Freescale Semiconductor, Inc.
User's Guide

Document Number: PMSMSAPUG
Rev. 0, 10/2015

PMSM Sensorless Application Package User's Guide

By: Marek Zeman

1. Introduction

This user’s guide provides a step-by-step guide on how to open, compile, and run PMSM projects. It describes the basic compiling steps for IAR Embedded Workbench® and Kinetis Design Studio (KDS) for a wide range of supported development platforms, mentioned in Section 2, “Supported development boards.” It also describes the initialization of the FreeMASTER GUI tool for controlling motor-control applications.

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 supported MCUs are shown in Table 1. The Tower System modular development platform and Freescale Freedom development platform are targeted for low-voltage and low-power applications with PMSM control type. The High-Voltage Platform (HVP) is designed to drive high-voltage (115/220 V) applications with up to 1 kW of power.

Table 1. Supported development platforms

MCU/board	Tower	Freedom	HVP
—	TWR-LV3PH	FRDM-MC-LVPMSM	HVP-MC3PH
KV10Z	TWR-KV10Z	FRDM-KV10Z	HVP-KV10Z
KV31F	TWR-KV31F	FRDM-KV31F	HVP-KV31F

3. Hardware setup

The PMSM sensorless application runs on Tower and Freedom development platforms with 24 V Linix Motor and High-Voltage Platforms with 220 V PMSM motor in the default configuration.

3.1. Linix 45ZWN24-40 motor

The Linix 45ZWN24-40 motor (described in Table 2) is a low-voltage three-phase motor used in BLDC and PMSM sensorless applications.

Table 2. Linix 45ZWN24-40 motor parameters

Characteristic	Symbol	Value	Units
Rated voltage	Vt	24	V
Rated speed @ Vt	—	4000	RPM
Rated torque	T	0.0924	Nm
Rated power	P	40	W
Continuous current	Ics	2.34	A
Number of pole pairs	pp	2	—

Figure 1. Linix motor

The motor has two types of connectors (cables). The first cable has three wires and it is designated to power the motor. The second cable has five wires and it is designated for Hall sensors signal sensing. For the PMSM sensorless application, you need only the power input wires.

3.2. MIGE 60CST-MO1330 motor

The MIGE 60CST-MO1330 motor (described in Table 3) is used by the PMSM sensorless application. You can also adapt the application to other motors, just by defining and changing the motor-related parameters. The motor is connected directly to the high-voltage development board via a flexible cable connected to the three-wire development board connector.

Table 3. MIGE 60CST-MO1330 motor parameters

Characteristic	Symbol	Value	Units
Rated voltage	V_t	220	V
Rated speed @ V_t	—	3000	RPM
Rated power	P	400	W
Number of pole pairs	Pp	4	—

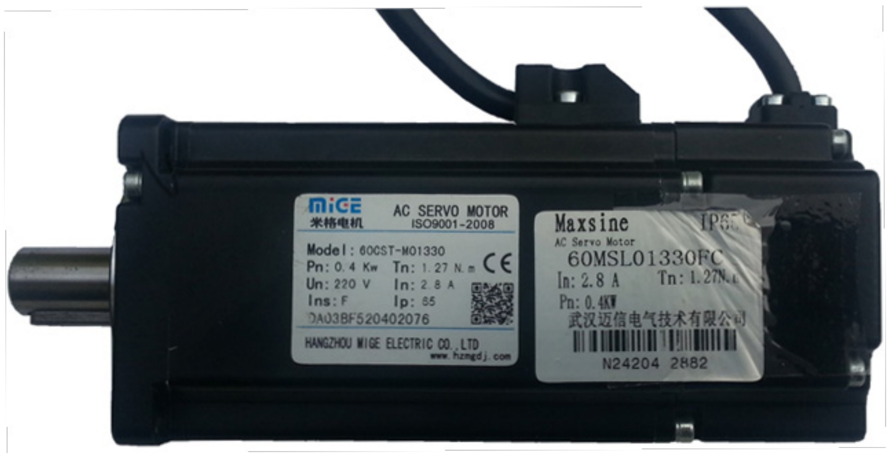


Figure 2. MIGE motor

3.3. Running PMSM application on Tower System

To run the PMSM application on the Tower System, you need the following Tower modules:

- Kinetis KV10Z Tower module (TWR-KV10Z32) 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).

You can order all Tower modules from www.freescale.com or from distributors to easily build the hardware platform for the target application.

3.3.1. TWR-MC-LV3PH peripheral module

The Three-Phase Low-Voltage Motor Control Board (TWR-MC-LV3PH) is a peripheral Tower module, interchangeable across the Tower System. Phase voltage and current feedback signals are provided. These signals enable a variety of algorithms for controlling three-phase PMSM and BLDC motors. The MC33937 predriver enables a high level of board protection (overcurrent, undervoltage, overtemperature, and so on). Before inserting the TWR-MC-LV3PH peripheral card into the Tower System, make sure that the jumpers on the 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	1-2	Selects I_SENSE_C
J11	1-2	Selects I _SENSE_B
J12	1-2	Selects I _SENSE_A
J13	2-3	Selects I_SENSE_DCB
J14	2-3	Selects V_SENSE_DCB_HALF

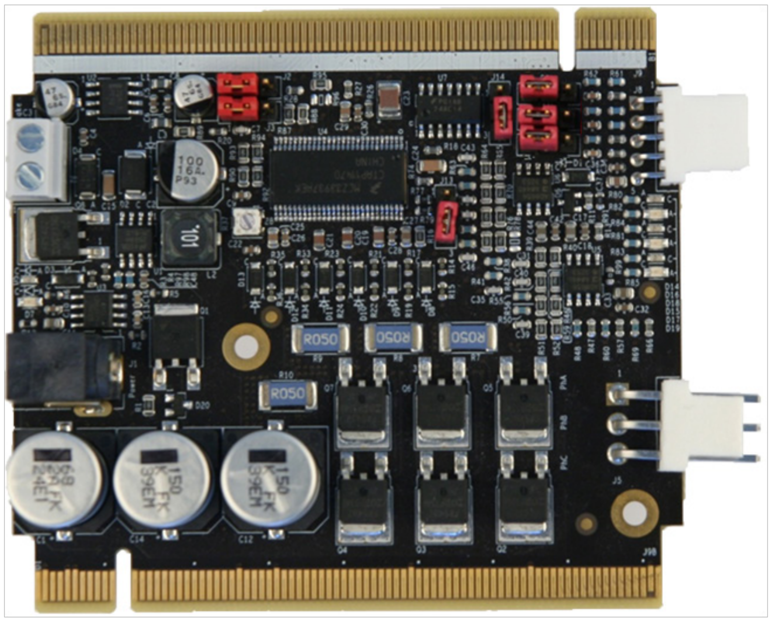


Figure 3. TWR-LV-MC3PH jumper setting for PMSM

3.3.2. TWR-KV10Z32 MCU module

The TWR-KV10Z32 MCU module is a part of Freescale Tower System, a modular development platform that enables rapid prototyping and tool reuse through reconfigurable hardware.

Table 5. TWR-KV10Z32 MCU module jumper 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 the on-board push-button SW2.

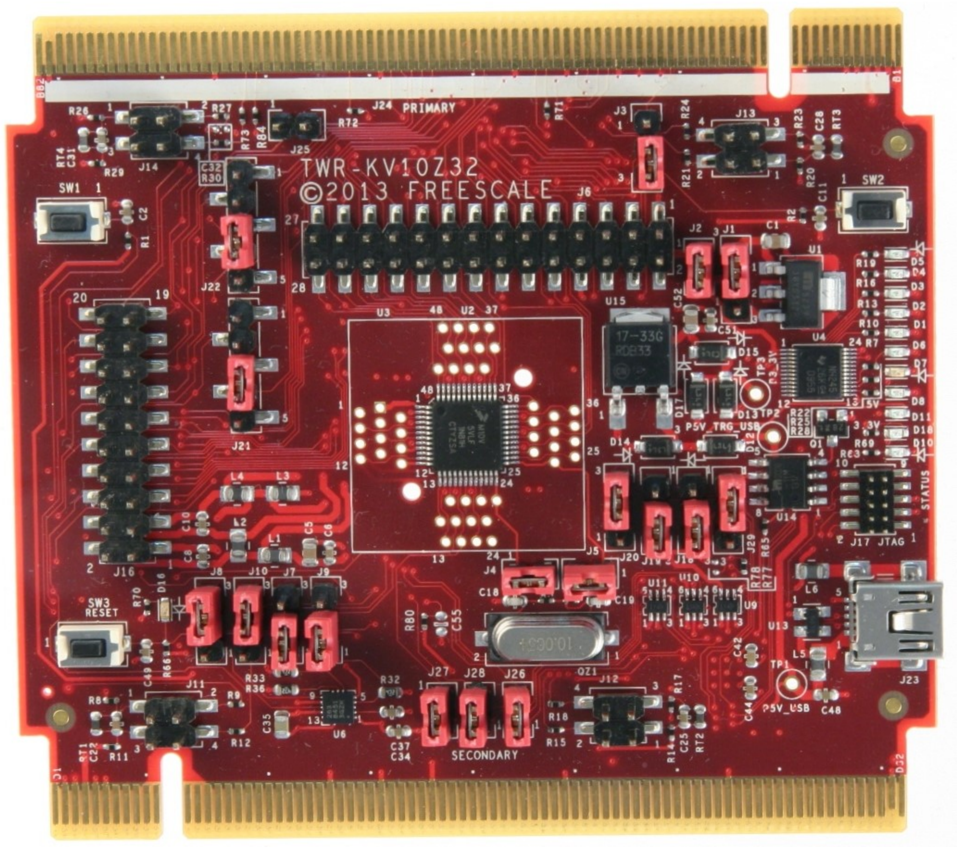


Figure 4. TWR-KV10Z32 MCU module

3.3.3. TWR-KV31F MCU module

The TWR-KV31F120M is a development tool for the Freescale Kinetis KV3x family of MCUs built around the ARM® Cortex®-M4 core. This MCU has enough power for use in motor-control applications (such as PMSM or BLDC motors with simple or advanced control techniques). The MCU has a wide range of peripherals, lots of memory (depending on the model used), and a powerful core.

