NXP Semiconductors
User's Guide

Document Number: PMSMCRSPUG
Rev. 4, 09/2016

PMSM Control Reference Solution Package

By: Josef Tkadlec

1. Introduction

This user's guide provides a step-by-step guide on how to open, compile, and run Permanent Magnet Synchronous Motor (PMSM) projects. It describes the basic compiling steps for IAR® Embedded Workbench®, Kinetis Design Studio (KDS), and µVision® Keil® IDEs 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 platforms with Kinetis V and E series MCUs for motor-control applications. The development boards and supported MCUs are shown in Table 1. The Tower System modular development platform and NXP 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

		Platform		
		Tower	**Freedom**	**HVP**
Power stage		TWR-MC-LV3PH	FRDM-MC-LVPMSM	HVP-MC3PH
MCU	**KV10Z**	TWR-KV10Z32	FRDM-KV10Z	HVP-KV10Z32
	KV31F	TWR-KV31F120M	FRDM-KV31F	HVP-KV31F120M
	KV11Z	TWR-KV11Z75M	—	—
	KV46F	TWR-KV46F256	—	HVP-KV46F120M
	KV58F	TWR-KV58F220M	—	HVP-KV58F
	KE15Z	—	FRDM-KE15Z	—
	KE18F	TWR-KE18F	—	HVP-KE18F

3. Motor Control versus SDK Peripheral Drivers

The motor-control examples use the SDK peripheral drivers to configure the general peripherals such as the clocks, SPI, SIM, and ports. However, the motor control requires critical application timing because most of the control algorithm runs in a 100-µs loop. To optimize the CPU load, most of the peripheral hardware features are implemented for the PWM signal generation, analog signal sampling, and synchronization between the PWM and ADC units.

The standard SDK peripheral drivers do not support the configuration and handling of all required features. The motor-control drivers are designed to configure the critical MC peripherals (eflexPWM, FTM, ADC, and PDB).

It is highly recommended not to modify the default configuration of the allocated MC peripherals due to a possible application timing conflict. The particular *mcdrv_< board&MCU >.c* source file contains the configuration functions of the allocated peripherals.

4. 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.

4.1. 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.

4.2. 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

4.3. Running PMSM application on Tower System

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

- Tower board with a Kinetis V or E series MCU (TWR-KV10Z32, TWR-KV11Z75M, TWR-KV31F120M, TWR-KV46F150M, TWR-KV58F220M, or TWR-KE18F).
- 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 nxp.com or from distributors to easily build the hardware platform for the target application.

4.3.1. TWR-MC-LV3PH Tower System module

The 3-phase Low-Voltage Motor Control board (TWR-MC-LV3PH) is a peripheral Tower System Module, interchangeable across the Tower development platform. The phase voltage and current feedback signals are provided. These signals enable a variety of algorithms to control 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 plugging the TWR-MC-LV3PH module into the Tower System development platform, ensure that the jumpers on your TWR-MC-LV3PH module are configured as follows:

Table 4. TWR-MC-LV3PH jumper settings

Jumper	Setting	Function
J2	1-2	Selects internal analog power supply.
J3	1-2	Selects internal analog power reference (GND).
J10	2-3	Selects I_SENSE_C.
J11	2-3	Selects I_SENSE_B.
J12	2-3	Selects I_SENSE_A.
J13	2-3	Selects I_SENSE_C.
J14	2-3	Selects I_SENSE_A.

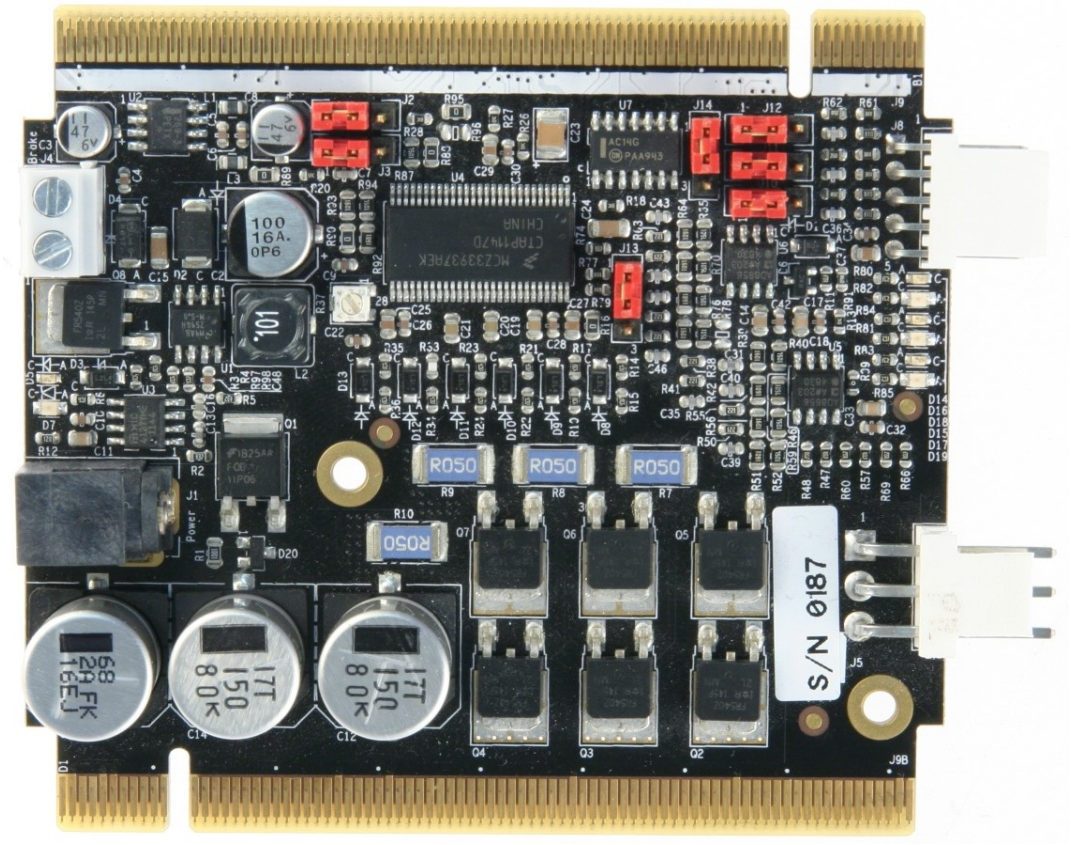

Figure 3. TWR-MC-LV3PH Tower System module

4.3.2. TWR-KV10Z Tower System module

The TWR-KV10Z32 is a development tool for the NXP Kinetis KV1x family of MCUs built around the ARM® Cortex®-M0+ 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.

To begin, configure the jumpers on the TWR-KV10Z32 and TWR-MC-LV3PH Tower System modules properly. The following table lists the specific jumpers and their settings for the TWR-KV10Z32 Tower System module.

Table 5. TWR-KV10Z jumper settings

Jumper	Setting	Jumper	Setting	Jumper	Setting
J1	2-3	J10	2-3	J21	3-4
J2	1-2	J11	open	J22	3-4
J3	2-3	J12	open	J25	open
J4	1-2	J13	open	J26	1-2
J5	1-2	J14	open	J27	1-2
J7	1-2	J18	2-3	J28	1-2
J8	2-3	J19	2-3	J29	1-2
J9	1-2	J20	2-3	—	—

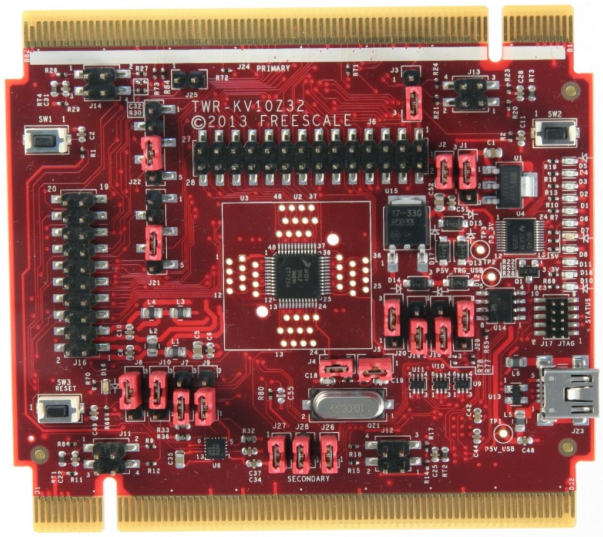

Figure 4. TWR-KV10Z Tower System module

4.3.3. TWR-KV11Z Tower System module

The TWR-KV11Z75M is a development tool for the NXP Kinetis KV1x family of MCUs built around the ARM Cortex-M0+ 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.

