale web page or from distributors, so the user can easily build the hardware platform for which the application is targeted.

3.2.1. TWR-MC-LV3PH Peripheral module

The 3 phase Low-Voltage Motor Control board (TWR-MC-LV3PH) is a peripheral Tower System Module, interchangeable across the Tower development platform. Phase voltage and current feedback signals are provided. These signals allow a variety of algorithms to control 3phase PMSM and BLDC motors. High level of board protection (overcurrent, undervoltage, overtemperature, etc.) provided by MC33937 predriver. Before inserting the TWR-MC-LV3PH peripheral card into the Tower System, you will want to make sure the jumpers on your TWR-MC-LV3PH are configured as follows:

Table 4. **Jumper settings on TWR-MC-LV3PH**

Jumper	Setting	Function
J2	1-2	Selects internal analog power supply
J3	1-2	Selects 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

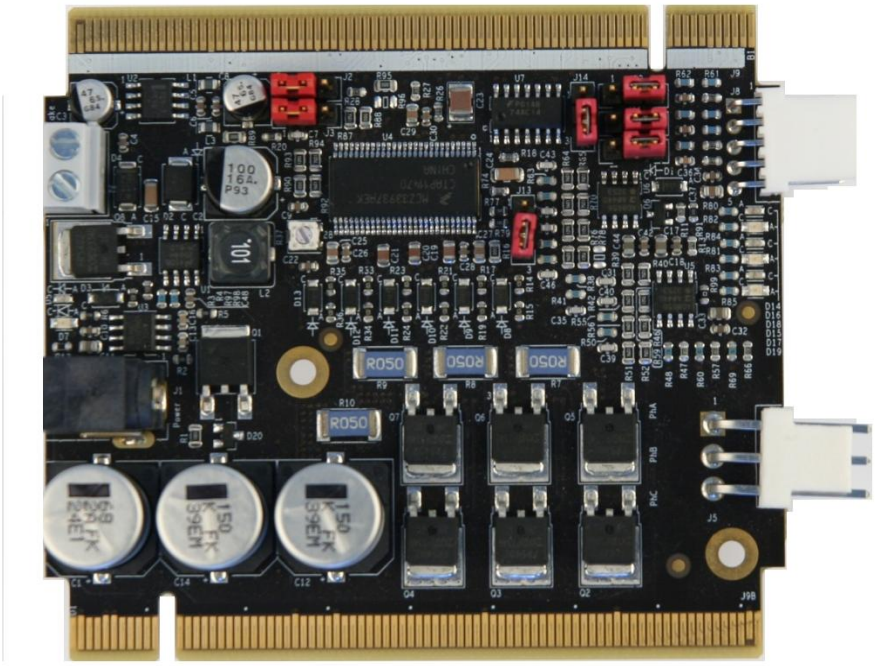

Figure 2. TWR-LV-MC3PH Jumper setting for BLDC

3.2.2. TWR-KV10Z32 MCU module

To begin, the jumpers of the TWR-KV10Z32 MCU module and the TWR-MC-LV3PH must be configured correctly. The following table lists the pertinent jumpers and their settings for the TWR-KV10Z32 MCU module.

Table 5. **TWR-KV10Z32 MCU Module jumpers settings**

Jumper	Setting	Function
J22	3-4	Selects UART1 for FreeMASTER connection through the OpenSDA terminal port
J21	3-4	Selects UART1 for FreeMASTER connection through the OpenSDA terminal port
J3	2-3	Connects PTB0 with on-board pushbutton SW2

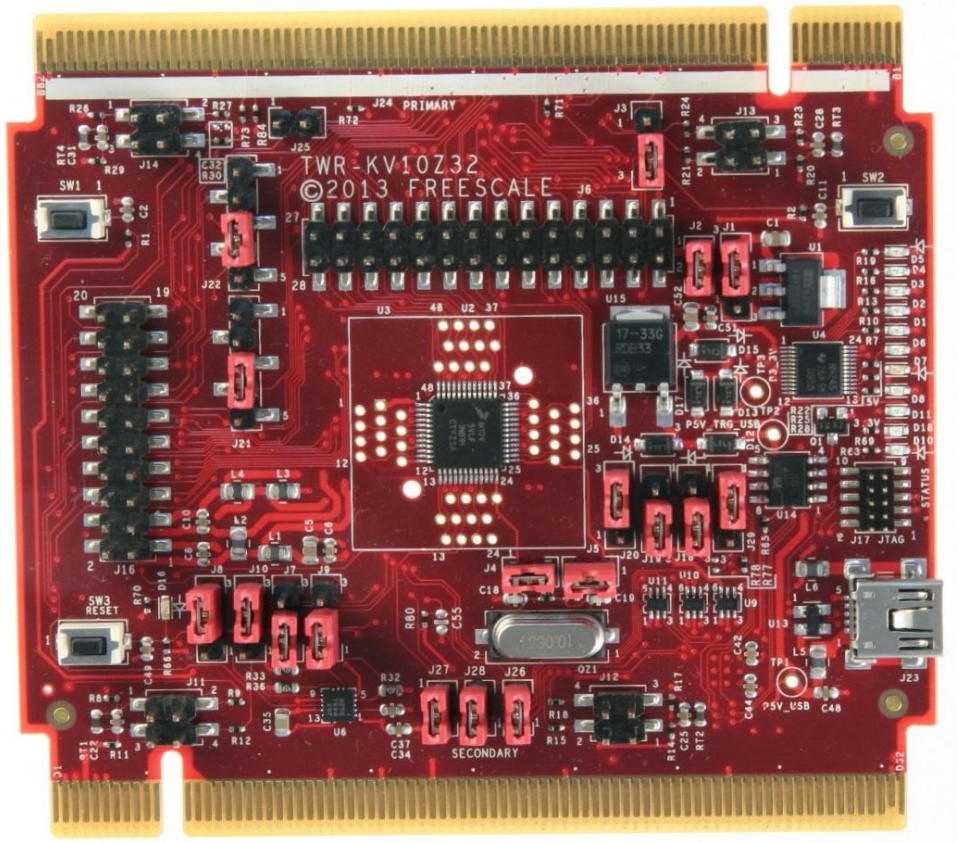

Figure 3. TWR-KV10Z32 MCU module

3.2.3. TWR-KV11Z75M MCU module

To begin, the jumpers of the TWR-KV11Z75M MCU module and the TWR-MC-LV3PH must be configured correctly. The following table lists the pertinent jumpers and their settings for the TWR-KV11Z75M MCU module.

Table 6. **TWR-KV11Z32 MCU Module jumpers settings**

Jumper	Setting	Function
J505	2-3	Selects UART1 for FreeMASTER connection through the OpenSDA terminal port
J506	2-3	Selects UART1 for FreeMASTER connection through the OpenSDA terminal port
J3	2-3	Connects PTB0 with on-board pushbutton SW2