To begin, configure the jumpers on the TWR-KV31F120 MCU and TWR-MC-LV3PH modules properly. The following table lists the specific jumpers and their settings for the TWR-KV31F120 MCU module.

Table 6. TWR-KV10Z32 MCU module jumper 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.

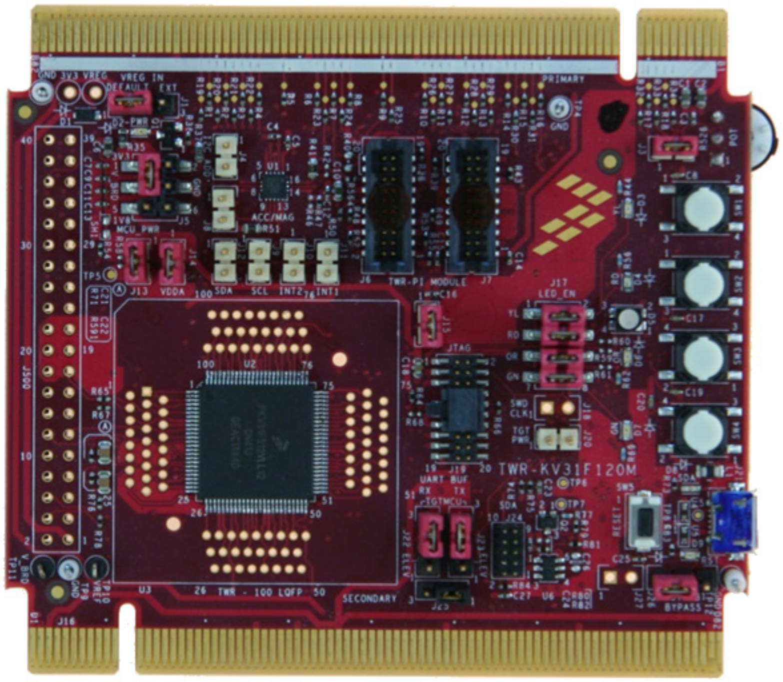


Figure 5. TWR-KV30F120M MCU module

3.3.4. Tower System assembling

1. Insert the TWR-KVxxXxx MCU module and the TWR-MC-LV3PH peripheral module into the TWR-ELEV cards. 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 assembling the Tower System, connect the required cables as follows:
   - Connect the power input cable (three-wire connector) of the Linix motor to its corresponding connector (J5) on the TWR-MC-LV3PH motor-control driver board.
   - Plug the power supply cable 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 either to J23 of the TWR-KV10Z32 MCU module, or J21 of the TWR-KV31F120 MCU module, and any USB port on the host PC.

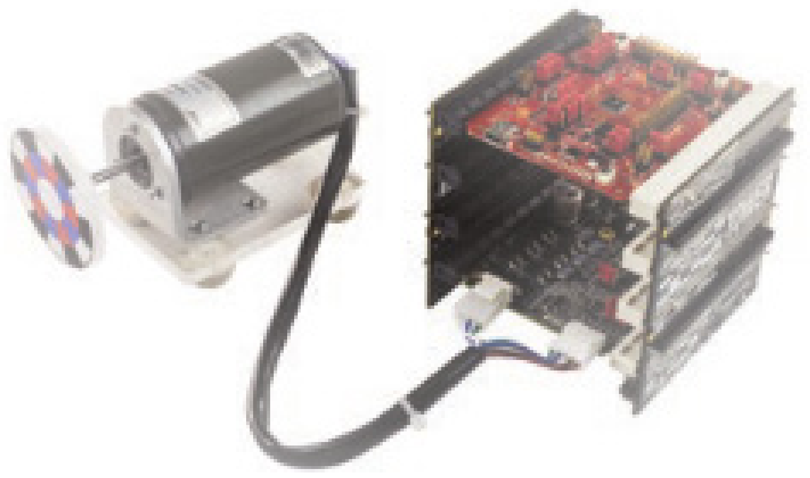


Figure 6. Assembled Tower System

3.4. Freescale Freedom platform

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

- 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.

You can order all Freedom modules from www.freescale.com or from distributors, and easily build the hardware platform for the target application.

3.4.1. FRDM-MC-LVPMSM

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

The FRDM-MC-LVPMSM board does not require any hardware configuration or jumper setting.

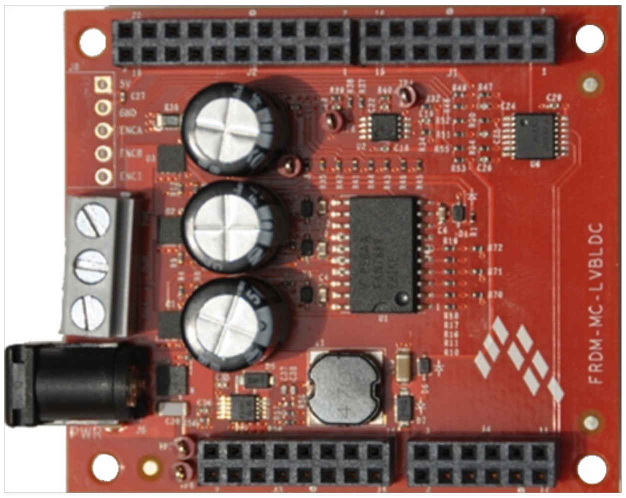


Figure 7. FRDM-MC-LVBLDC

3.4.2. FRDM-KV10Z

The FRDM-KV10Z is a low-cost development tool for Kinetis KV1x family of MCUs built around the ARM Cortex-M0+ core. 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.

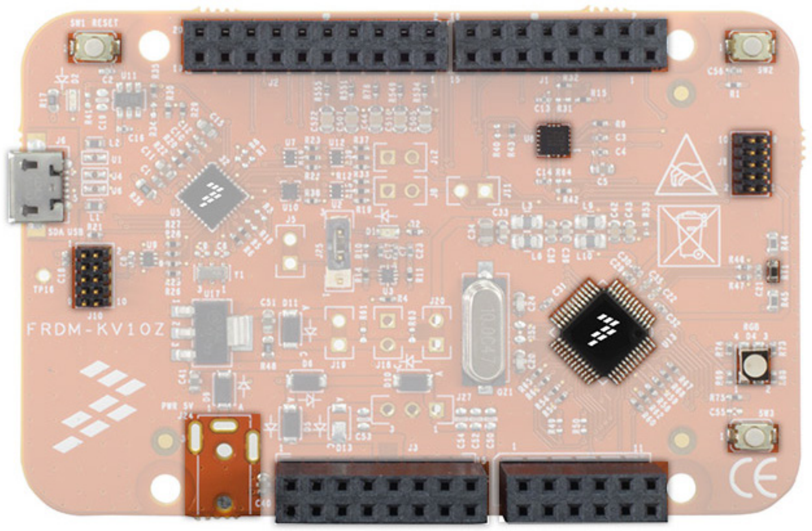


Figure 8. FRDM-KV10Z Freedom development board

3.4.3. FRDM-KV31F

FRDM-KV31F is a low-cost development tool for Kinetis KV3x family of MCUs built around the ARM Cortex-M4 core. 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.

FRDM-KV31F 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.