To begin, configure the jumpers on the TWR-KV11Z75M and TWR-MC-LV3PH Tower System modules properly. The following table lists the specific jumpers and their settings for the TWR-KV11Z75M Tower System module.

Table 6. TWR-KV11Z jumper settings

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

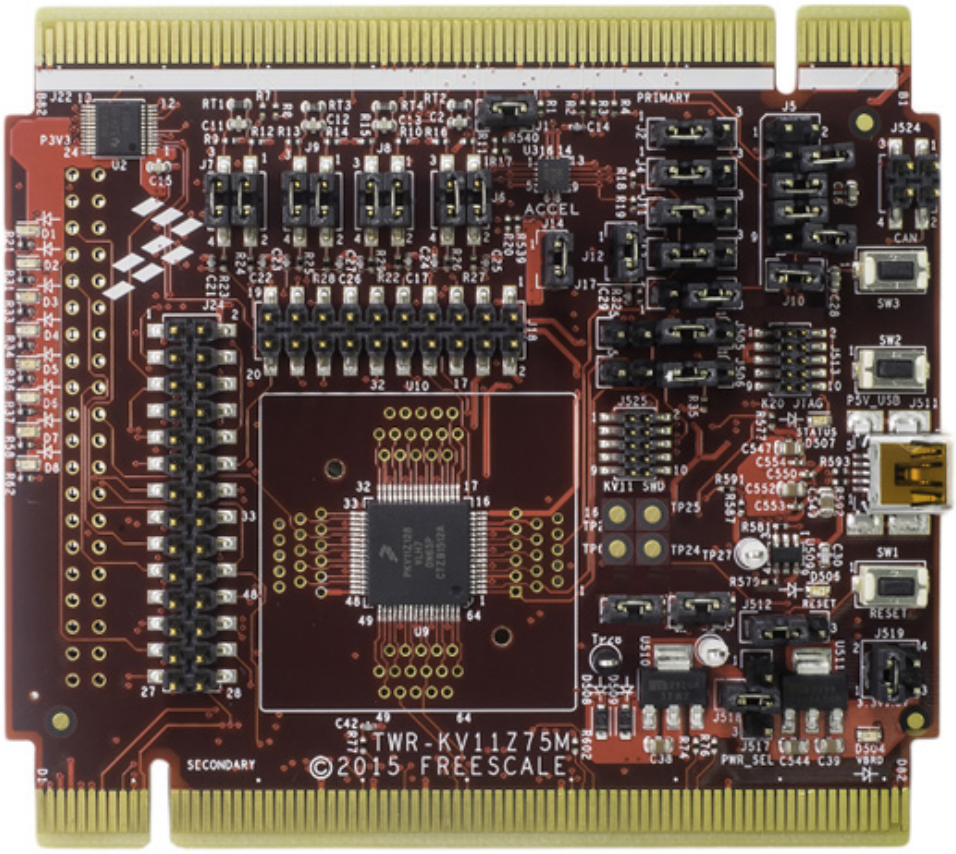

Figure 5. TWR-KV11Z Tower System module

4.3.4. TWR-KV31F Tower System module

The TWR-KV31F120M is a development tool for the NXP 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-KV31F120M and TWR-MC-LV3PH Tower System modules properly. The following table lists the specific jumpers and their settings for the TWR-KV31F120M Tower System module.

Table 7. TWR-KV31F jumper settings

Jumper	Setting	Jumper	Setting	Jumper	Setting
J1	1-2	J10	open	J17	1-2, 3-4, 5-6, 7-8
J3	1-2	J11	open	J20	open
J4	open	J12	open	J22	2-3
J5	1-3	J13	1-2	J29	2-3
J8	open	J14	1-2	J25	1-2
J9	open	J15	1-2	J26	2-3

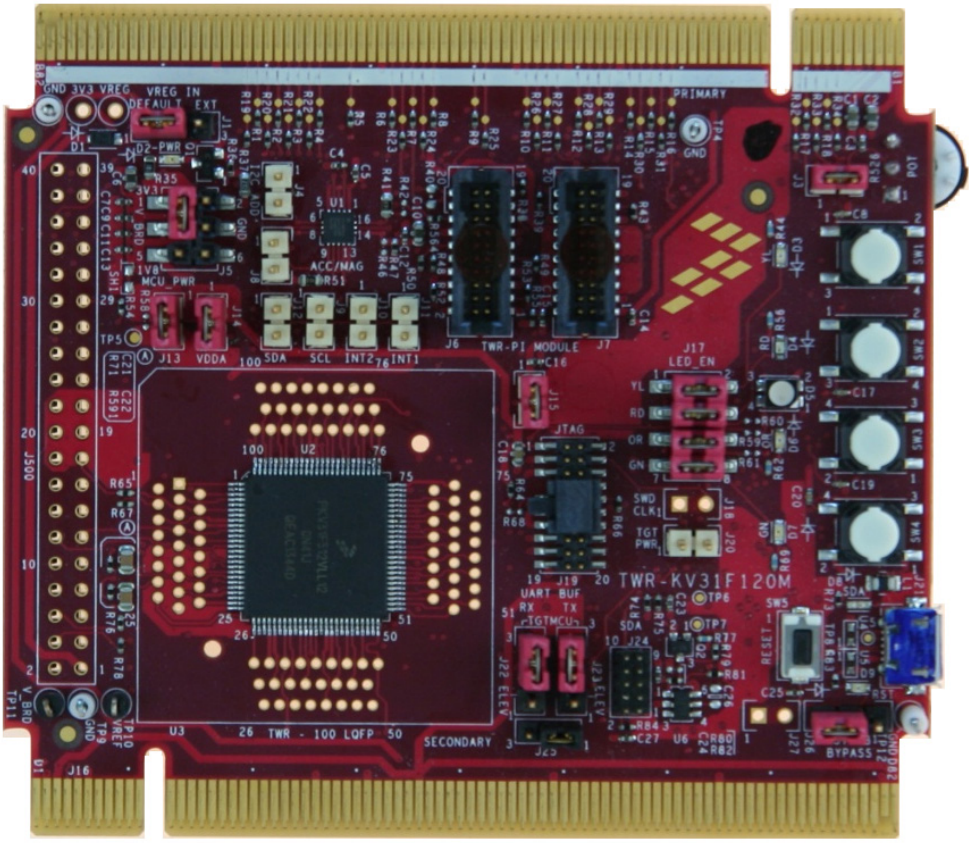

Figure 6. TWR-KV31F Tower System module

4.3.5. TWR-KV46F Tower System module

The TWR-KV46F150M is a development tool for the NXP Kinetis KV4x 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 motor-control peripherals, lots of memory (depending on the model used), and a powerful core.

To begin, configure the jumpers on the TWR-KV46F150M and TWR-MC-LV3PH Tower System modules properly. The following table lists the specific jumpers and their settings for the TWR-KV46F150M Tower System module.

Table 8. TWR-KV46F jumper settings

Jumper	Setting	Jumper	Setting	Jumper	Setting
J1	open	J16	open	J505	3-4
J2	open	J19	open	J506	3-4
J4	2-3	J20	1-2	J512	1-2
J5	open	J21	open	J514	2-3
J13	1-2, 3-4	J23	open	J517	2-3
J15	1-2	J25	1-2	J519	3-4