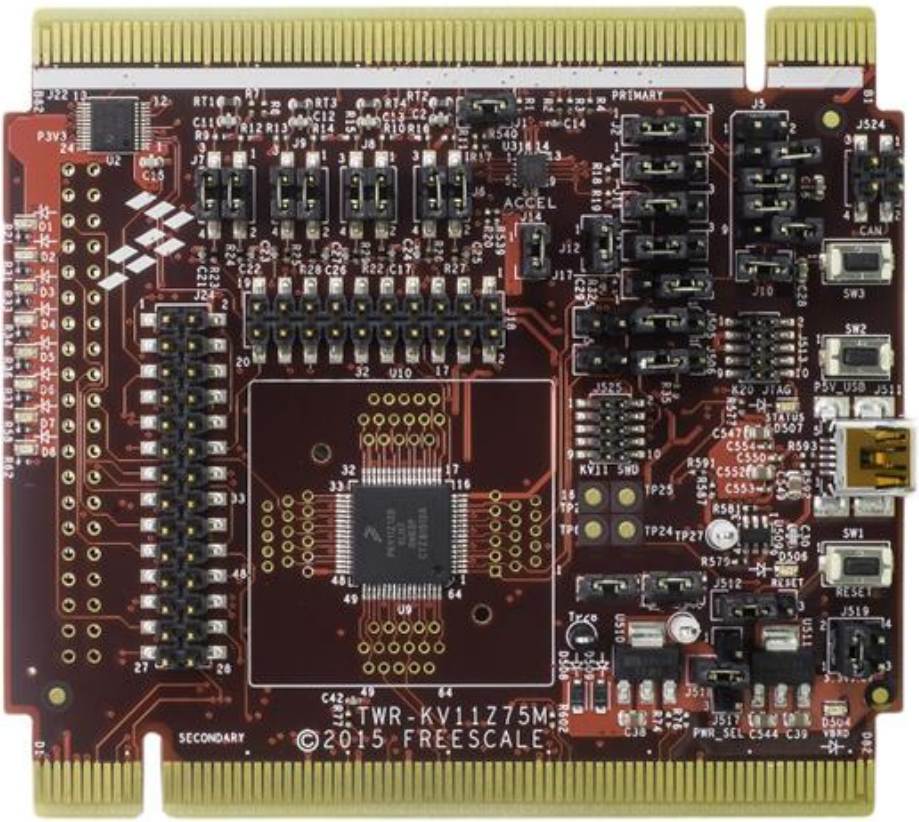

Figure 4. TWR-KV11Z75M MCU module

3.2.4. TWR-KV31F MCU module

To begin, the jumpers of the TWR-KV31F120 MCU module and the TWR-MC-LV3PH must be configured correctly. The following table lists the pertinent jumpers and their settings for the TWR-KV31F120 MCU module.

Table 7. **TWR-KV10Z32 MCU Module jumpers settings**

Jumper	Setting	Function
J22	2-3	Selects UART0 for FreeMASTER connection through the OpenSDA terminal port
J23	2-3	Selects UART0 for FreeMASTER connection through the OpenSDA terminal port

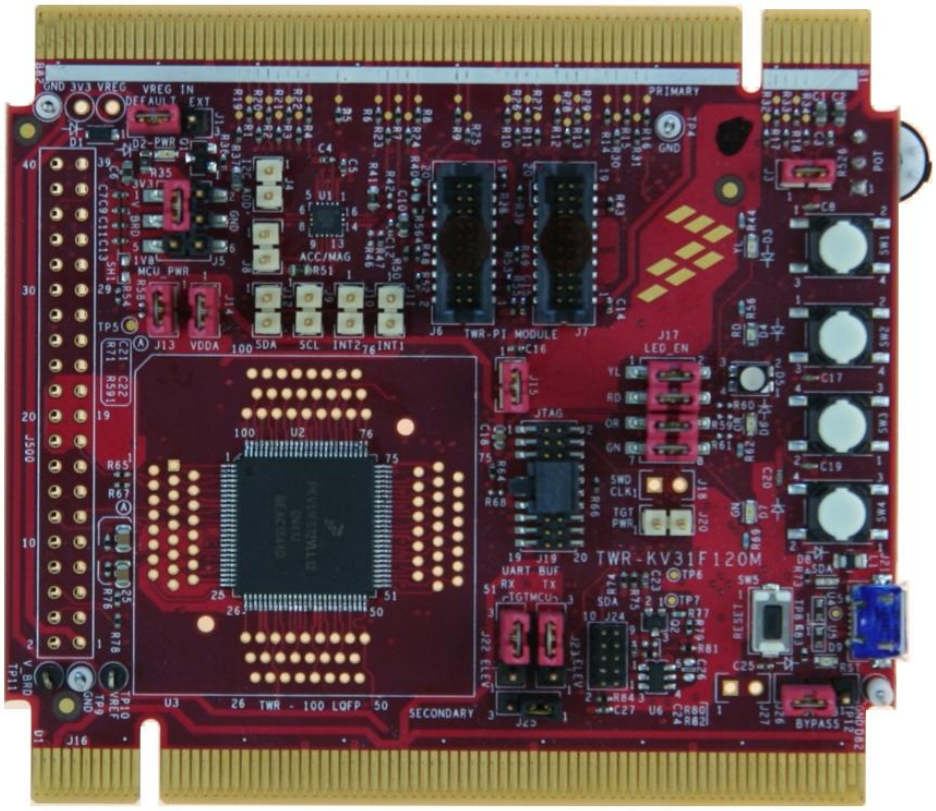

Figure 5. TWR-KV31F120M MCU module

3.2.5. Tower System Assembling

1. Insert the TWR-KVxxXxx MCU module and the TWR-MC-LV3PH peripheral module into the TWR-ELEV cards. Take care to ensure that the primary sides of the modules (marked by a white stripe) are inserted into the primary elevator card (marked by white connectors).
2. After Tower system assembling completion, connect required cables as follows:
   - Connect 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 cable of the power supply 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 6. Assembled Tower System

3.3. Freedom System

To run the BLDC application using the Freescale Freedom System, following Freedom boards are required:

- Kinetis KV10Z Freedom board (FRDM-KV10Z) or Kinetis KV31F Freedom board (FRDM-KV31F)
- Three-phase low voltage power Freedom shield (FRDM-MC-LV3PH) with included Linix motor

All modules of the Freedom system are available for order via the Freescale web page or from distributors, so the user can easily build the hardware platform for which the application is targeted.

3.3.1. FRDM-MC-LVBLDC

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

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

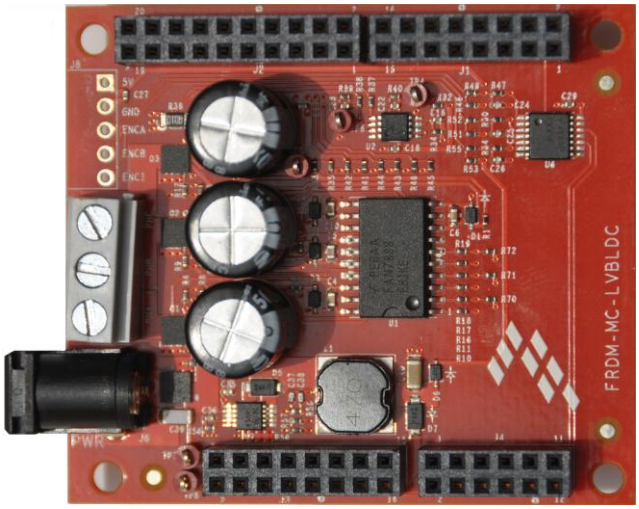

Figure 7. FRDM-MC-LVBLDC

3.3.2. FRDM-KV10Z

The FRDM-KV10Z is a low-cost development tool for the Kinetis V series KV1x MCU family built on the ARM® Cortex®-M0+ processor. The FRDM-KV10Z 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.