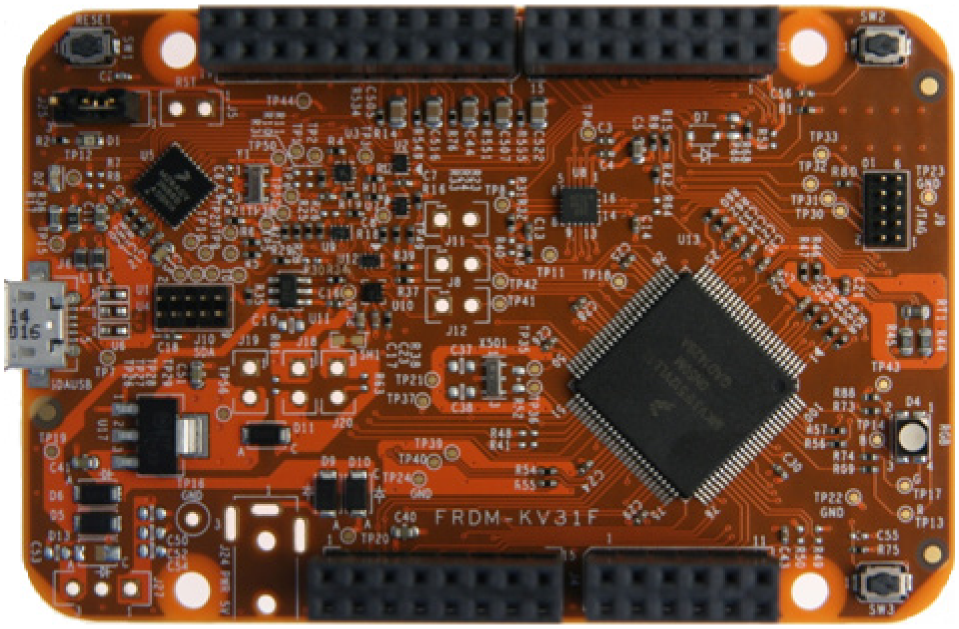


Figure 9. FRDM-KV31F development board

3.4.4. Freedom system assembling

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

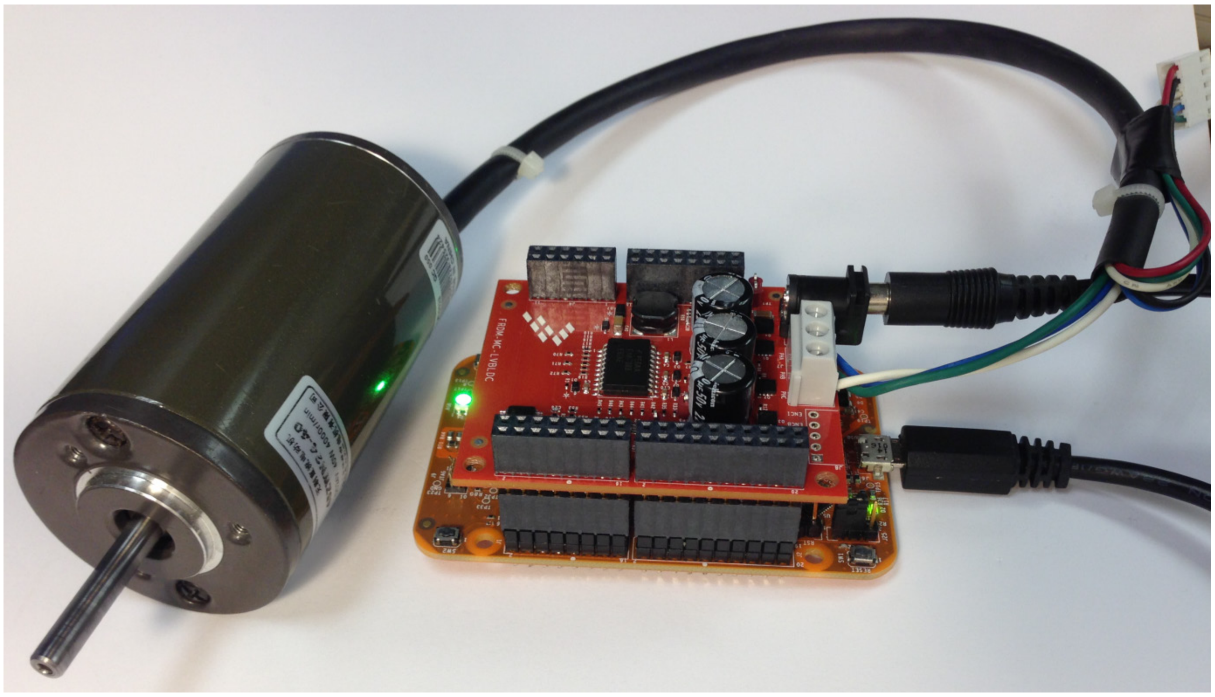


Figure 10. Assembled Freedom system

3.5. High-Voltage Platform

To run the PMSM application within the Freescale High-Voltage Platform, you need these components:

- Kinetis KV10Z high-voltage daughter board (HVP-KV10Z32) or Kinetis KV31F high-voltage daughter board (HVP-KV31F512).
- High-Voltage Platform (HVP-MC-3PH) (motor not included).

You can order all modules of the High-Voltage Platform from www.freescale.com or from distributors, and easily build the hardware platform for the target application.

3.5.1. HVP-MC3PH main board

The Freescale High-Voltage Platform is an evaluation and development solution for Kinetis V series MCUs. This platform enables development of three-phase PMSM, BLDC, and ACIM motor-control and power factor correction solutions in a safe high-voltage environment. The boards work with the default configuration, and you don't have to set any jumpers to run the attached application. See *High-Voltage Motor Control Platform User's Guide* (document HVPMC3PHUG).

You don't have to set up the HVP-MC3PH high-voltage development board in any way. The board does not contain any jumpers.

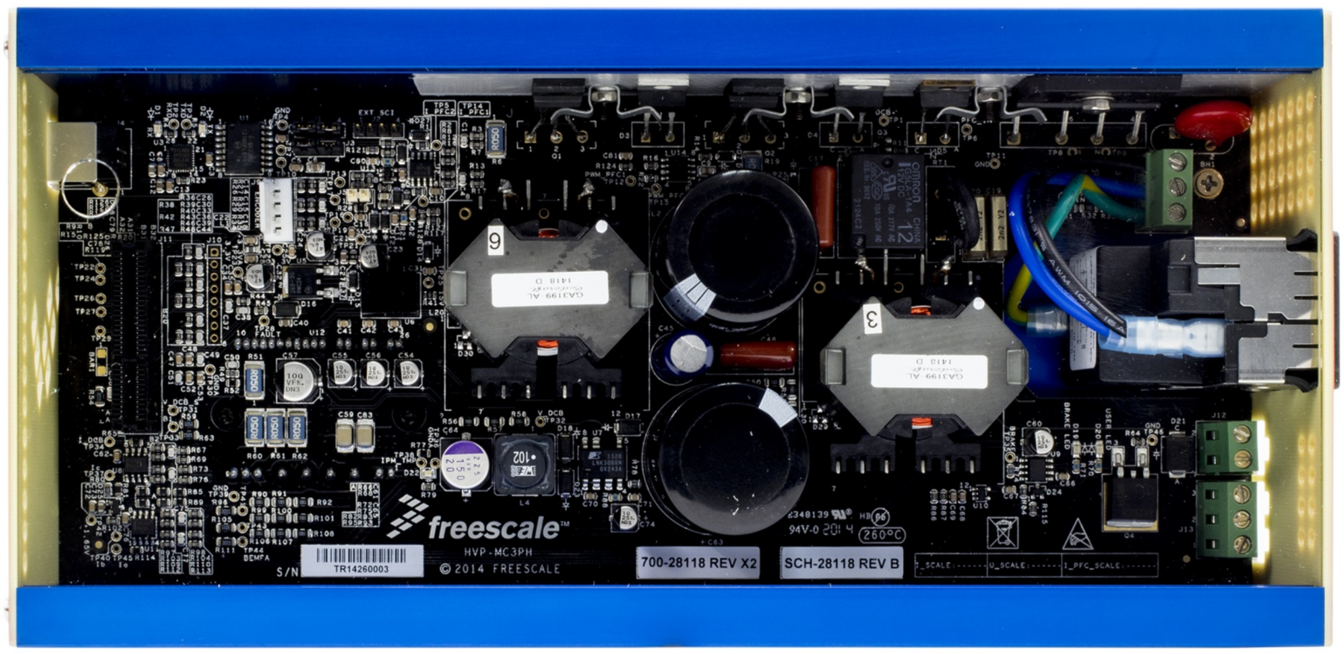


Figure 11. HVP-MC3PH High-Voltage Platform

3.5.2. HVP-KV10Z daughter board

The HVP-KV10Z MCU daughter board contains Kinetis KV10 family MCU built around the ARM Cortex-M0+ core. This daughter board is developed for use in motor-control applications, together with the High-Voltage Platform main board. This daughter board features OpenSDA, the Freescale open-source hardware embedded serial and debug adapter running an open-source bootloader.

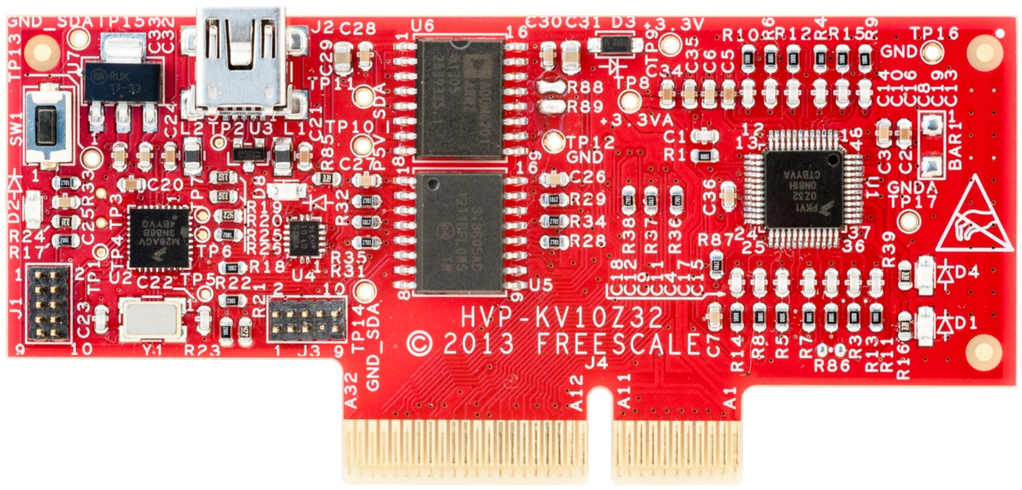


Figure 12. HVP-KV10Z32 daughter board

3.5.3. HVP-KV31F512

The HVP-KV31F512 MCU daughter board contains Kinetis KV3x family MCU built around the ARM Cortex-CM4 core (which is a powerful chip with a lot of flash memory), and the board has the capability to drive a motor. This daughter board is developed for use in motor-control applications, together with the High-Voltage Platform main board. This daughter board features OpenSDA, the Freescale open-source hardware embedded serial and debug adapter running an open-source bootloader.