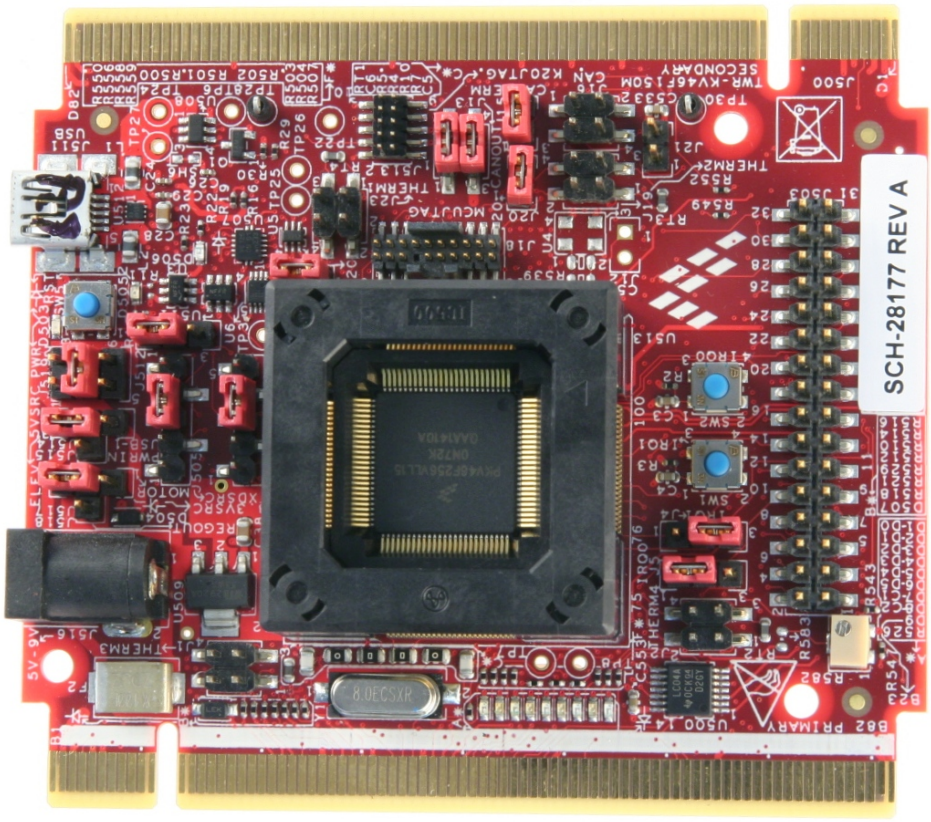

Figure 7. TWR-KV46F Tower System module

4.3.6. TWR-KV58F Tower System module

The TWR-KV58F220M is a development tool for the NXP 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 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. The following table lists the specific jumpers and their settings for the TWR-KV58F220M Tower System module.

Table 9. 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	1-2	J29	open
J9	open	J20	1-2	J30	1-2, 3-4, 5-6, 7-8
J10	1-2	J21	1-2	—	—

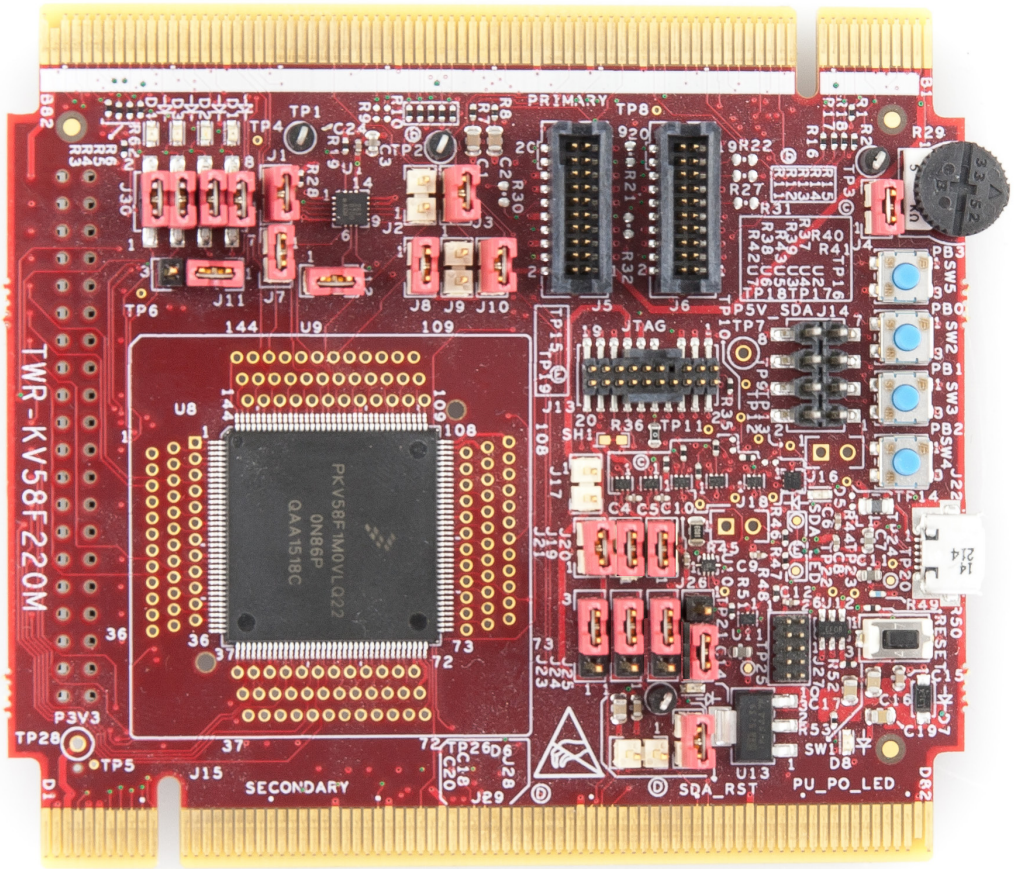

Figure 8. TWR-KV58F Tower System module

4.3.7. TWR-KE18F Tower System module

The TWR-KE18F is a development tool for the NXP Kinetis KE1xF 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-KE18F and TWR-MC-LV3PH Tower System modules properly. The following table lists the specific jumpers and their settings for the TWR-KE18F Tower System module.

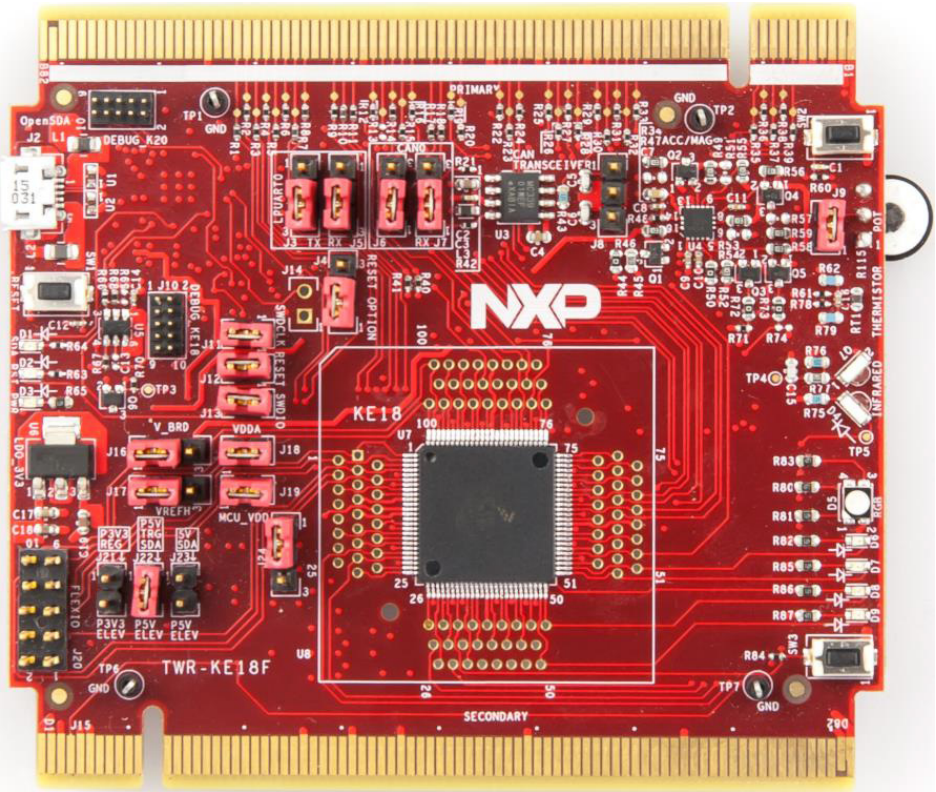

Figure 9. TWR-KE18F Tower System Module

4.3.8. Tower System assembling

1. Insert the Tower System 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 any USB port on the host PC via a USB cable.