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

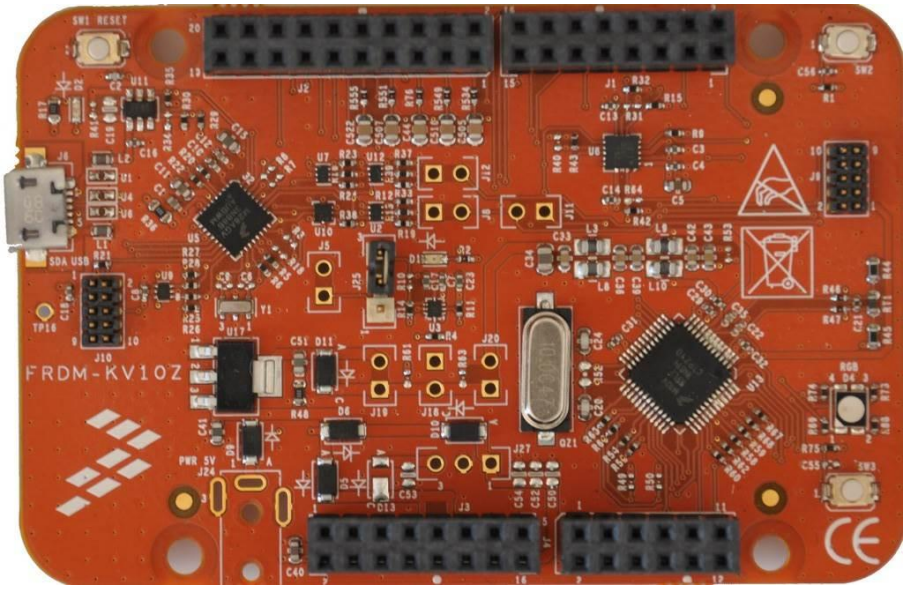

Figure 8. FRDM-KV10Z Freescale Freedom Development board

3.3.3. FRDM-KV31F

The FRDM-KV31F is a low-cost development tool for the Kinetis V series KV3x MCU family built on the ARM® Cortex®-M4 processor. The FRDM-KV31F 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 permanent magnet and brushless DC motor control.

The FRDM-KV31F platform features OpenSDA, the Freescale 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.

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

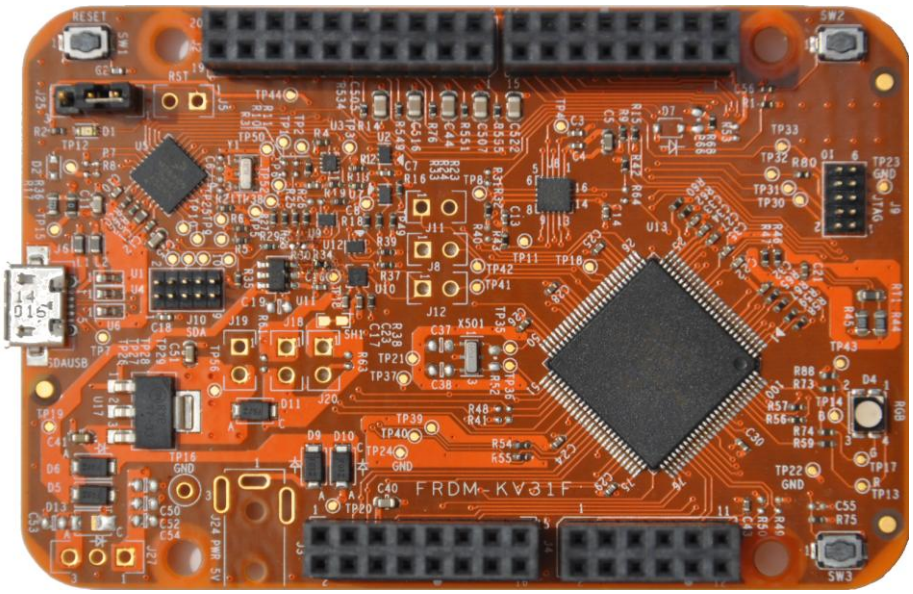

Figure 9. FRDM-KV31F Freescale Freedom Development board

3.3.4. Freedom System Assembling

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

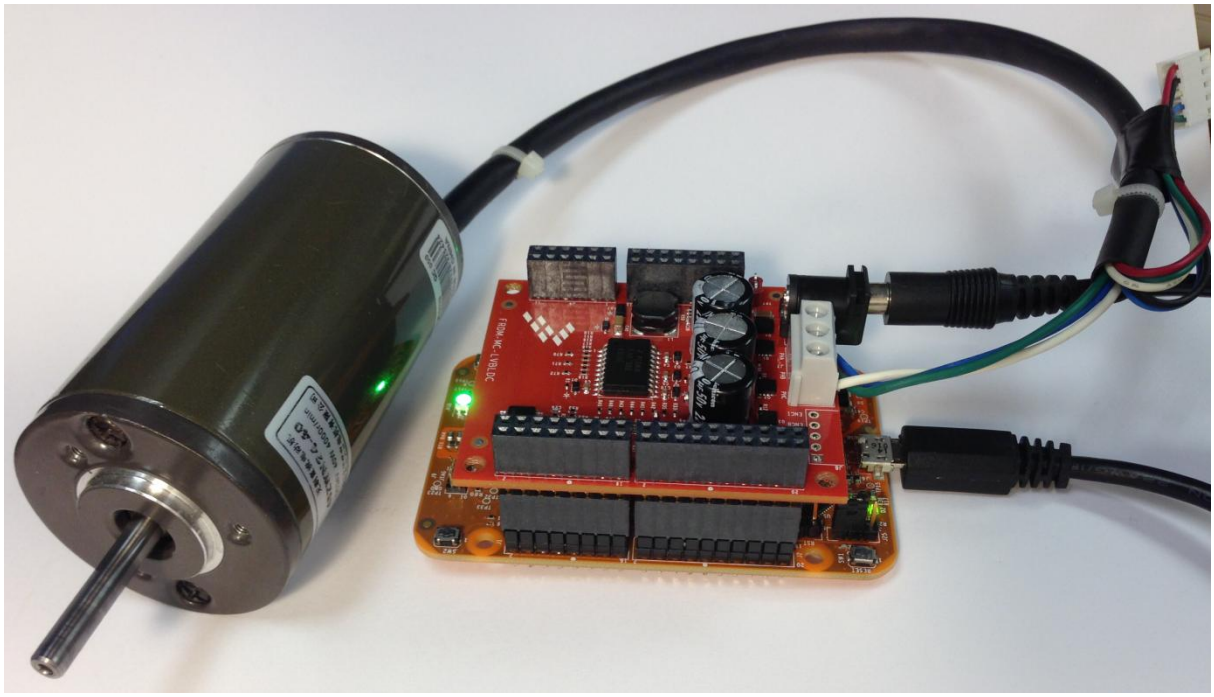

Figure 10. Assembled Freescale Freedom System

4. Project File structure

BLDC package includes source codes for development boards produced by Freescale such as Kinetis Tower and Freedom development platforms.

All necessary files are placed to one package which simplifies distribution and size of final package. Directory structure of the package is simple, easy to use and logically organized. Folder structure do not fully correspond to structures show in IDE workspace. The detail IDE description is explained in following chapters.

4.1. Structure organization in BLDC installation

A folder tree of the BLDC package installation is shown in *Figure 11*.

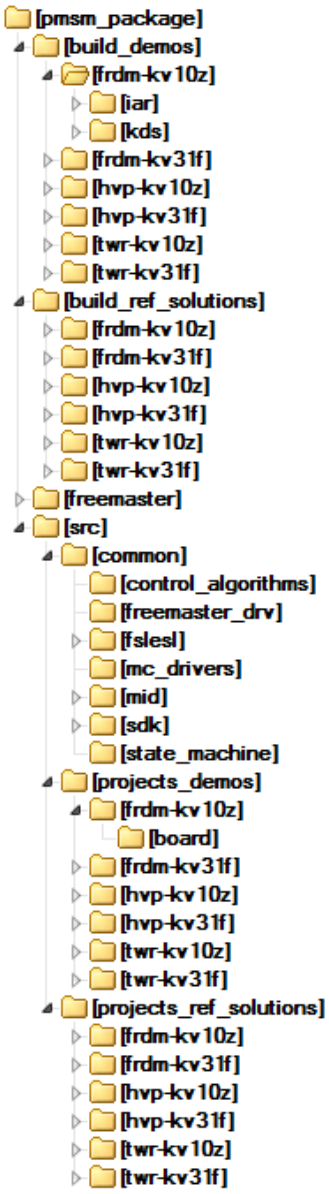

Figure 11. BLDC - Package structure TBD

BLDC package is composed of these three folders:

- build_ref_solutions
- freemaster
- src

Main demo project folder /*build_ref_solutions*\<*platform_MCU*>\ contains this folders and files

- IAR folder –
    a) contains configuration files for the IAR compiler, as well as the compiler's output executable and object files. If the IAR Embedded Workbench for ARM is installed on your computer, double-click the workspace "build_ref_solutions

_<platform_MCU>.eww" to launch the IAR IDE.

b) This folder contains Freemaster project "build_ref_solutions _<platform_MCU>.pmp" which can be opened by FreeMASTER tool and it is configured to tune a motor application compiled by IAR.