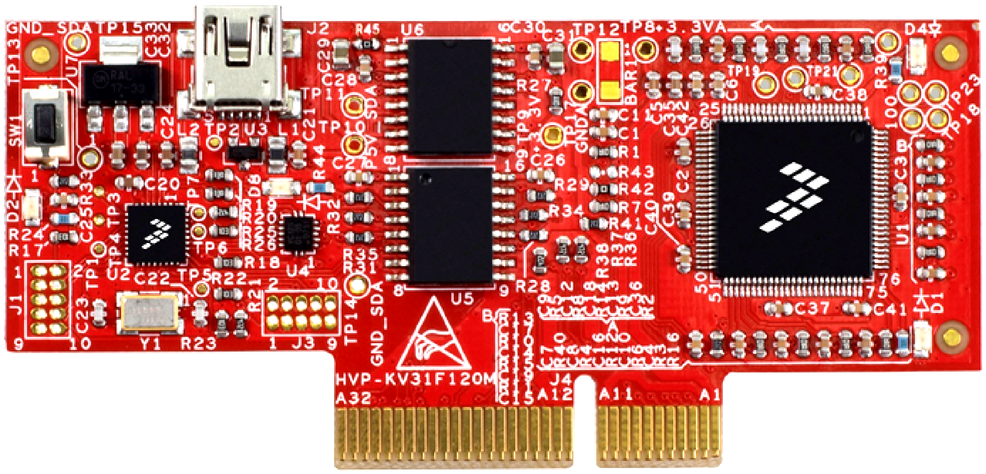


Figure 13. HVP-KV31F512 daughter board

3.5.4. High-Voltage Platform assembling

1. Check whether the HVP-MC3PH main board is unplugged from the voltage source.
2. Insert the HVP-KVxxx daughter board to the HVP-MC3PH main board (there is only one possible option).
3. Connect the PMSM motor three-phase wires into the screw terminals on the board.
4. Plug the USB cable from the USB host to the OpenSDA micro USB connector.
5. Plug a 230 V power supply to the power connector, and switch it on.

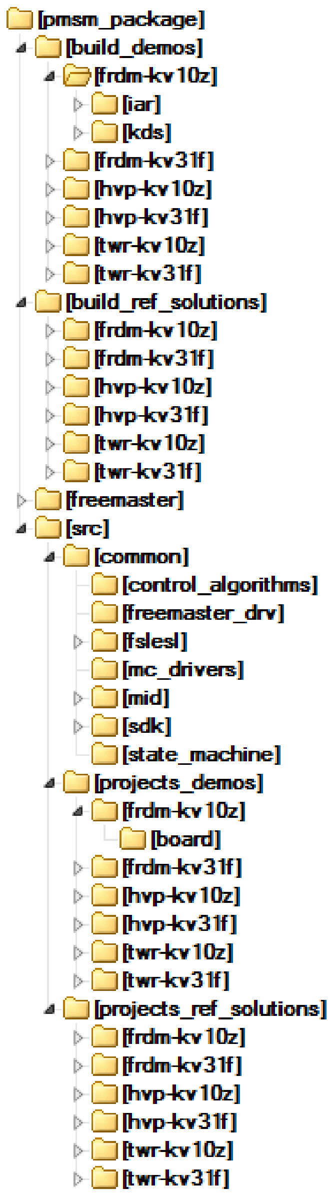
4. Project file structure

The PMSM package includes source code for Freescale development boards, such as Kinetis Tower System, Freedom development platform, and High-Voltage Platform.

All necessary files are included in one package, which simplifies the distribution and decreases the size of the final package. The directory structure of this package is simple, easy to use, and it is organized in a logical manner. Folder structure used in the IDE is different from the structure of the PMSM package installation, and it is described in separate sections.

4.1. PMSM package structure

The folder tree of the PMSM package installation is shown in this figure:



```
[pmsm_package]
  [build_demos]
    [frdm-kv10z]
      [iar]
      [kds]
    [frdm-kv31f]
    [hvp-kv10z]
    [hvp-kv31f]
    [twr-kv10z]
    [twr-kv31f]
  [build_ref_solutions]
    [frdm-kv10z]
    [frdm-kv31f]
    [hvp-kv10z]
    [hvp-kv31f]
    [twr-kv10z]
    [twr-kv31f]
  [freemaster]
  [src]
    [common]
      [control_algorithms]
      [freemaster_drv]
      [fslesl]
      [mc_drivers]
      [mid]
      [sdk]
      [state_machine]
    [projects_demos]
      [frdm-kv10z]
        [board]
      [frdm-kv31f]
      [hvp-kv10z]
      [hvp-kv31f]
      [twr-kv10z]
      [twr-kv31f]
    [projects_ref_solutions]
      [frdm-kv10z]
      [frdm-kv31f]
      [hvp-kv10z]
      [hvp-kv31f]
      [twr-kv10z]
      [twr-kv31f]
```

Figure 14. Package structure

The PMSM package is composed of these four folders:

- *build_demos*
- *build_ref_solutions*
- *freemaster*
- *src*

The main demo project folder /*build_demos*\<*platform_MCU*>\ contains these folders and files:

- *IAR* folder—contains configuration files for IAR Embedded Workbench, as well as the compiler output executable and object files. If IAR Embedded Workbench for ARM is installed on your computer, double-click the workspace "*projects_demos_<platform_MCU>.eww*" to launch the IAR IDE. This folder contains the FreeMASTER project "*projects_demos__<platform_MCU>.pmp*" which you can open in the FreeMASTER tool. The project is configured for tuning of motor-control applications.
- *KDS* folder—contains configuration files for KDS IDE and launch configuration for debuggers. If KDS is installed on your computer, open the project using KDS IDE. This folder contains the FreeMASTER project "*projects_demos__<platform_MCU>.pmp*" which you can open in the FreeMASTER tool. The project is configured for tuning of motor-control applications.

The main demo project folder /*build_ref_solutions*\<*platform_MCU*>\ contains these folders and files:

- *IAR* folder—contains configuration files for IAR Embedded Workbench, as well as the compiler output executable and object files. If 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. This folder contains the FreeMASTER project "*build_ref_solutions_<platform_MCU>.pmp*" which you can open in the FreeMASTER tool. The project is configured for tuning of motor-control applications compiled in IAR.
- *KDS* folder—contains configuration files for KDS IDE and launch configuration for debuggers. If KDS is installed on your computer, open the project using KDS IDE. This folder contains the FreeMASTER project "*build_ref_solutions _<platform_MCU>.pmp*" which you can open in the FreeMASTER tool. The project is configured for tuning of motor-control applications compiled in KDS.

The */freemaster/* folder contains auxiliary files for the FreeMASTER tool.

The */src/* folder contains these subfolders with the source and header files for each project:

- *common*—contains common source and header files used in all projects.
- *projects_demos*—contains source code sorted by <platform_MCU>. These source code files are used as /*build_demos*\<*platform_MCU*>\ in IDE.
- *projects_ref_solutions*—contains source code sorted by <platform_MCU>. These source code files are used as /*build_ref_solutions*\<*platform_MCU*>\ in IDE.

The */src/common/* folder contains the subfolders common to the entire project in this package:

- *control_algorithms*—contains the main control algorithms used to control the FOC and speed control loop.
- *freemaster_drv*—contains FreeMASTER header and source files.
- *fslesl*—contains mathematical functions used in the project. This folder includes the required header files, source files, and library files used in the project. It contains two subfolders ("*cm0*" and "*cm4*") that contain precompiled library files for two types of ARM cores (CM0 and CM4). The *fslesl* folder is taken from the official FSLESL release 4.1, and it is fully compatible with the official release. The library names are changed for easier use in the available IDEs (in *pmsm_package*). See freescale.com/fslesl for more information about FSLESL.

- *mc_drivers*—contains the source and header files used to initialize and run motor-control applications.
- *mid*—contains the source code for automated parameter-identification routines of the motor. See *Automated PMSM Parameter Identification* (document AN4986) for more information.
- *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 taken from Kinetis SDK. You can find all information about the files used in this port of SDK at freescale.com/KSDK.
- *state_machine*—contains state machine functions for the Fault, Initialization, Stop, and Run states.

The */src/projects_demos/* folder is linked to the "*build_demos*" project, and it contains all source files used in the workspace. These files are specific for each project available in the build folders. They specify peripheral initialization routines, FreeMASTER initialization, 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_pmsm_appconfig.h*—contains definitions of constants for the application control processes (parameters of the motor and regulators, and the constants for other vector control-related algorithms). When you tailor the application for a different motor using the Motor Control Application Tuning Tool (MCAT), the tool generates this file at the end of the tuning process. Prefix "*m1_*" (meaning motor 1) is used in one-motor applications, this number designates the motor number in multi-motor applications.
- *m1_state_machine.c*—contains 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 for 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*—contains fault macros and control structure.
- *board/app_init.c*—contains application initialization, UART (FreeMASTER initialization), and initialization functions for buttons and LEDs.
- *board/app_init.h*—header file for *app_init.h*, contains LED and UART macros.
- *board/freemaster_cfg.h*—FreeMASTER common configuration file.
- *board/mcdrv_<board_MCU>.c*—contains motor-control driver peripherals initialization functions, specific for the board and MCU used.
- *board/mcdrv_<board_MCU>.h*—header file for *board/mcdrv_<board_MCU>.c*. This file contains macros for changing the FlexTimer PWM period, and ADC channels assigned to phase currents and board voltage.

4.2. IDE workspaces structure

The structure of workspaces (Section 6, “Building and debugging application”) is different to the structure described in these sections, but it uses the same files. The different organization is chosen due to a better manipulation with folders and files in workplaces, and due to the possibility of adding or removing files and directories.

4.3. Application types

The PMSM package includes two projects for each board used (“*build_demos*“ and “*project_ref_solutions*“). The „build_demos“ project serves for demonstration purposes, and includes only the functions and routines neccessary for motor control. This project includes only the speed Field-Oriented Control (FOC). It does not include automatic Motor Identification Parameters (MID) and other types of motor control, such as scalar or vector control.