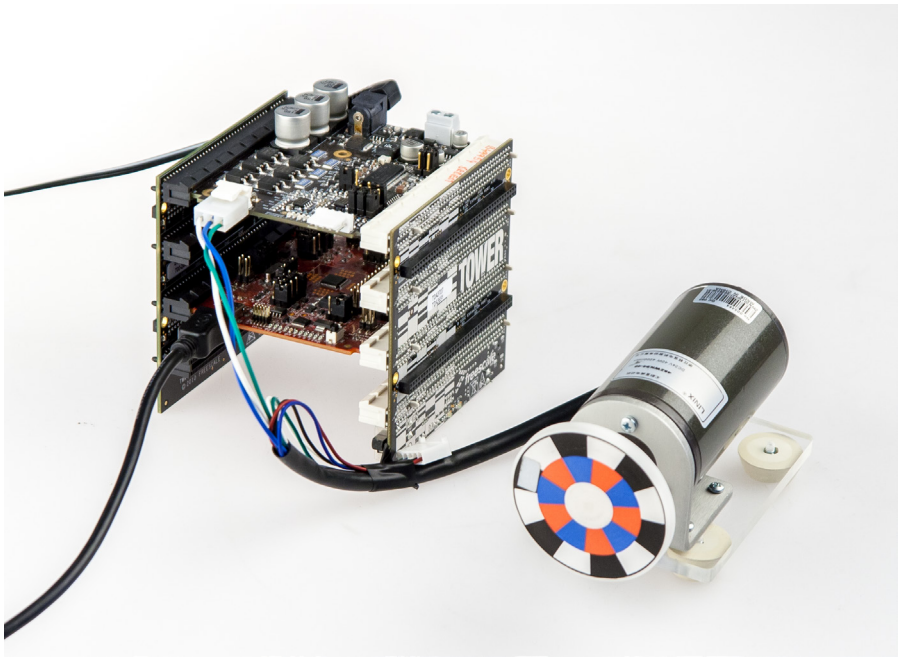

Figure 10. Assembled Tower System

4.4. NXP Freedom development platform

To run the PMSM application using the NXP Freedom development platform, you need these Freedom boards:

- Freedom board with a Kinetis V series MCU (FRDM-KV10Z, FRDM-KV31F, or FRDM-KE15Z).
- Three-phase low-voltage power Freedom shield (FRDM-MC-LV3PH) with included Linix motor.

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

4.4.1. FRDM-MC-LVPMSM

The FRDM-MC- LVPMSM low-voltage evaluation board (in a shield form factor) turns an NXP Freedom development board into a complete motor-control reference design compatible with existing Freedom development platforms (FRDM-KV31F, FRDM-KV10Z, and FRDM-KE15Z).

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

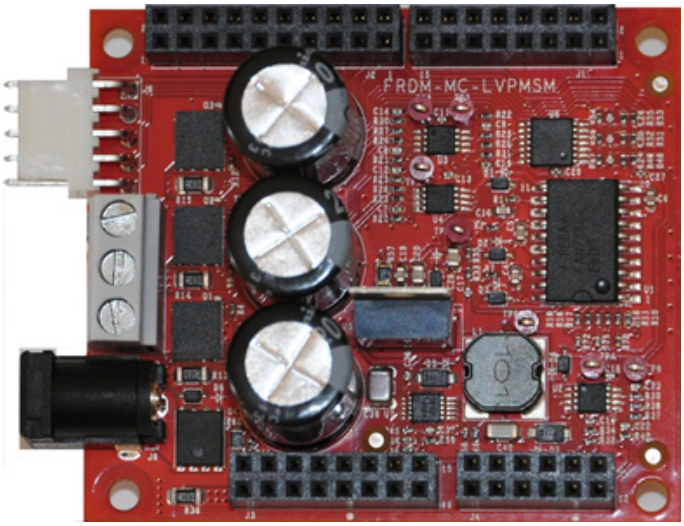

Figure 11. FRDM-MC-LVPMSM

4.4.2. 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 NXP open-source hardware embedded serial and debug adapter running an open-source bootloader.

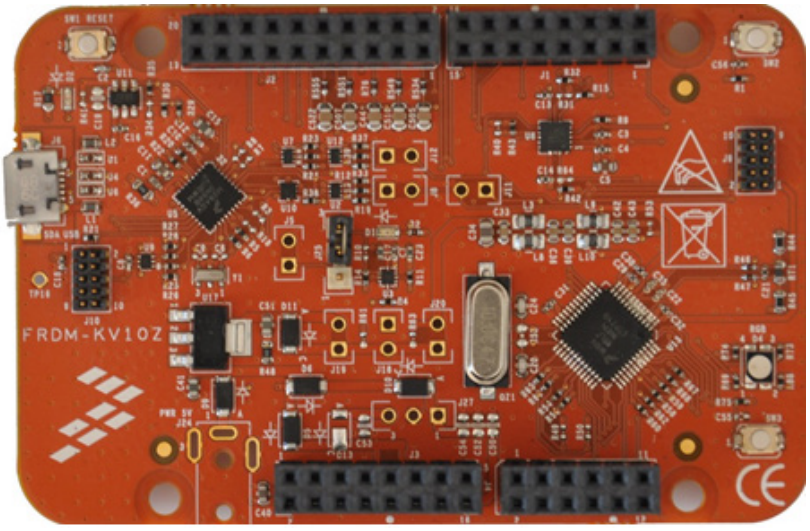

Figure 12. FRDM-KV10Z Freedom development board

4.4.3. 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 NXP 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.

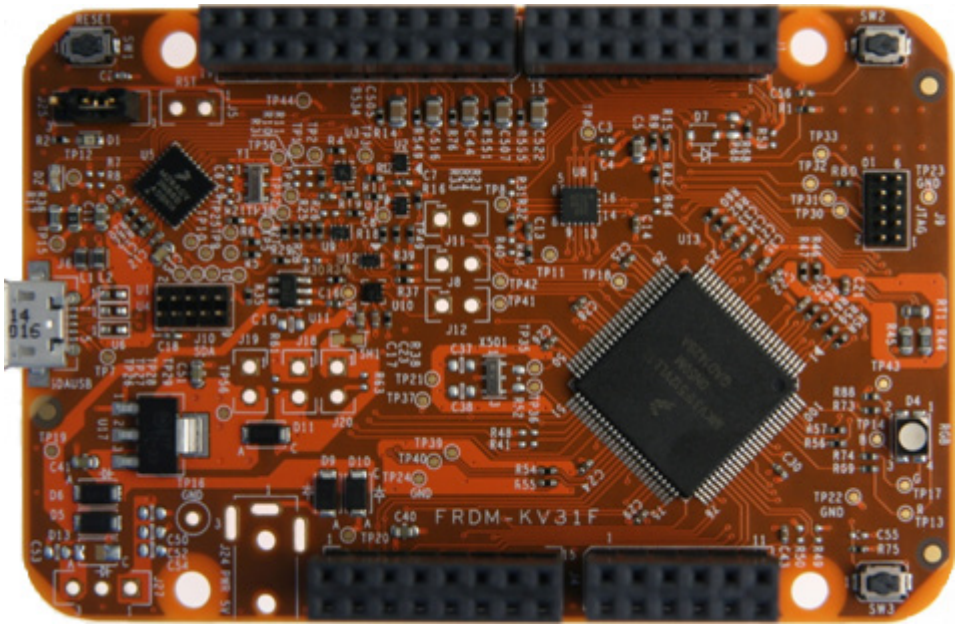

Figure 13. FRDM-KV31F development board

4.4.4. FRDM-KE15Z

The FRDM-KE15Z is a low-cost development tool for Kinetis KE1xZ family of MCUs built around the ARM Cortex-M0+ core. The FRDM-KE15Z hardware is form-factor compatible with the Arduino R3 pin layout, providing a broad range of expansion board options. The FRDM-KE15Z platform features OpenSDA, the NXP open-source hardware embedded serial and debug adapter running an open-source bootloader.

To begin, configure the jumpers on the FRDM-KE15Z Freedom System module properly. The following table lists the specific jumpers and their settings for the FRDM-KE15Z Freedom System module.