- KDS folder –

c) contains configuration files for the KDS IDE, and launch configuration for debuggers. If the KDS is installed on your computer, project must be open with KDS IDE.

d) This folder contains FreeMASTER project "build_ref_solutions _<platform_MCU>.pmp" which can be opened by Freemaster tool and it is configured to tune a motor application compiled by KDS.

Folder */freemaster/* contains auxiliary files for FreeMaster tool.

Folder */src/* contains subfolder where are source and header files for the each project:

- common – Common source and header file used in all projects.
- projects_ref_solutions – Folder with source code sorted by <platform_MCU>, this source code files are used in IDE as */build_ref_solutions\<platform_MCU>*

Folder */src/common/* contains these subfolders which are common for the entire project in this package.

- control_algorithms – Control algorithms used to control BLDC motor.
- freemaster_drv – FreeMASTER header and source code required for FreeMASTER tool.
- fslesl – Folder with mathematical functions used in project. This folder includes required header files, source files and library files which are used in project, thus, contains two subfolders "cm0" and "cm4", where are precompiled library files fitted to the two ARM core CM0/CM4. FSLESL folder is taken from official FSLESL release version "4.1" and it is fully compatible with official release, library names are changed for better use in available IDEs in bldc_package. More information about FSLESL can be found ("freescale.com/fslesl").
- mc_drivers – This folder includes source and header files used to initialize and drive motor control application.
- sdk – Contains functions for startup routines, header files, core specific functions, linker files used by available IDEs in this package and routines for core clock settings. This source and header files are obtained from official Kinetis SDK ("freescale.com/KSDK"), user may to find all necessary information about this files used in this port of SDK.
- state_machine – Contains state machine functions for Fault, Initialization, Stop and Run state.

Folder */src/projects_ref_solutions/\<platform MCU>* is linked with "build_ref_solutions" project and contains all source files used in workspace. This files are specific for each of the project available in there build folders, it specifies a peripheral initialization routines, freemaster initialization, motor definitions and state machine. Source codes contains lot of comments thus in next lines is described a point of files.

- m1_bldc_appconfig.h – Contains constants definitions for the application control processes (parameters of the motor and regulators, and the constants for BLDC sensorless control related algorithms). When the application is tailored for another motor using the Motor Control Application Tuning Tool, this file is generated by the tool at the end of the tuning process. Prefix “m1_” as “motor1” is used in one-motor application, in multi-motor application this number counting to use motor number.
- m1_state_machine.c – Contain the software routines that are executed when the application is in a particular state or state transition.
- m1_state_machine.h – Header file and macros for m1_state_machine.c.
- main.c – Contains basic application initialization (enabling interrupts), subroutines accessing the MCU peripherals, and interrupt service routines. The FreeMASTER communication is performed in the background infinite loop.
- main.h – Header file for main.c.
- motor_def.h – Fault macros, control structure.
- board/app_init.c – Application init, not necessary for motor control – UART (FreeMASTER initialization), buttons and LEDs initialization functions.
- board/app_init.h – Header file for app_init.h, LED and UART macros.
- board/freemaster_cfg.h – FreeMASTER common configuration file
- board/mcdrv_<board_MCU>.c – Motor control driver peripherals initialization functions specific for board and used MCU.
- board/mcdrv_<board_MCU>.h – Header file for board/mcdrv_<board_MCU>.c, this files contains macros for changing a FlexTimer pwm periode and ADC channels assigned to phase currents and board voltage.

4.2. Structure organization in IDE workspaces

Structure organization in workspaces (section 6-Building and debugging application) is different compare to structure organization described in these sections but using same files as mentioned before. Different organization has been chosen for better manipulation of folders and files in workplaces and for the possibility to add or remove files and directories by user.

5. Tools

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

5. Iar Embedded Workbench IDE v7.40 or higher
6. Kinetis Design Studio IDE v3.0 or higher
7. FreeMASTER Run-Time Debugging Tool 2.0

6. Building and debugging application

The package contains projects, for KDS and for IAR Embedded Workbench tools. Both of them are equivalent from motor control application point of view. IDE configurations are default and there are no special conditions to run or debug a demo application.

6.1. IAR Embedded Workbench

Users who have installed IAR Embedded Workbench can use this IDE to compile and run a reference solution project.