The “*build_ref_solutions*“ project includes all available functions and routines, MID functions, scalar and vector control of the motor, FOC control, and FreeMASTER MCAT project. This project serves for development and testing purposes. Building of a project is described in Section 6, “Building and debugging applications.“

5. Tools

Install the following software on your PC to run and control the PMSM sensorless application properly:

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

6. Building and debugging applications

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

6.1. IAR Embedded Workbench

You can use IAR Embedded Workbench to compile and run the demonstration or solution projects. The first step is to choose the demonstration or solution project, development board, and MCU. For example, to run a demonstration project for Freedom development platform and Kinetis KV10 MCU, the project is located in the *pmsm_package\build_demos\frdm-kv10z\iar* folder, which contains all necessary files. Double-click “*projects_demos_frdm-kv10z.eww*” to run this project. The point 1 in Figure 15 shows the IAR workspace with “*projects_demos_frdm-kv10z*” opened.

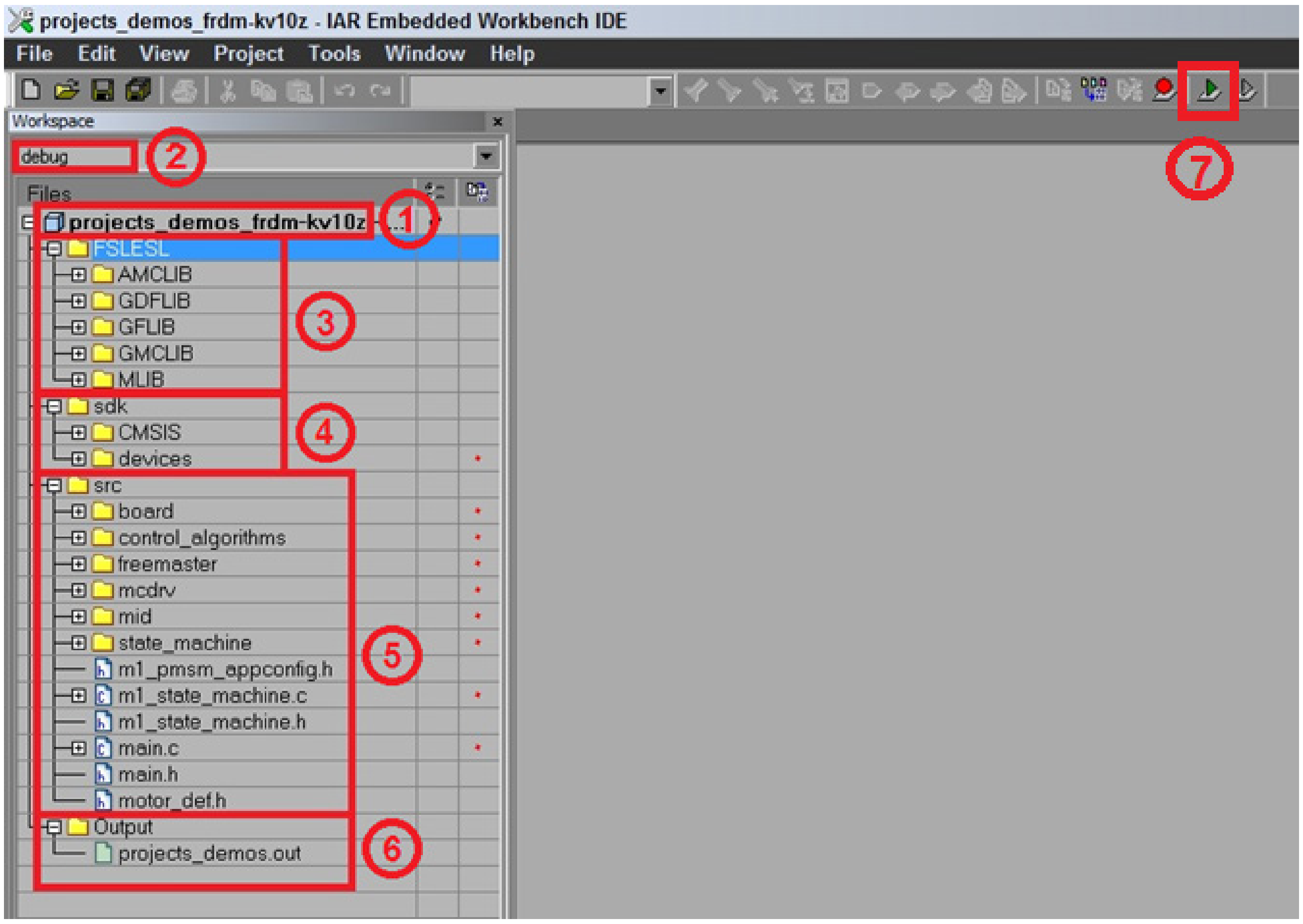


Figure 15. IAR Embedded Workbench

The project opened in IAR Embedded Workbench is fully configured and includes all necessary source and header files required by the application, such as startup code, clock configuration, and peripherals configuration. You can choose between two compiling conditions ("debug" or "release"), shown in Figure 3, 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 smaller flash (for example KV10Z32).

The source code shown in Figure 15 includes these source files and folders:

- Point 3—the FSLESL mathematical library source folder contains header files for 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 freescale.com/fslesl.
- Point 4—the *sdk* folder contains startup routines for the MCU, system initialization and clock definition, linker file, and header file for the MCU. It also contains the basic CMSIS routines for interrupt handling.

- Point 5—the *src* folder contains the application source code. The structure of this folder is different than the one mentioned in Section 4, "Project file structure." However, it is composed of the same files, and it is organized to fit into the IDE workspace.
- Point 6—shows the output file generated by the compiler, and ready to use with the default debugger (PEMicro – OpenSDA). This debugger is set as default for Tower boards, and can be changed in project options by right-clicking Point 1, selecting "Options", and clicking "Debugger". You can start the project debugging by clicking Point 7 (Figure 15).

6.2. Kinetis Design Studio (KDS)

Kinetis Design Studio (KDS) is an IDE tool that you can use to develop and test software for Freescale MCUs. It supports a wide range of Kinetis devices, such as the powerful K series, low-power KL series, and KV series targeted at motor control. KDS includes tools for compiling, linking, and debugging of source code. KDS supports a wide range of debuggers, such as PEMicro or Jlink (and others). You can download the latest release of KDS from the official Freescale website freescale.com/kds. For installation and configuration, see *Kinetis Design Studio V3.0.0 User Guide* (document KDSUG).

To open a demonstration or solution project, 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 at *pmsm_package\build_demos\frdm-kv10z\kds*, run KDS IDE from the default installation path or from installed programs, and then perform these steps:

- Click the "File" menu in the top-left corner of the IDE, and select "Import" (Figure 16).

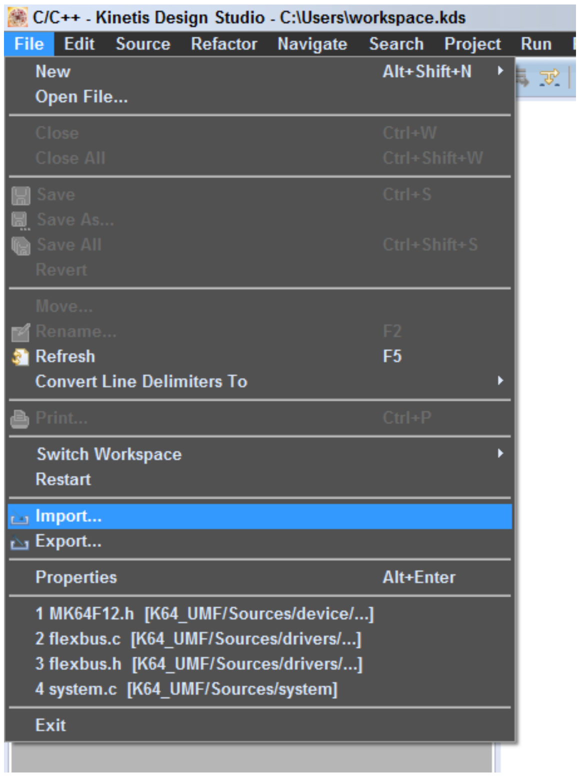
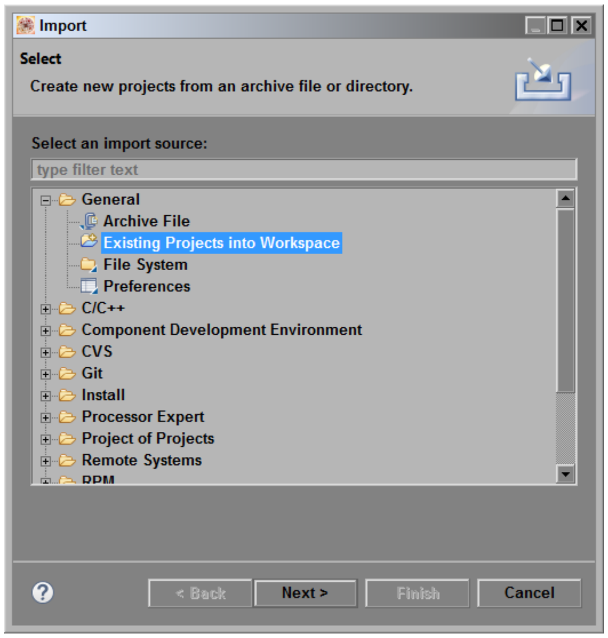


Figure 16. KDS—importing project

- The “Import“ window opens. Highlight “Existing Projects into Workspace“ in “General“ folder, and click “Next“ (Figure 17).



Import

Select

Create new projects from an archive file or directory.