Table 10. FRDM-KE15Z jumper settings

Jumper	Setting	Jumper	Setting	Jumper	Setting
J7	1-2	J10	1-2	J15	2-3
J8	1-2	J14	1-2	—	—

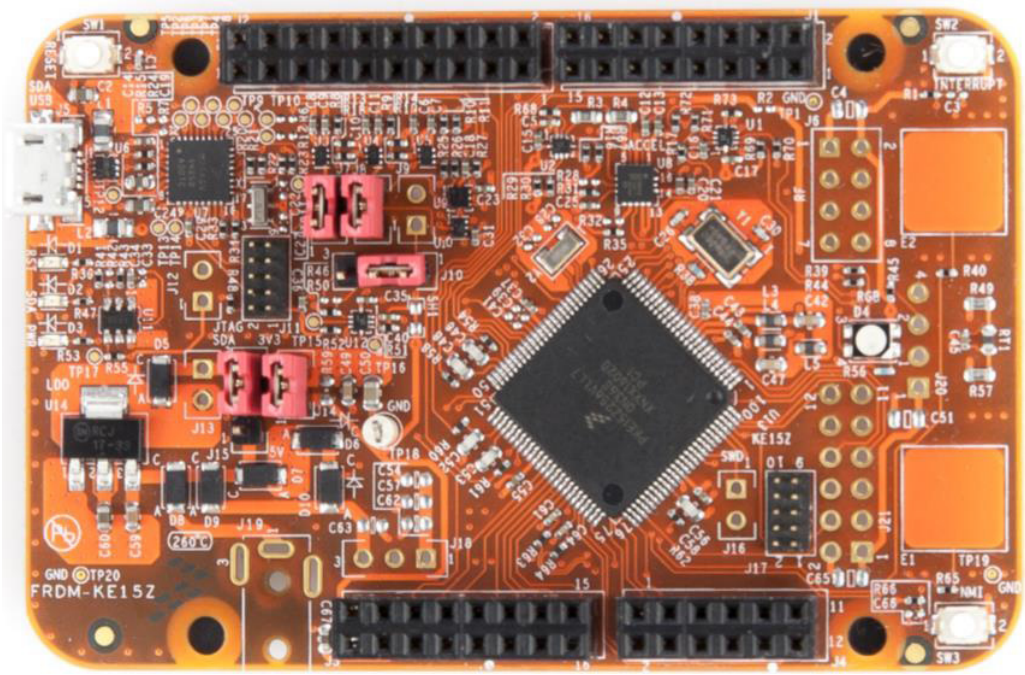

Figure 14. FRDM-KE15Z Freedom development board

4.4.5. NXP Freedom system assembling

1. Connect the FRDM-MC-LVPMSM shield on top of the FRDM-KVxxx board (there is only one possible option).
2. Connect the Linix 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 24 V DC power supply to the DC Power connector.

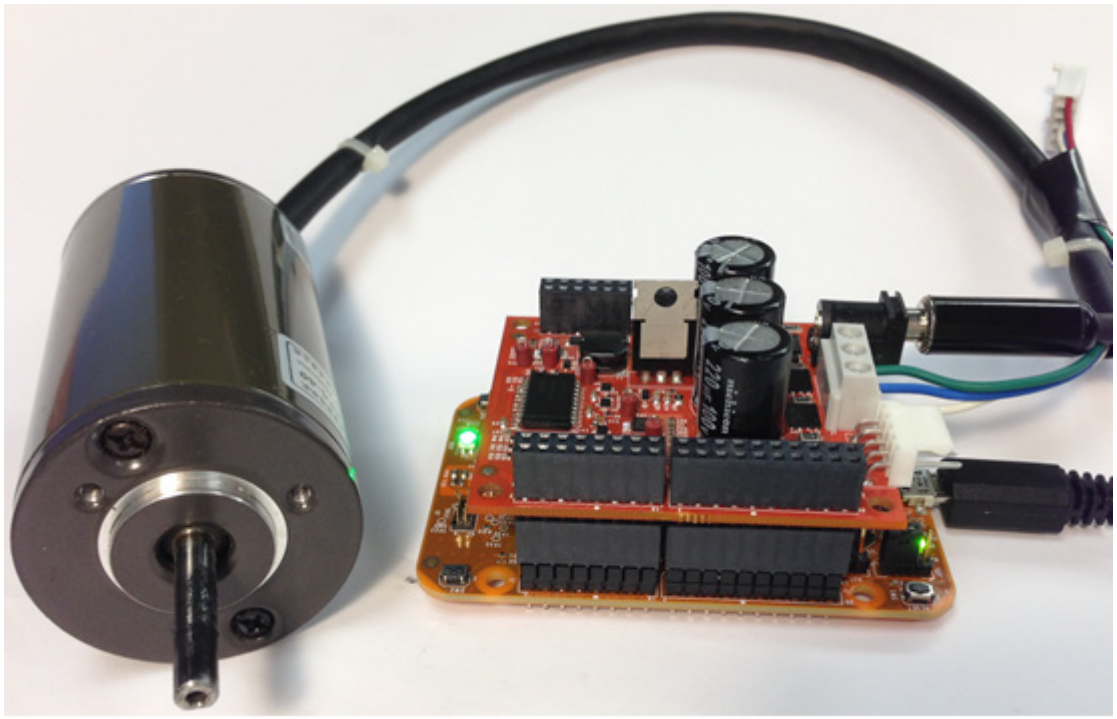

Figure 15. Assembled Freedom system

4.5. High-Voltage Platform

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

- HVP daughter card with a Kinetis V or E series MCU (HVP-KV10Z32, HVP-KV31F120M, HVP-KV46F150M, or HVP-KE18F).
- High-Voltage Platform power stage (HVP-MC3PH) (motor not included).

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

4.5.1. HVP-MC3PH power stage

The NXP High-Voltage Platform is an evaluation and development solution for Kinetis V and E 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 16. HVP-MC3PH High-Voltage Platform

4.5.2. HVP- KV10Z32 daughter card

The HVP-KV10Z MCU daughter card contains Kinetis KV10 family MCU built around the ARM Cortex-M0+ core running at 75 MHz and containing up to 32 KB of flash memory. This daughter card is developed for use in motor-control applications, together with the High-Voltage Platform power stage. This daughter card features OpenSDA, the NXP open-source hardware embedded serial and debug adapter running an open-source bootloader.

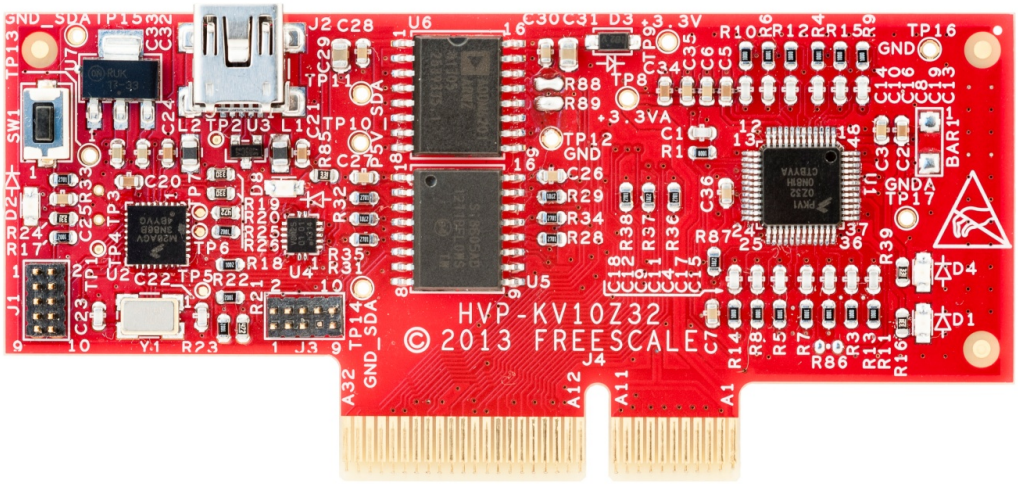

Figure 17. HVP-KV10Z32 daughter card

4.5.3. HVP-KV31F512M daughter card

The HVP-KV31F512 MCU daughter card contains a Kinetis KV3x family MCU built around the ARM Cortex-CM4 core with floating-point unit, running at 120 MHz and containing up to 512 KB of flash memory. This daughter card is developed for use in motor-control applications, together with the High-Voltage Platform power stage. This daughter card features OpenSDA, the NXP open-source hardware embedded serial and debug adapter running an open-source bootloader.

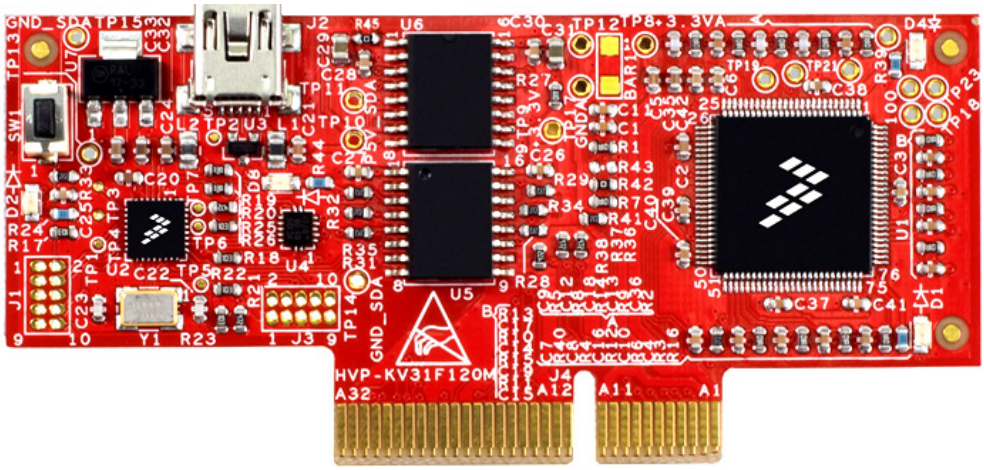

Figure 18. HVP-KV31F120M daughter card

4.5.4. HVP-KV46F150M daughter card

The HVP-KV46F150M MCU daughter card contains a Kinetis KV4x family MCU built around the ARM Cortex-CM4 core with floating-point unit, running at 150 MHz and containing up to 256 KB of flash memory. This daughter card is developed for use in motor-control applications, together with the High-Voltage Platform power stage. This daughter card features OpenSDA, the NXP open-source hardware embedded serial and debug adapter running an open-source bootloader.