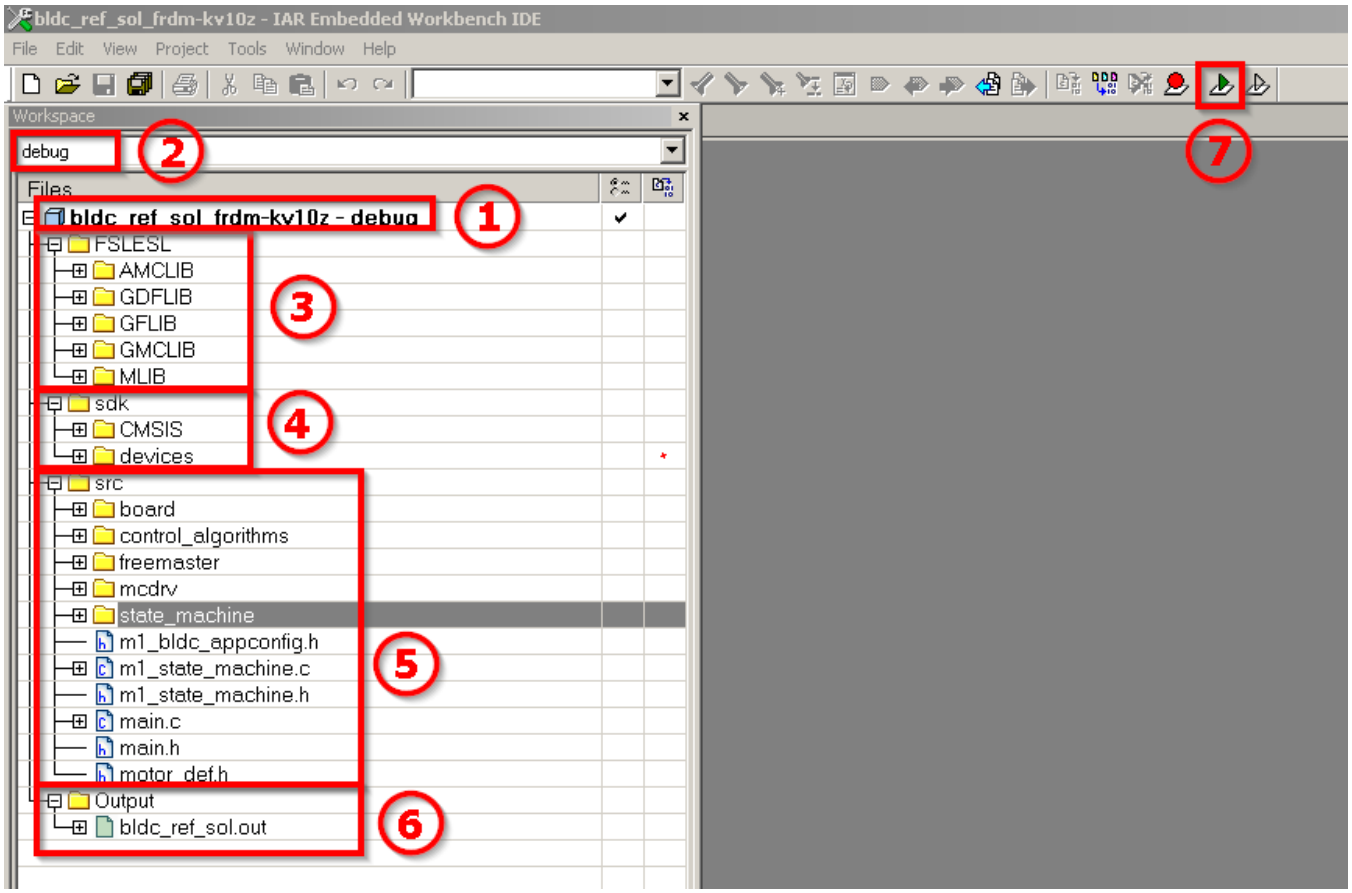

Figure 12. IAR Embedded Workbench – for bldc TBD

Project opened in IAR workbench is fully configured, includes all necessary source and header files required by the application, such as startup code, clock configuration, and peripherals configuration. User may choose two compiling conditions “debug” and “release” shown in *Figure 12* point “2”. This two conditions have own settings where:

- Debug condition – Used for debugging, optimization have flag “None – turned off”.
- Release condition – Used for release, optimization have flag “High – Highest optimization for speed”.

NOTE

Debug condition have turned off optimization, output file may be not fit in MCU with lower size of Flash memory (eg. KV10Z32).

Points “3”, “4” and “5” show source and header files used by application necessary to run project.

- Point “3” shows FSLESL source folder contains header files for mathematical and control functions used in this project, theory about use and apply these functions are described in own user guides for all of these libraries such as “AMCLIB” and can be found in “freescale.com/fslesl” link.
- Point “4” shows “sdk” folder which contains two subfolders, “CMSIS” and “devices”, both of this subfolders are device and/or MCU-core specific, defining startup conditions of MCU, clock speed and interrupt routines.
- Point “5” shows own source code of application, this folder organization is different in comparison with mentioned in chapter 4.1, however, it is composed from this files and it is organized by fit for user interaction in IDE workspace.
- Point “6” shows generated output file by compiler and ready-to use via default debugger which is “PEMicro – OpenSDA”, this debugger is as default in tower boards, can be changed in project options by right-click on Point “1”, “Options” and in menu “Debugger” is able to change debugger as is required.

6.2. Kinetis Design Studio (KDS)

Kinetis Design Studio known as KDS is IDE tool. KDS can be used for developing and testing applications with Freescale MCUs. It supports wide range of Freescale MCU from Kinetis devices such as powerful “K” family, low-power “KL” family and family aimed to motor control, family “KV”. KDS includes tools for compiling, linking and debugging source code and support a wide range of debuggers such as PEMicro or Jlink and more. The latest release of KDS can be obtained from official Freescale website at address “freescale.com/kds” where user can find installation and configuration which is mentioned in “Kinetis Design Studio V3.0.0 User Guide - KDSUG”.

To open demo or solution project, it is needed to choose development board and MCU. For example if we want to open project for Freedom development platform, MCU Kinetis KV10, we locate project in

path *bldc_package\build_ref_solutions\frdm-kv10z\kds*, then we run KDS IDE from default installation place or from installed, then we do steps described below :

We click to “File” menu in left-top corner of the IDE, then we chose “Import” from “File” list menu, *Figure 13.*

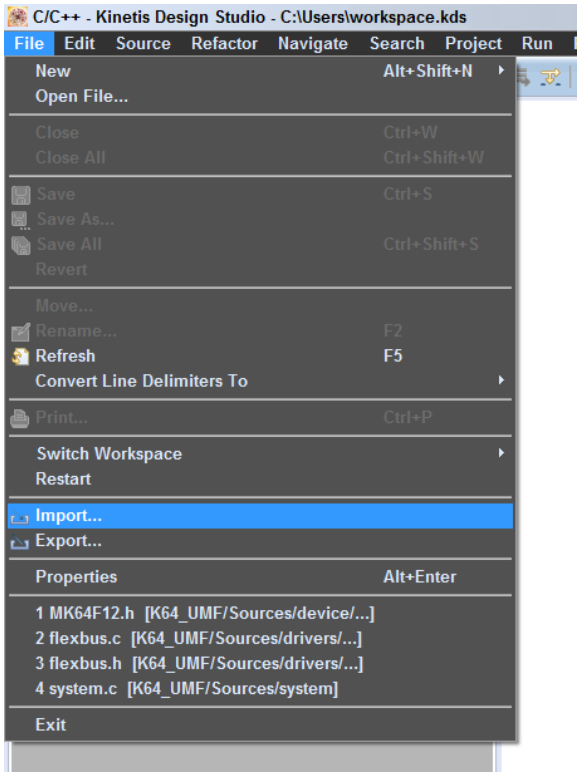

Figure 13. KDS – Import project

Then the „Import“ window is opened where we locate „Existing Projects into Workspace“ in „General“ folder and then continue pressing „Next“ button *Figure 14.*

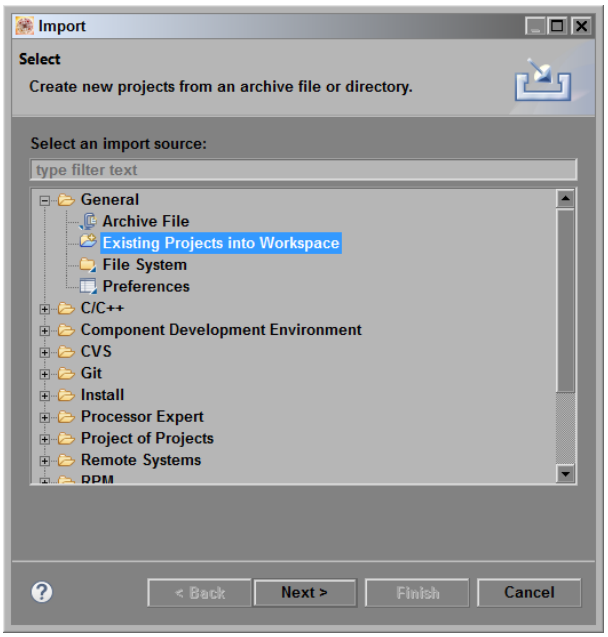

Figure 14. KDS - Import project

The “Import” window is opened as shown *Figure 15*, where we click on “Browse” button and then locate chosen project “*//bldc_package\build_ref_solutions\frdm-kv10z\kds”*, confirmed by “OK” button.