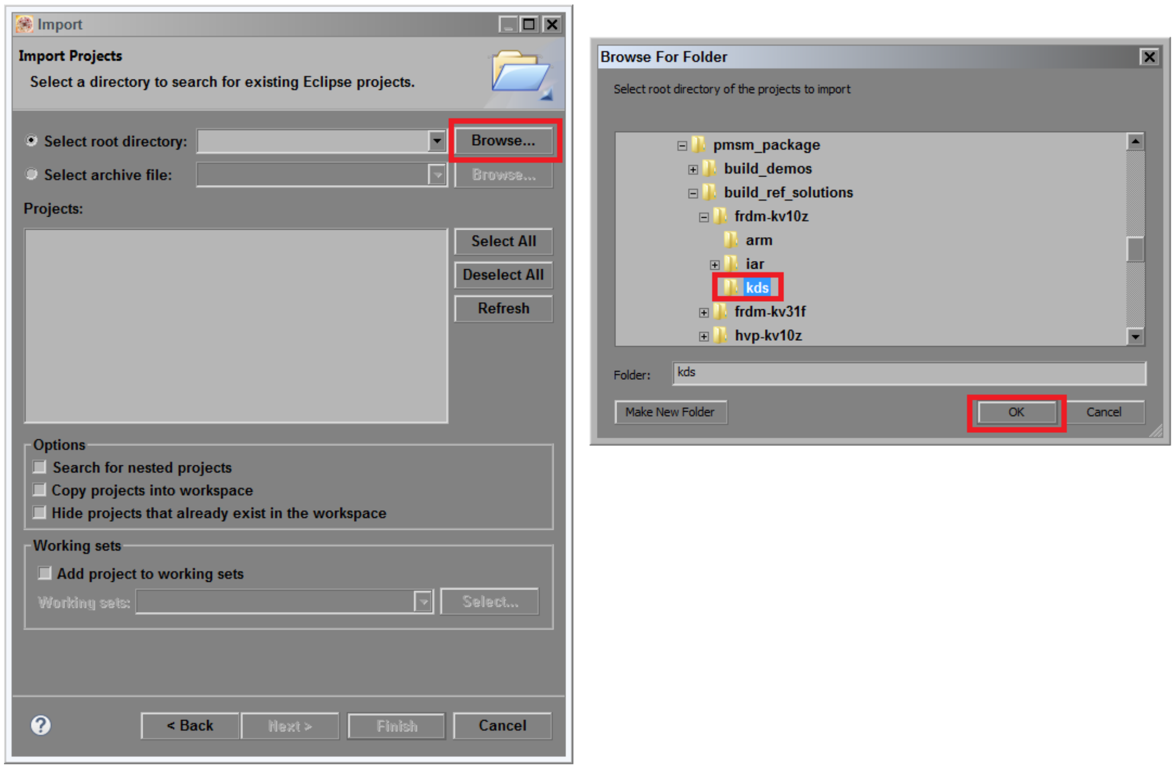
Select an import source:

type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
- C/C++
- Component Development Environment
- CVS
- Git
- Install
- Processor Expert
- Project of Projects
- Remote Systems
- RPM

Figure 17. KDS—importing project

- The “Import” window opens. Click “Browse”, and then locate the project “*//pmsm_package\build_demos\frdm-kv10z\kds*”. Click “OK”.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

Select root directory:

Select archive file:

Projects:

Options

- Search for nested projects
- Copy projects into workspace
- Hide projects that already exist in the workspace

Working sets

- Add project to working sets

Working sets:

Browse For Folder

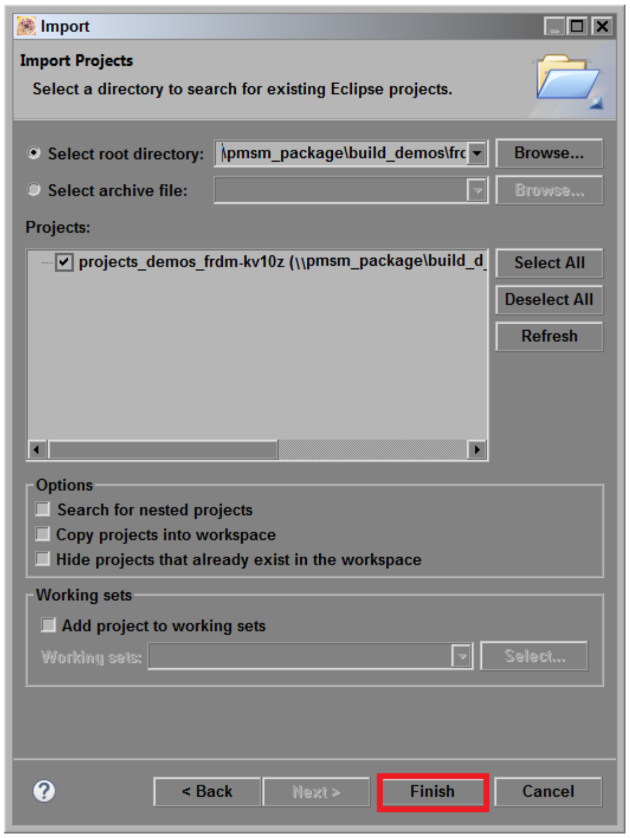
Select root directory of the projects to import

- pmsm_package
  - build_demos
  - build_ref_solutions
    - frdm-kv10z
      - arm
      - iar
      - kds
    - frdm-kv31f
    - hvp-kv10z

Folder: kds

Figure 18. KDS—importing project

- Confirm the project by clicking “Finish”.


Import

Import Projects

Select a directory to search for existing Eclipse projects.

- Select root directory: \pmsm_package\build_demos\frc
- Select archive file:

Projects:

[✓] projects_demos_frdm-kv10z (\\pmsm_package\build_d

Options

- [ ] Search for nested projects
- [ ] Copy projects into workspace
- [ ] Hide projects that already exist in the workspace

Working sets

- [ ] Add project to working sets

Working sets:

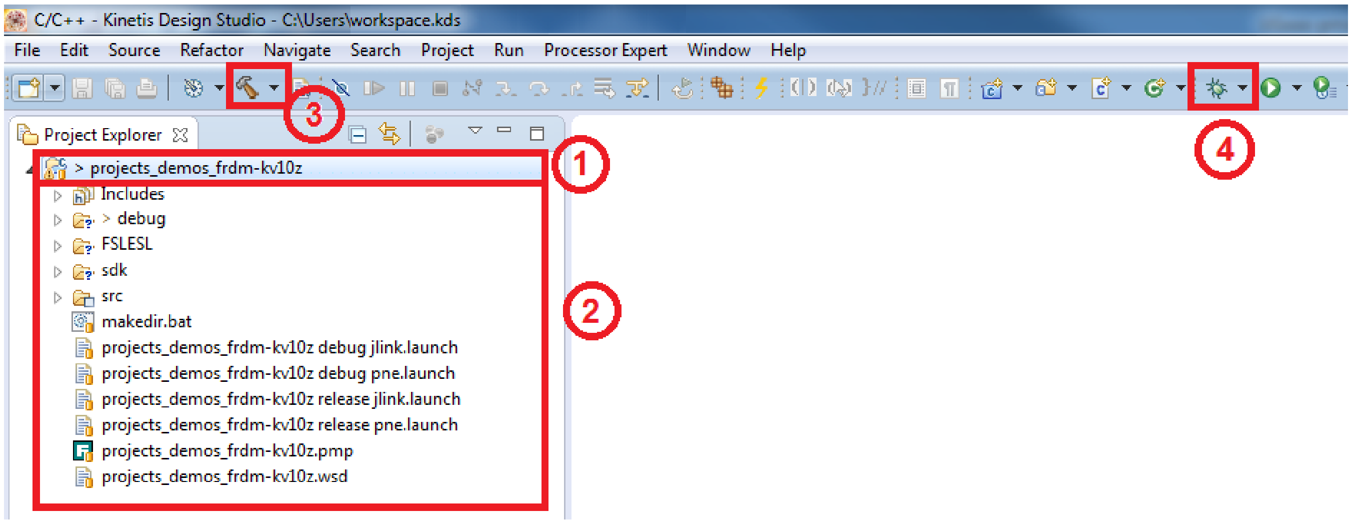
Figure 19. KDS—importing project

The project is now imported to Kinetis Design Studio (Figure 20). Point 1 shows the imported project in “Project Explorer”, and Point 2 shows the source code of this project. You can build the project by clicking the “build” icon (Point 3) where the “debug” configuration is set as default. You can change the configuration to “release”. 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 smaller flash (for example KV10Z32).


C/C++ - Kinetis Design Studio - C:\Users\workspace.kds

File Edit Source Refactor Navigate Search Project Run Processor Expert Window Help
Project Explorer
3
4
1
2