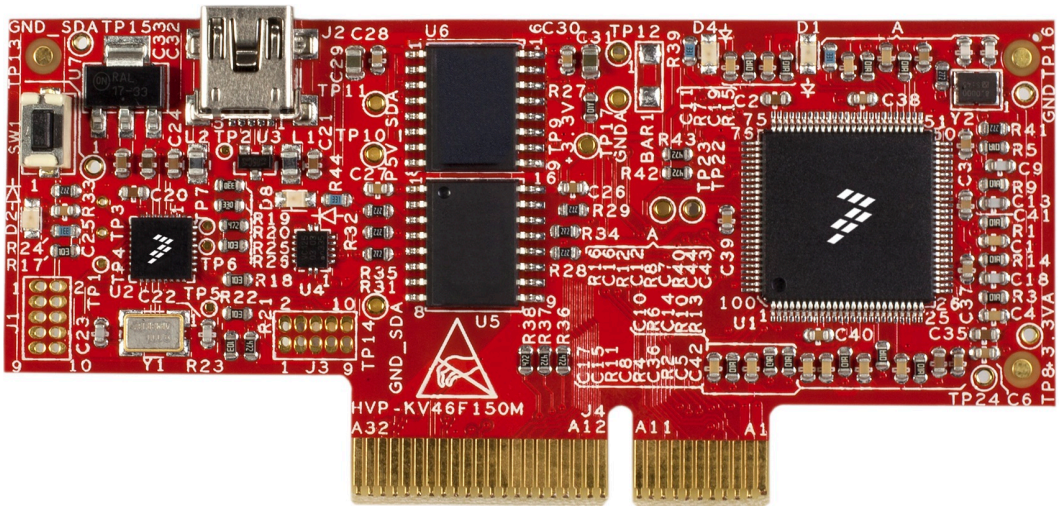

Figure 19. HVP-KV46F150M daughter card

4.5.5. HVP-KV58F daughter card

The HVP-KV58F MCU daughter card contains a Kinetis KV5x family MCU built around the ARM Cortex-CM7 core with a floating-point unit, running at 240 MHz and containing up to 1 MB of flash memory. This daughter card is developed for use in motor-control and connectivity applications, together with the High-Voltage Platform power stage. This daughter card features OpenSDA, the NXP open-source hardware embedded serial and debug adapter running an open-source bootloader.

Figure 20. HVP-KV58F daughter card

4.5.6. HVP-KE18F daughter card

The HVP-KE18F MCU daughter card contains a Kinetis KE1xF family MCU built around the ARM Cortex-CM4 core with a floating-point unit, running at 168 MHz and containing up to 256 KB flash memory. This daughter card is developed for use in motor-control applications, together with the High-Voltage Platform power stage. This daughter card features OpenSDA, the NXP open-source hardware embedded serial and debug adapter running an open-source bootloader.

Figure 21. HVP-KE18F development board

4.5.7. 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.

5. Project File Structure

The PMSM package includes source code for NXP 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.

5.1. PMSM package structure

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

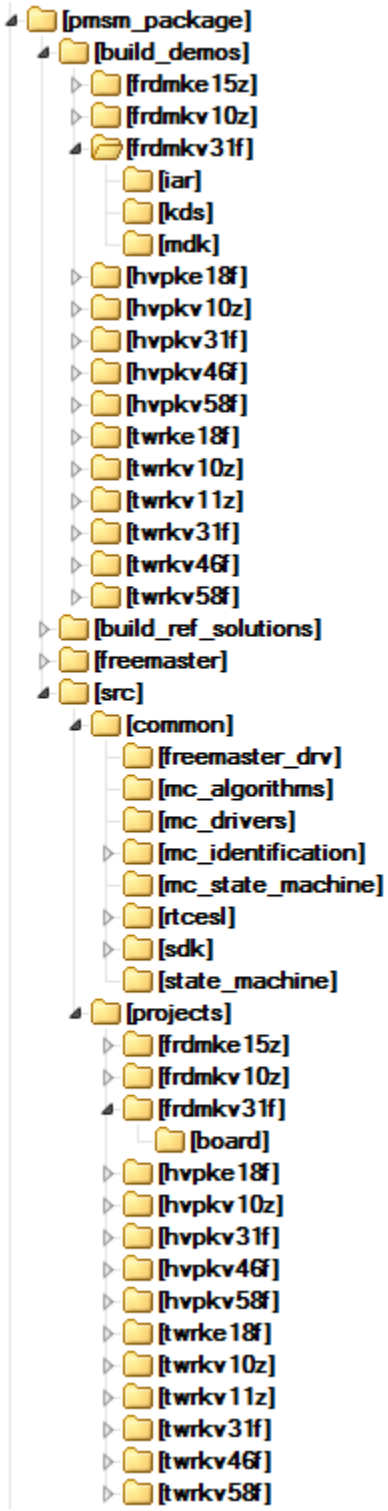

Figure 22. Package structure

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 IDE, 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 "*pmsm_demo_<platform_MCU>.eww*" to launch the IAR IDE. The projects are tuned to spin the reference motor.
- *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. The projects are tuned to spin the reference motor.
- *MDK* folder—contains configuration files for µVision Keil IDE. If Keil IDE is installed on your computer, open the project using Keil IDE. The projects are tuned to spin the reference motor.

The main reference solution project folder */build_ref_solutions/<platform_MCU>/* contains the *IAR* and *KDS* folders. The content of these folders is similar to the demonstration project folder (see above). The main difference is that the reference solution features motor-identification algorithms and scalar control algorithm.

The */freemaster/* folder contains auxiliary files for the MCAT tool. This folder contains also the FreeMASTER projects *pmsm_ref_sol.pmp* and *pmsm_demo.pmp*. You can open these files in the FreeMASTER tool and use them to control the *demo* or *ref_sol* applications respectively.

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*—contains platform and MCU-dependent source code sorted by <platform_MCU>.

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

- *freemaster_drv*—contains FreeMASTER header and source files.
- *rtcesl*—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 three subfolders ("*cm0*", "*cm4*", and "*cm7*") that contain precompiled library files for three types of ARM cores (CM0, CM4, and CM7). 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 in the available IDEs (in *pmsm_package*). See nxp.com/fslesl for more information about RTCESL.
- *mc_algorithms*—contains the main control algorithms used to control the FOC and speed control loop.
- *mc_drivers*—contains the source and header files used to initialize and run motor-control applications.

- *mc_identification*—contains the source code for automated parameter-identification routines of the motor. See the "PMSM Parameter Identification" section of *Sensorless PMSM Field-Oriented Control on Kinetis KV* (document AN5237) for more information.
- *mc_state_machine*—contains the software routines that are executed when the application is in a particular state or state transition. There are separate state machines for reference solution projects (m1_sm_ref_sol) and demo projects (m1_sm_demo).
- *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 nxp.com/KSDK.
- *state_machine*—contains state machine functions for the Fault, Initialization, Stop, and Run states.
- *fm_tsa_pmsm*—TSA table containing all necessary variables for the FreeMASTER control
- *freemaster_cfg*—FreeMASTER configuration file containing setup of the FreeMASTER communication and features.

The */src/projects/* folder contains device-specific files for both the demonstration and reference solution software. 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:

- *board/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.
- *board/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.
- *board/board.c*—contains functions for the UART, GPIO, and SysTick initialization.
- *board/board.h*—contains definitions of board LEDs, buttons, UART instance used for FreeMASTER, and so on.
- *board/clock_config*—contains CPU clock setup functions.
- *board/mcdrv*—this file ensures the abstraction of the *mcdrv_<board>.h* file inclusion.
- *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 PWM period, and ADC channels assigned to phase currents and board voltage.

- *board/pin_mux*—port configuration files. These files are going to be generated by the pin tool in the future.