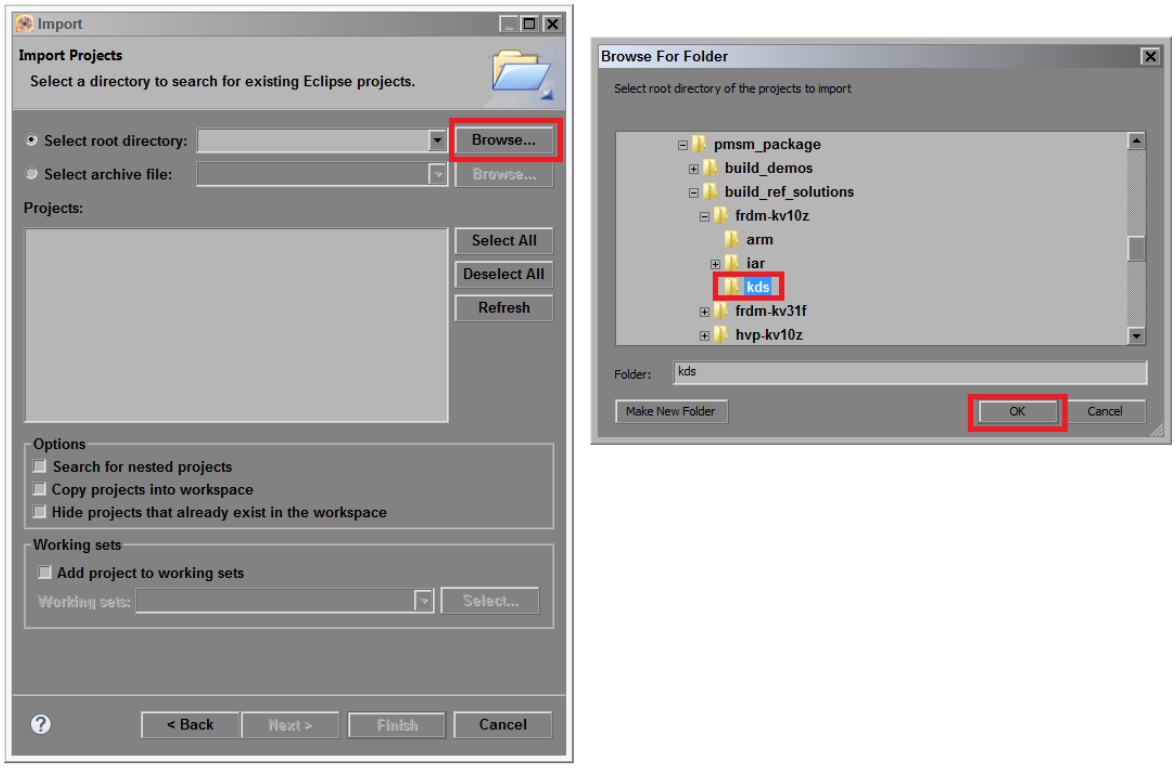

Figure 15. KDS - Import project

Last step is the confirmation of project by clicking on “Finish” button, *Figure 16.*

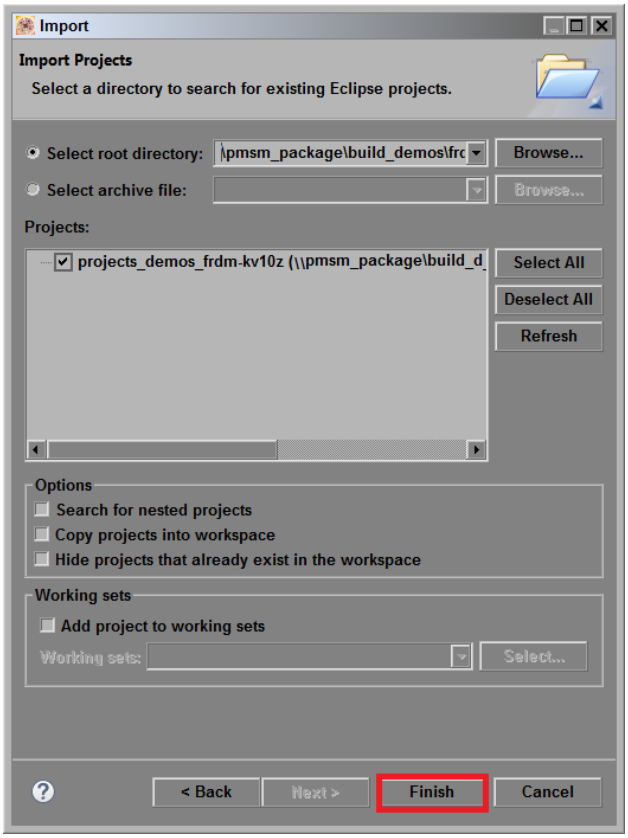

Figure 16. KDS - Import project

Now the project is imported to Kinetis Design Studio as shown in Figure 17.

- Point “1” shows imported project in “Project Explorer”
- Point “2” shows source code of this project.
- The project can be build by clicking on build icon (Point “3”) where the default configuration is set to “debug”. User can change configuration to “release”.

This two conditions have own setting where:

- Debug condition – Used for debugging, optimization have flag “None – turned off”.
- Release condition – Used for release, optimization have flag “High – Highest optimization for speed”.

NOTE

Debug condition have turned off optimization, output file may be not fit in MCU with lower size of Flash memory (eg. KV10Z32).

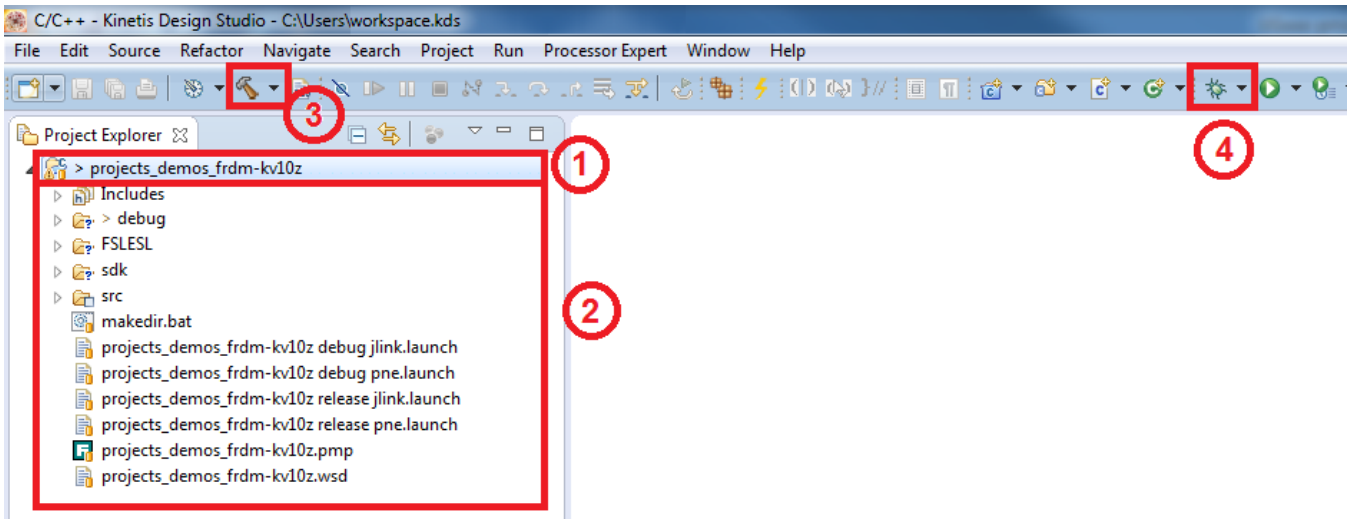

Figure 17. KDS BLDC project - TBD

Once project is compiled, the “debug” or “release” directory is created depending on chosen build configuration. The *.elf binary file is created in one or both of those directories. Now it is possible to use debugger (Point “4”, Figure 17) with predefined debugger option as PEMicro – OpenSDA as default debugger. User can use another from the menu or to define own in top list menu, “Run-> Debug Configuration” where user can define another type of debugger or change condition of debugging such as optimization level.

6.3. OpenSDA debugger

Freescale development boards (mentioned in sections before) includes OpenSDA serial and debugger adapter which serves as bridge serial and debug communication between target processor and Host computer. This debugger provides virtual serial port similar between host computer, where can be accessed as “COM” port and on the embedded target is connected to UART peripheral, also in same time can control debug interface that control JTAG or SWD debug interface in target developing platform. OpenSDA debug interface provides Mass Storage Device Flash Programmer (MSD Flash Programmer). MSD Flash Programmer provides easy way to program applications into the flash of the target processor. It appears as a removable drive in the host operating system with a volume label that matches the board name. Raw binary Motorola S-Record files that are copied to the drive are programmed directly into the target memory. The MSD Flash Programmer is designed to program a specific target configuration. It does not support verification or configuration and is not recommended as a production programmer.