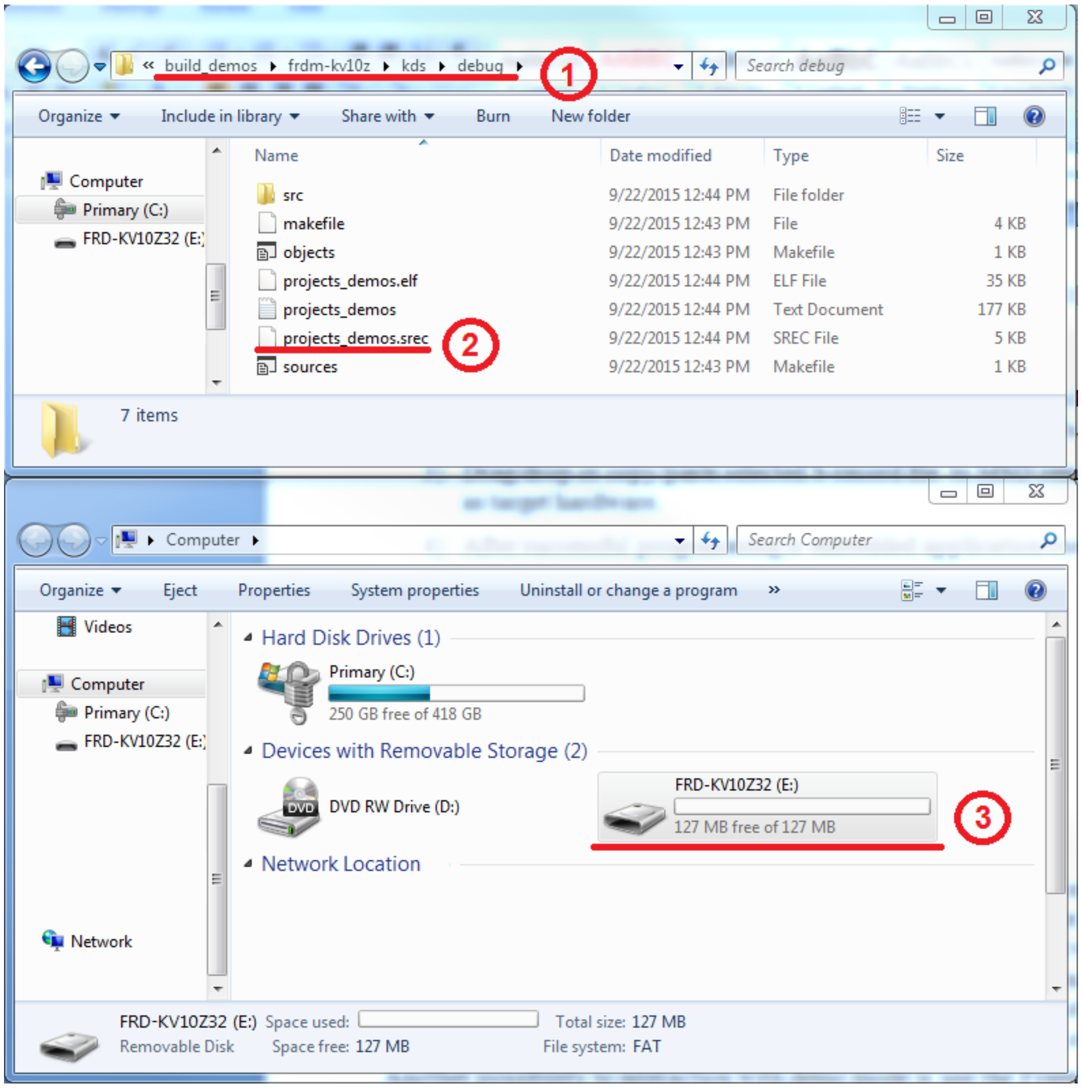
> projects_demos_frdm-kv10z
Includes
> debug
FSLESL
sdk
src
makedir.bat
projects_demos_frdm-kv10z debug jlink.launch
projects_demos_frdm-kv10z debug pne.launch
projects_demos_frdm-kv10z release jlink.launch
projects_demos_frdm-kv10z release pne.launch
projects_demos_frdm-kv10z.pmp
projects_demos_frdm-kv10z.wsd

Figure 20. KDS PMSM project

When the project is compiled, either the "debug" or "release" directories are created, depending on the build condition selected. An *.elf binary file is created in one or both of those directories. Now use the debugger (Point 4 in Figure 20). You can use the predefined debugger (such as PEMicro— 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. OpenSDA debugger

Freescale development boards include OpenSDA serial and debugger adapter, which is used as a bridge for serial and debug communication between target processor and Host computer. This debugger provides a virtual serial port on the host computer, which you can access as a "COM" port.
The embedded target is connected to the UART peripheral. In the same time, you can control the debug interface that controls the JTAG or SWD debug interfaces in the target development platform.
The OpenSDA debug interface provides Mass Storage Device Flash Programmer (MSD Flash Programmer), which is an 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. The raw binary Motorola S-Record files that are copied to the drive are programmed directly into the target device memory. The MSD Flash Programmer is designed to program a specific target configuration. It does not support verification or configuration, and it is not recommended as a production programmer.



Name	Date modified	Type	Size
src	9/22/2015 12:44 PM	File folder	
makefile	9/22/2015 12:43 PM	File	4 KB
objects	9/22/2015 12:43 PM	Makefile	1 KB
projects_demos.elf	9/22/2015 12:44 PM	ELF File	35 KB
projects_demos	9/22/2015 12:44 PM	Text Document	177 KB
projects_demos.srec	9/22/2015 12:44 PM	SREC File	5 KB
sources	9/22/2015 12:43 PM	Makefile	1 KB

Primary (C:)
250 GB free of 418 GB

DVD RW Drive (D:)

FRD-KV10Z32 (E:)
127 MB free of 127 MB

FRD-KV10Z32 (E:) Space used: Total size: 127 MB
Removable Disk Space free: 127 MB File system: FAT

Figure 21. OpenSDA MSD

To load the generated application directly to the target MCU, perform these steps (applicable in Windows):

1. Open Windows Explorer.
2. Locate the generated S-record file (.srec, .s19 or .bin file extension) (for example *pmsm_package\build_demos\<board_MCU>\<tool>\debug\projects_demos.srec*, as shown in Figure 21, Point 1).
3. Drag/drop or copy/paste the selected S-record file (Figure 21, Point 2) to the MSD removable drive with the volume labelled as the target hardware (Figure 21, Point 3).
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 can be suspended, and these warnings do not show during the compiling process. One of such special cases is the "unused function" warning, where the function is implemented in the source code with its body, but this function is not used. This case can occur in situations where you implement the function as a supporting function for better usability, but you don't use the function for any special purposes for a while.

The PMSM project contains two projects—the "demo" project and the "reference solutions" project. In these projects (especially in IDEs), some warnings are suppressed. These warnings are mentioned below, and the other warnings are set to a standard configuration.

IAR Embedded Workbench suppresses these warnings:

- Pa082—undefined behavior; the order of volatile accesses is not defined in this statement.
- Pe177—<entity> is declared, but not referenced.

Kinetis Design Studio (KDS) suppresses these warnings:

- parentheses—parentheses are omitted in certain contexts.
- unused function—function is defined, but not used.
- uninitialized—variable is used, but not initialized.
- main—due to optimization, main is void (not integer). It does not return a value.

There are no other warnings shown during the compiling process by default.

7. User interface

The application contains the demo application mode to demonstrate the motor rotation. You can operate it either using the user button, or using FreeMASTER. Tower and Freedom boards include a user button associated with a 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 button, the demo mode starts; when you press the same button again, the application stops and transitions back to the STOP state.

The other way to interact with the demo mode is to 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 via the RS232 interface; this interface is provided by the OpenSDA debugger included in the boards. To run the FreeMASTER application including the MCAT tool, double-click the **.pmp* file located in the *\pmsm_package\build_demos\<board_MCU>\<tool>* folder. The FreeMASTER application starts, and the environment is created automatically, as defined in the **.pmp* file. This application enables you to tailor the PMSM sensorless application demonstration for any motor.

You can control the application using these two interfaces:

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

7.1. Button control

When you press the corresponding button, the demonstration mode switches on (or off if it is already switched on). See this table to identify the correct control button on your development board:

Table 7. Control button assignment

Board name	Control button
TWR-KV10Z32	SW1
TWR-KV31F120	SW3
FRDM-KV10Z	SW2
FRDM-KV31F	SW2
HVP-KV10Z	N/A
HVP-KV31F	N/A

7.2. Remote control using FreeMASTER

The remote operation is provided by FreeMASTER via the USB interface. FreeMASTER 2.0 is required for the application to operate properly. You can download FreeMASTER 2.0 at www.freescale.com/freemaster.

Perform these steps to control a PMSM motor using FreeMASTER:

1. Open the FreeMASTER project file (.pmp). Open the FreeMASTER project corresponding to the IDE used:
   - For KDS, the FreeMASTER project file is located at this path: *Project_directory/build/kds/*
   - For IAR Embedded Workbench, the FreeMASTER project file is located at this path: *Project_directory/build/iar/*
2. Click the communication button (the red STOP button in the top left-hand corner) to establish the communication.

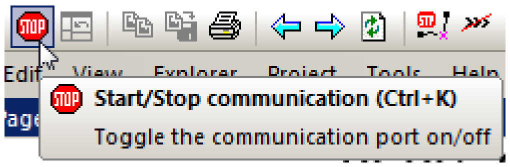

Edit
View
Explorer
Project
Tools
Help
Start/Stop communication (Ctrl+K)
Toggle the communication port on/off

Figure 22. Red STOP button placed in top left-hand corner

If the communication is established successfully, the FreeMASTER communication status in the bottom right-hand corner changes from “Not connected” to “RS232 UART Communication; COMxx; speed=19200”. Otherwise, the FreeMASTER warning popup window appears.