5.2. IDE workspaces structure

The structure of workspaces (Section 7, "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.

5.3. Application types

The PMSM package includes two projects for each board used ("*build_demos*" and "*build_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 7, "Building and Debugging Application".

6. Tools

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

- Iar Embedded Workbench IDE v7.70 or higher
- Kinetis Design Studio IDE v3.2 or higher
- ARM-MDK - Keil® µVision® version 5.20
- FreeMASTER Run-Time Debugging Tool 2.0

7. Building and Debugging Applications

The package contains projects for Kinetis Design Studio, IAR Embedded Workbench, and µVision Keil IDEs.

7.1. IAR Embedded Workbench IDE

Use IAR Embedded Workbench IDE 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 the 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 open this project. The point 1 in Figure 23 shows the IAR workspace with "*projects_demos_frdm-kv10z*" opened.

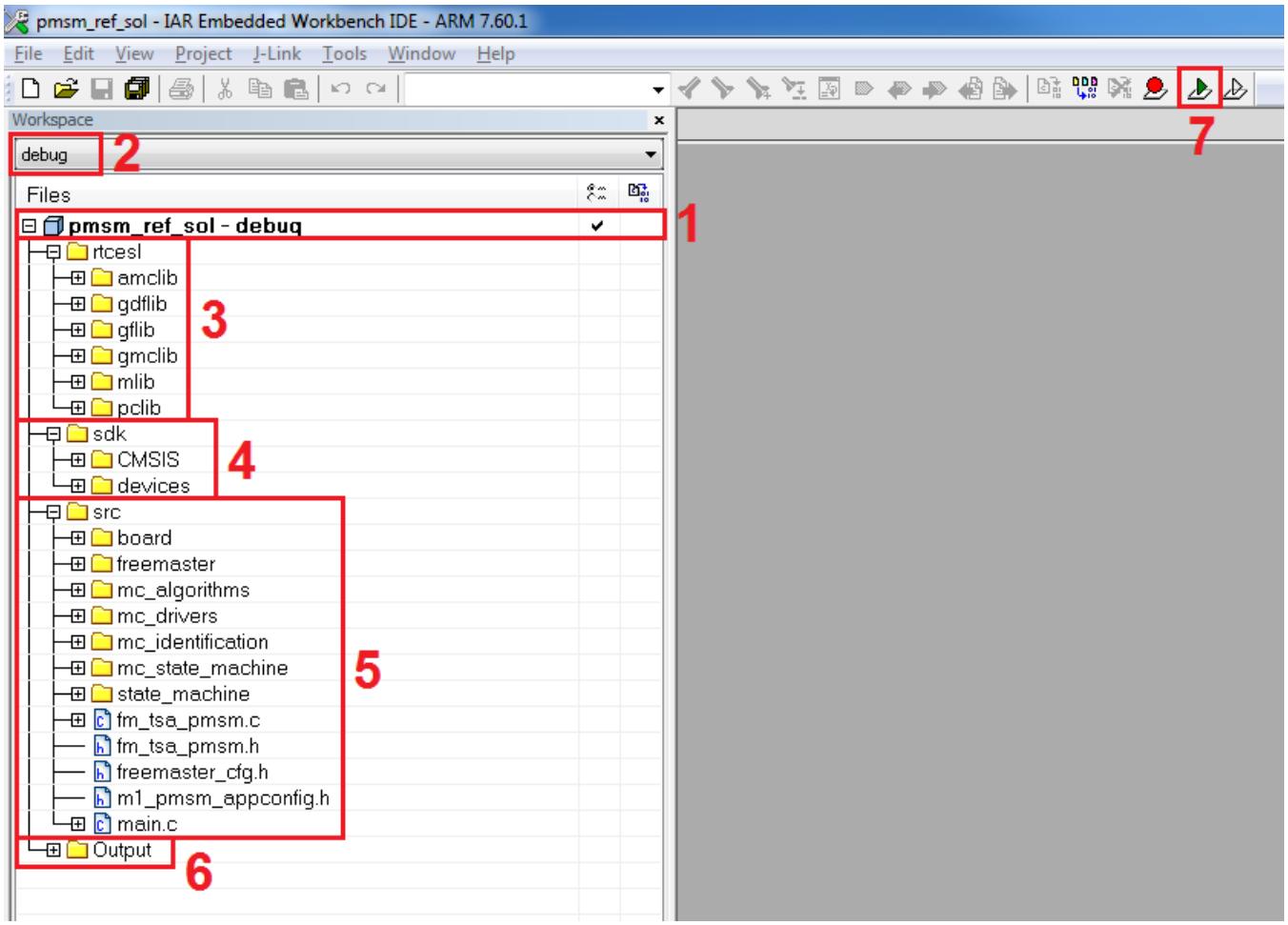

Figure 23. IAR Embedded Workbench

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 23, 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 23 includes these source files and folders:

- Point 3—the RTCESL 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 nxp.com/fslesl.
- Point 4—the */sdk/* folder contains startup routines, 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 5, "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 (P&E Micro – 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". Start the project debugging by clicking Point 7 (Figure 23).

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 OpenSDA driver requires selecting a valid device (see the following figure) during the first code build and loading to the MCU.

Figure 24. IAR select device alert

After you click the "OK" button, the "P&E Connection Manager" window opens (see the following figure). Click the "Select New Device" button and select the particular MCU according to Table 11.

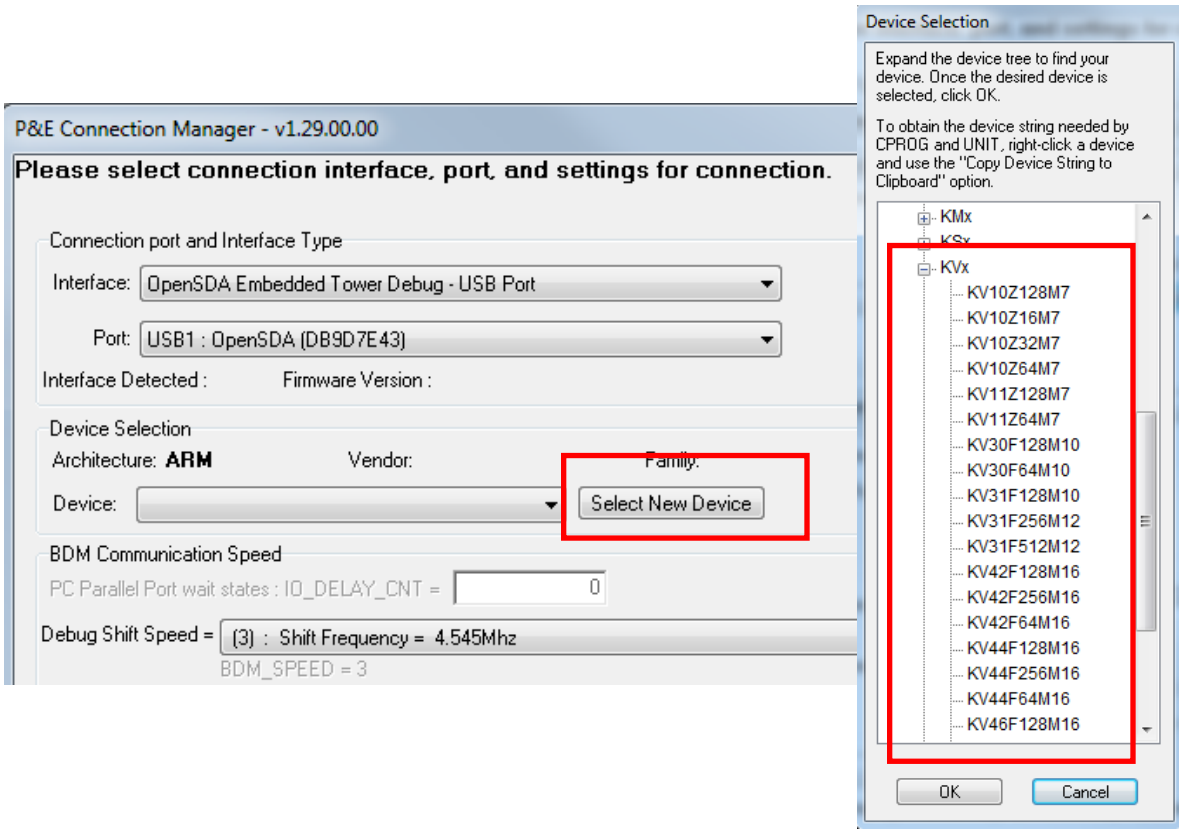

Figure 25. P&E Connection Manager

Table 11. Supported development platforms

		Platform		
		Tower	**Freedom**	**HVP**
MCU	**KV10Z**	KV10Z32M7	KV10Z32M7	KV10Z32M7
	KV31F	KV31F512	KV31F512	KV31F512
	KV11Z	KV11Z128M7	—	—
	KV46F	KV46F256	—	KV46F256
	KV58F	KV58F1M	—	KV58F1M0
	KE15Z	—	KE15Z256M7	—
	KE18F	KE18F512M16	—	KE18F512M16

After you select the right MCU, click the “OK” button and the setting is saved into the cache project files. The next time the code is built, the prompt window does not appear.