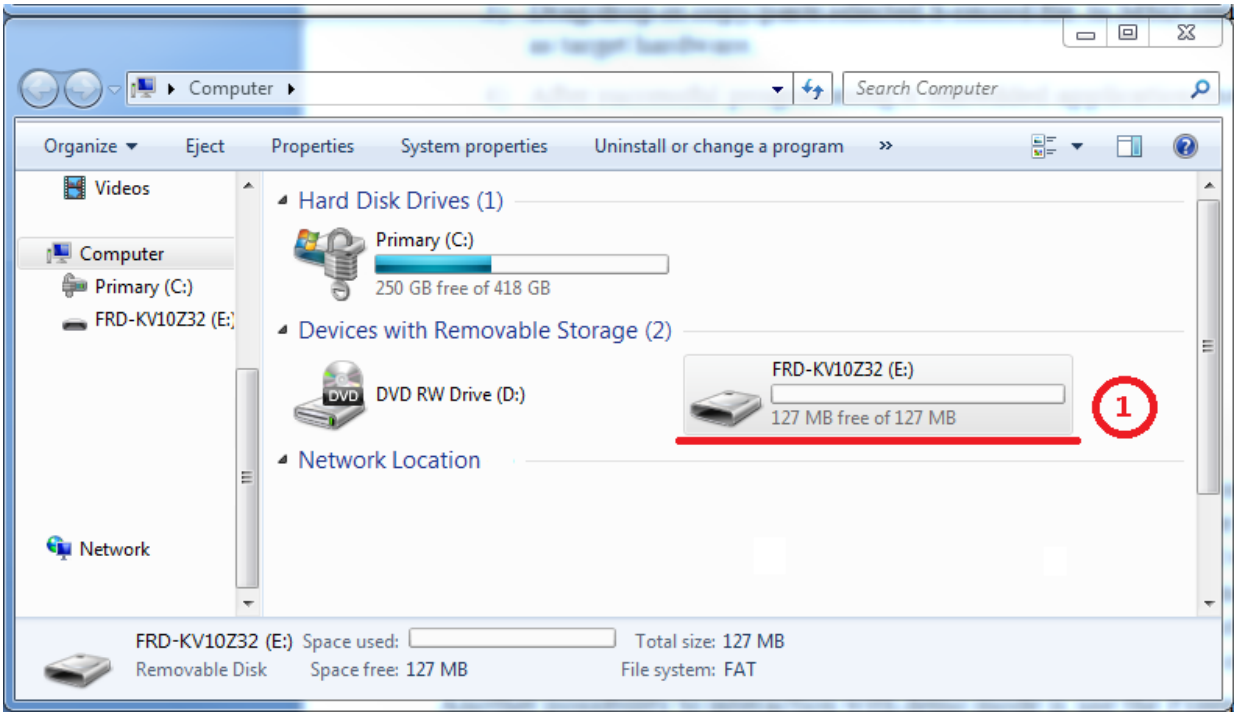

Figure 18. OpenSDA MSD

To load a generated application directly to target MCU follow the next steps. This steps are applicable in Windows host System.

1) Open Windows Explorer
2) Locate generated S-record file (file with extension .srec, .s19 or .bin) (for example *pmsm_package\build_ref_resolutions\<board_MCU>\<tool>\debug\projects_ref_solutions.srec* as shown.
3) Drag/drop or copy/paste selected S-record file to MSD removable drive with a label of volume as target hardware (**Error! Reference source not found.** "Point 1").
4) After successful programming the embedded application executes automatically.
5) Reconnect the target device.

6.4. Compiler warnings

Warnings are diagnostic messages that report constructions that are not inherently erroneous and warn about potential runtime, logic and performance errors. In some cases warnings may be suspended and these warning are not shown during compiling process. One of the on special cases is warning about "Unused function" where function and their body are implemented in source code, but this function is not used. This case may occur in situation where we implementing function for better usability, as supporting function, but in some special purposes we don't using function yet.

BLDC project contains “reference solutions” projects. In these projects, especially in IDEs are some warning suppressed, these warnings are listed below, and other warning are setup to standard configuration.

IAR workbench has suppressed following warnings:

- Pa082 - Undefined behavior: the order of volatile accesses is undefined in this statement
- Pe177 - <entity> was declared but never referenced

Kinetis Design Studio (KDS) has suppressed following warnings:

- parentheses - Warn if parentheses are omitted in certain contexts
- unused-function – function is defined but no used
- uninitialized – variable is used but uninitialized
- main – due to optimization, main is void not integer, never return value

During compiling process there are no other warnings shown by default.

7. User Interface

The application contains demo application mode for demonstration purposes of motor rotating and can be operated either using the user button or with using a FreeMASTER Tool. Tower and Freedom boards have included user button key associated with port interrupt, generated whenever button is pressed. Pressing the button causes the start of demo mode; next pressing of the same button causes the stop state and transition back to the STOP state. Location and number of User button is defined in Table 8.

Another possibility to interaction with demo mode is use the FreeMASTER tool. The FreeMASTER application consists of two parts: the PC application used for variable visualization, and the set of software drivers running in the embedded application. The data is transferred between the PC and the embedded application using the RS232 interface, this interface is provided by OpenSDA debugger included in boards.

The application can be controlled using two interfaces:

- Button on Freescale Kinetis V Tower and Freedom development boards
- Remote control using FreeMASTER

7.1. Button Control

Pressing the corresponding button the demonstration mode is switched on (or demonstration mode is switched off if it is on). Check the following table to recognize the correct switch button on your development board.

Table 8. **Control button assignment**

Board's name	Control Button
TWR-KV10Z32	SW1
TWR-KV11Z75	SW2
TWR-KV31F120	SW3
FRDM-KV10Z	SW2
FRDM-KV31F	SW2

7.2. Remote control using FreeMASTER

Remote operation can be provided by FreeMASTER software via the USB interface. FreeMASTER 2.0 is required to guarantee the correct operation of application. FreeMASTER 2.0 application installation is available to download on www.freescale.com/freemaster.

Follow the following steps to be able to control BLDC motor using FreeMASTER:

1. Open the FreeMASTER project file (.pmp). According used IDE open corresponding FreeMASTER project.
   - In case of KDS, the FreeMASTER project file is located in following path:

   Project_directory/build/kds/
   - In case of Iar, the FreeMASTER project is located in following path:

   Project_directory/build/iar/
2. The next step is to toggle the communication button (red STOP button in top left hand corner) to establish the communication.

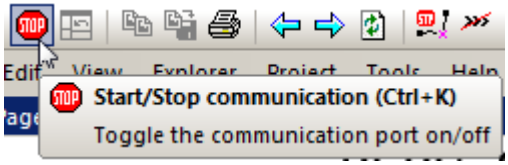

Figure 19. Red STOP button is placed in top left hand corner

If communication has been established successfully, the FreeMASTER communication status in the bottom right hand corner should change from Not connected to RS232 UART Communication; COMxx; speed=19200, otherwise FreeMASTER warning popup window should appear.

Figure 20. FreeMASTER communication status when communication is established successfully

3. Enjoy controlling the BLDC motor with control page.

If the communication has not been established successfully, try the following steps:

1. Go to Project->Options->Comm tab and make sure, that in port option is written SDA and

communication speed is set to 19200 bps.