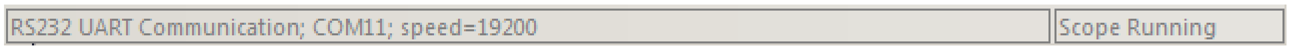
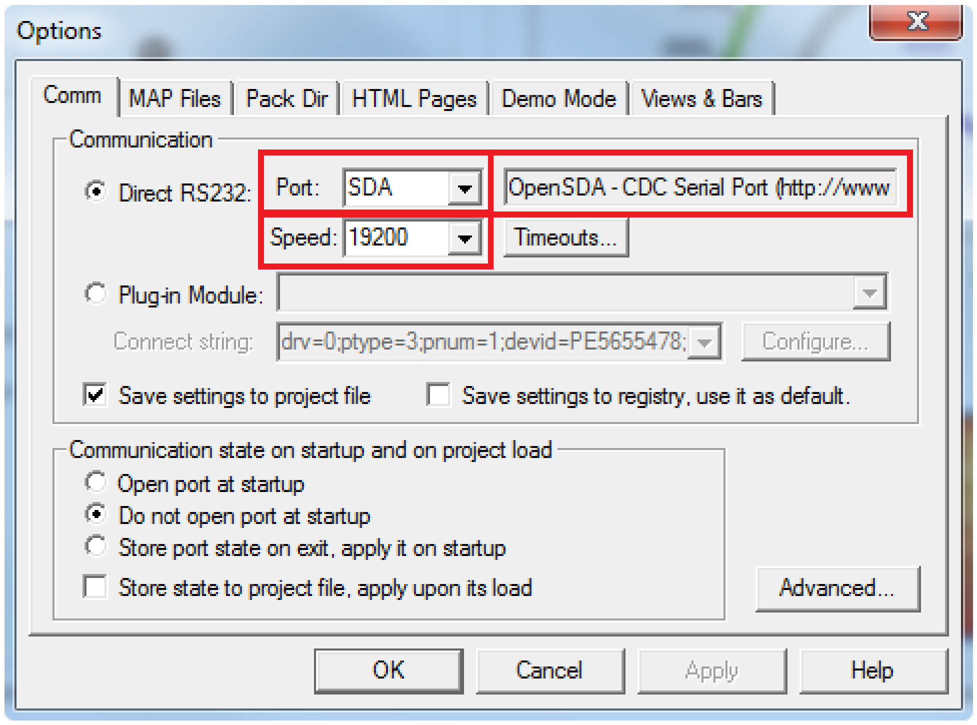

RS232 UART Communication; COM11; speed=19200 | Scope Running

Figure 23. FreeMASTER—communication is established successfully

3. Control the PMSM motor using the control page.

If the communication is not established successfully, perform these steps:

1. Go to “Project->Options->Comm” tab and make sure that “SDA” is set in the “Port” option, and the communication speed is set to 19200 bps.


Options

Comm | MAP Files | Pack Dir | HTML Pages | Demo Mode | Views & Bars

Communication

(●) Direct RS232: Port: SDA — OpenSDA - CDC Serial Port (http://www

Speed: 19200 — Timeouts...

( ) Plug-in Module:

Connect string: drv=0;ptype=3;pnum=1;devid=PE5655478; — Configure...

[✓] Save settings to project file [ ] Save settings to registry, use it as default.

Communication state on startup and on project load

( ) Open port at startup

(●) Do not open port at startup

( ) Store port state on exit, apply it on startup

[ ] Store state to project file, apply upon its load

Advanced...

OK | Cancel | Apply | Help

Figure 24. FreeMASTER communication setup window

2. If “OpenSDA-CDC Serial Port” is not printed out in the message box next to the “Port” dropdown menu, unplug and then plug in the USB cable, and reopen the FreeMASTER project.
3. Make sure to supply your development board from a sufficient energy source. Sometimes the PC USB port is not sufficient for supplying the development board.

7.2.1. Control page

After launching the application and performing all necessary settings, you can control the PMSM motor using the FreeMASTER control page. The FreeMASTER control page contains:

- Speed gauge—shows the actual and required speeds.
- Required speed—sets up the required speed.
- DC-bus voltage—shows the actual DC-bus voltage.

- DC-bus current—shows the actual DC-bus current.
- Current limitation—sets up the DC-bus current limit.
- Demo mode—turns the demonstration mode on/off.
- STOP button—stops the whole application.
- Notification—shows the notification about the actual application state (or faults).

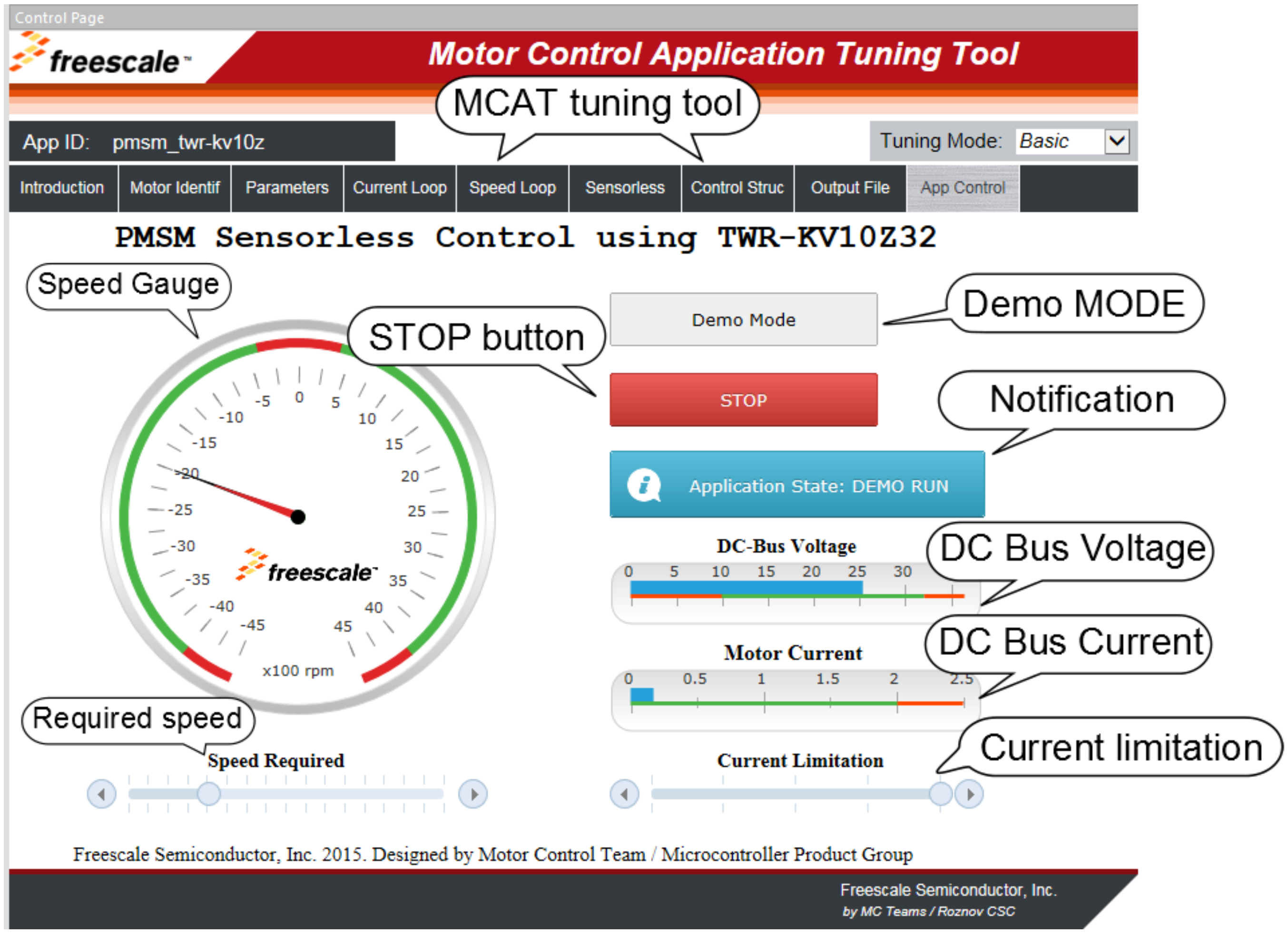


Figure 25. FreeMASTER control page

The MCAT tuning tool is available only for FreeMASTER projects from *build_ref_solutions*, and it is not available for the demonstration project. This tool is targeted at motor-control applications, where you can change the motor-control parameters, or measure their values. Each tab represents one submodule of the embedded-side control, and tunes its parameters. For more information, see *Motor Control Application Tuning (MCAT) Tool for 3-Phase PMSM* (document AN4642).

Here are the basic instructions:

- To start the motor, set up the required speed using the speed slider.
- In case of a fault, click on the fault notification to clear the fault.
- Click the “Demo Mode” button to turn the demonstration mode on/off.
- Click the “STOP” button to stop the motor.

8. Performing basic tasks

8.1. Running the motor

1. Assembly the Freescale hardware according to the instructions in Section 3, “Hardware setup.”
   a) In case of Tower development board follow these instructions.
   b) In case of Freedom development board follow these instructions.
   c) In case of High-Voltage Platform follow these instructions.
2. Flash the correct project into the target device via the OpenSDA debug interface.
3. Open the FreeMASTER project, and establish the communication between the MCU and the PC according to the instructions in Section 7.2, “Remote control using FreeMASTER.”
4. Set up the required motor speed using the speed slider, as shown in Figure 25.

8.2. Stopping the motor

1. Click the “STOP” button in the FreeMASTER control page (Figure 25).
2. Set the speed to zero using the speed slider.
3. In case of emergency, turn off the power supply.

8.3. Clearing the fault

To clear the fault, click the fault notification in the control page (Figure 25).

8.4. Turning the demonstration mode on/off

1. If you use the FreeMASTER control page, click the “Demo” button.
2. If you don’t use the FreeMASTER control page, push the corresponding button on your development board to turn demonstration mode on/off, as described in Section 7.1, “Button control.”

9. Acronyms and abbreviations

Table 8. Acronyms and abbreviations

Term	Meaning
AC	Alternating Current
AN	Application Note
DRM	Design Reference manual
FOC	Field-Oriented Control
MCAT	Motor Control Application Tuning tool
MCU	Microcontroller
MSD	Mass Storage Device
PMSM	Permanent Magnet Synchronous Motor

10. References

The following references are available on www.freescale.com:

- Sensorless PMSM Field-Oriented Control (document DRM148).
- Tuning Three-Phase PMSM Sensorless Control Application Using MCAT Tool (document AN4912).
- Automated PMSM Parameters Identification (document AN4986).
- Motor Control Application Tuning (MCAT) Tool for 3-Phase PMSM (document AN4642).

11. Revision history

The following table summarizes the changes done to this document since the initial release.

Table 9. Revision history

Revision number	Date	Substantive changes
0	10/2015	Initial release.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
www.freescale.com/support

Document Number: PMSMSAPUG
Rev. 0
10/2015