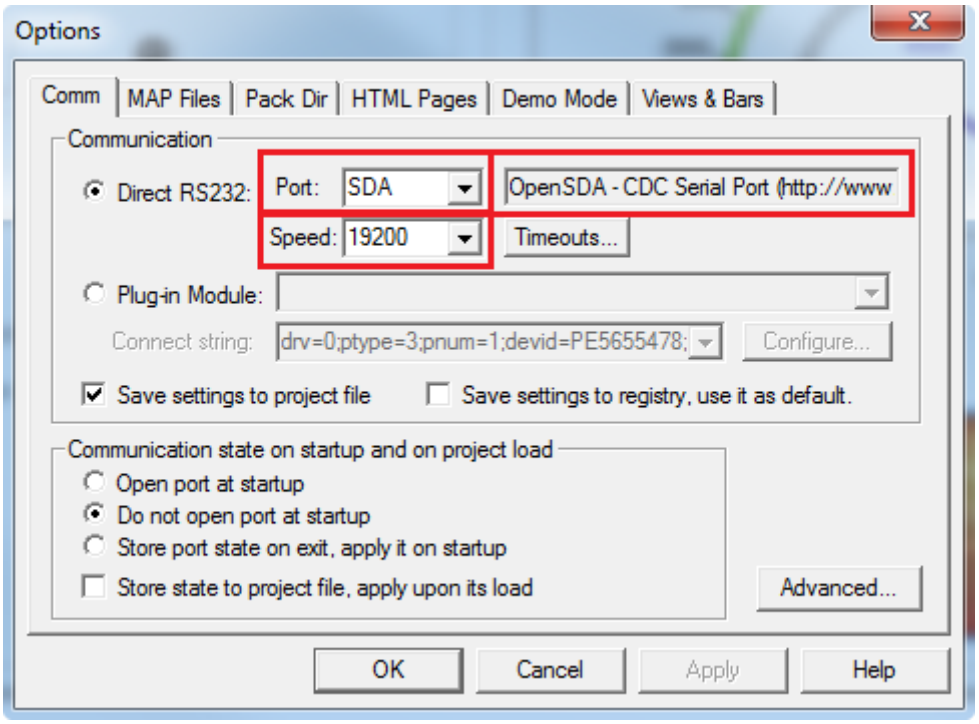

Figure 21. FreeMASTER communication setup window

2. If OpenSDA-CDC Serial Port is not printed out in message window near to port listbox, try to unplug and plug in usb cable and reopen the FreeMASTER project again.
3. Make sure that your development board is supplied from sufficient energy source. Sometimes the USB PC port is not sufficient for supplying the development board.

7.2.1. Control Page

After launching the application and performing all necessary settings BLDC motor can be controlled from FreeMASTER control page. FreeMASTER control page contains:

- Speed gauge – showing actual and required speed
- Speed slider – to setup the required speed
- DC bus voltage gauge – showing actual dc bus voltage
- DC bus current gauge – showing actual dc bus current
- DC bus current limitation slider – to setup dc bus current limit
- Demo mode button – To turn on/off demonstration mode
- STOP Button – to stop the whole application
- Application State Notification – showing actual application state, faults also

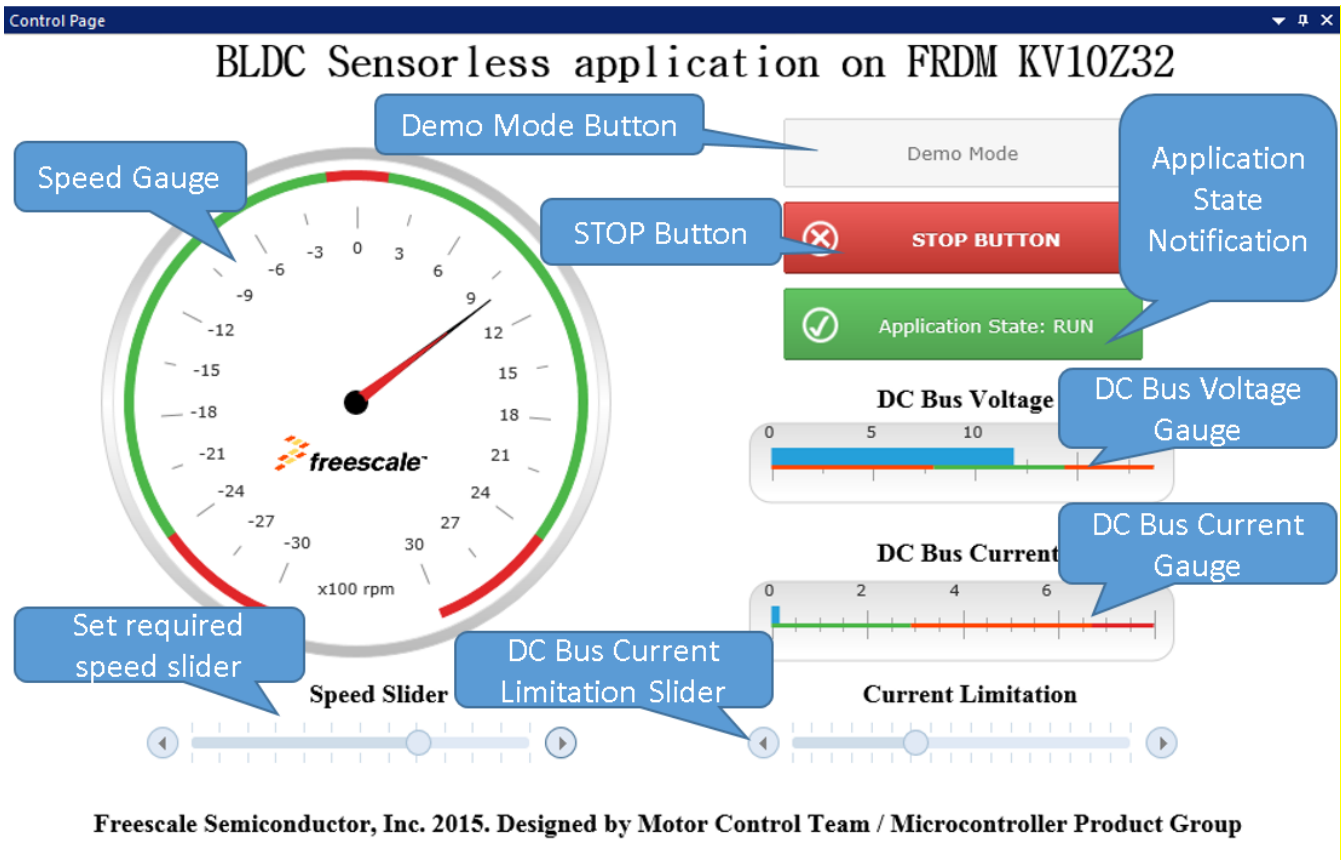

Figure 22. FreeMASTER control page

Basic Instructions:

- To start the motor run, use the Speed Slider to setup required speed.
- In case of a fault is indicated, click on “CLEAR FAULT” button to clear fault.
- Click on Demo Mode button to turn on/off demonstration mode
- Click on Stop button to stop the motor.

8. How to

8.1. How to Run motor

1. Assembly your Freescale hardware according to instructions in chapter 2.
   a) In case of Tower development board follow this instuctions.
   b) In case of Freedom development board follow this instructions.
2. Download the correct project into your target via OpenSDA debug interface.
3. Open the FreeMASTER project, establish the communication between MCU and PC according instructions in this chapter

4. Set up the required speed of the motor on the speed slider on the FreeMASTER control page.

8.2. How to Stop motor

a) Click on Stop button on FreeMASTER control page.
b) Require zero speed with using speed slider.
c) In emergency cases, turn off power supply.

8.3. How to Clear the fault

To clear the fault just click on “CLEAR FAULT” button control page when occurs.

8.4. How to Turn on Demonstration mode

d) If you use FreeMASTER control page, just click on Demo Button.
e) If you don’t use FreeMASTER control page push the corresponding button on your development board to turn demonstration mode on/off.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN5205
Rev. 0
10/2015