7.2. Kinetis Design Studio (KDS) IDE

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 at 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 V3.0.0 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”:

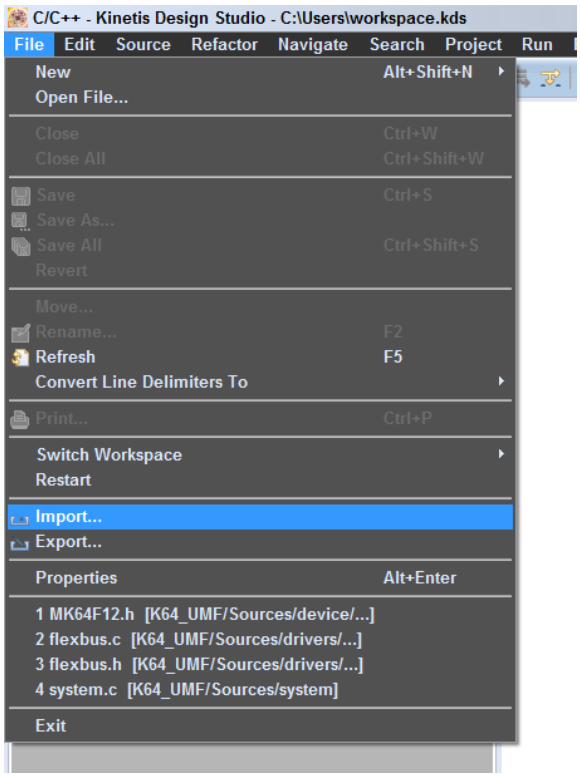

Figure 26. KDS—importing project

- The “Import” window opens. Highlight “Existing Projects into Workspace” in “General” folder, and click the “Next” button:

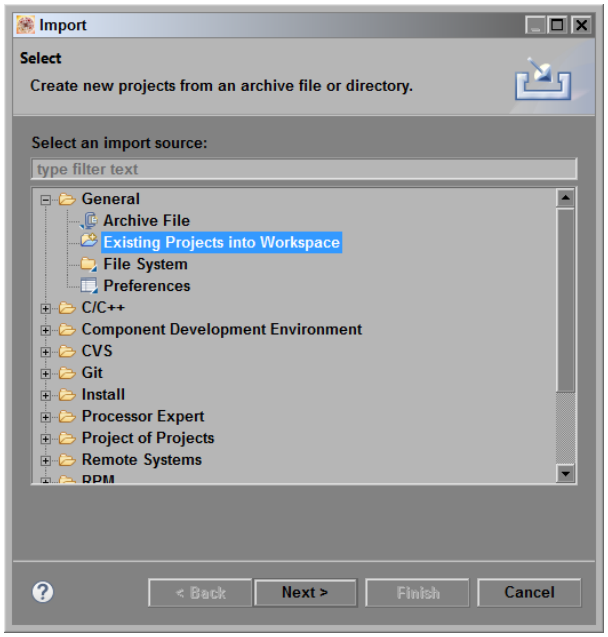

Figure 27. KDS—importing project

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

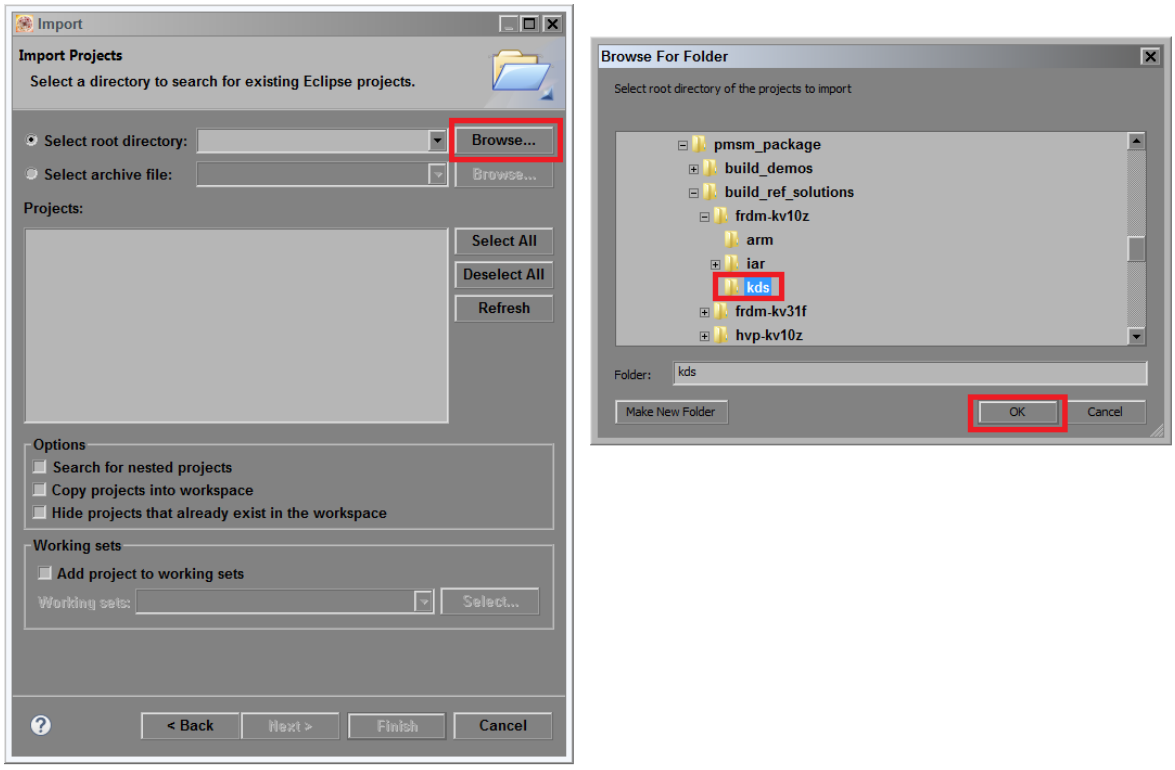

Figure 28. KDS—importing project

- Confirm the project by clicking “Finish”:

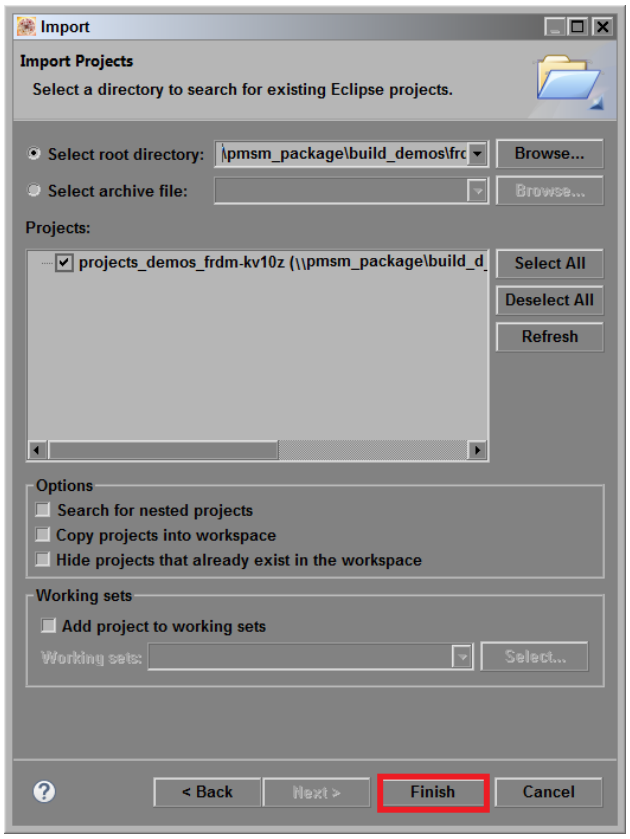

Figure 29. KDS—importing project

The project is now imported to Kinetis Design Studio (Figure 30). 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 smaller flash (for example KV10Z32).

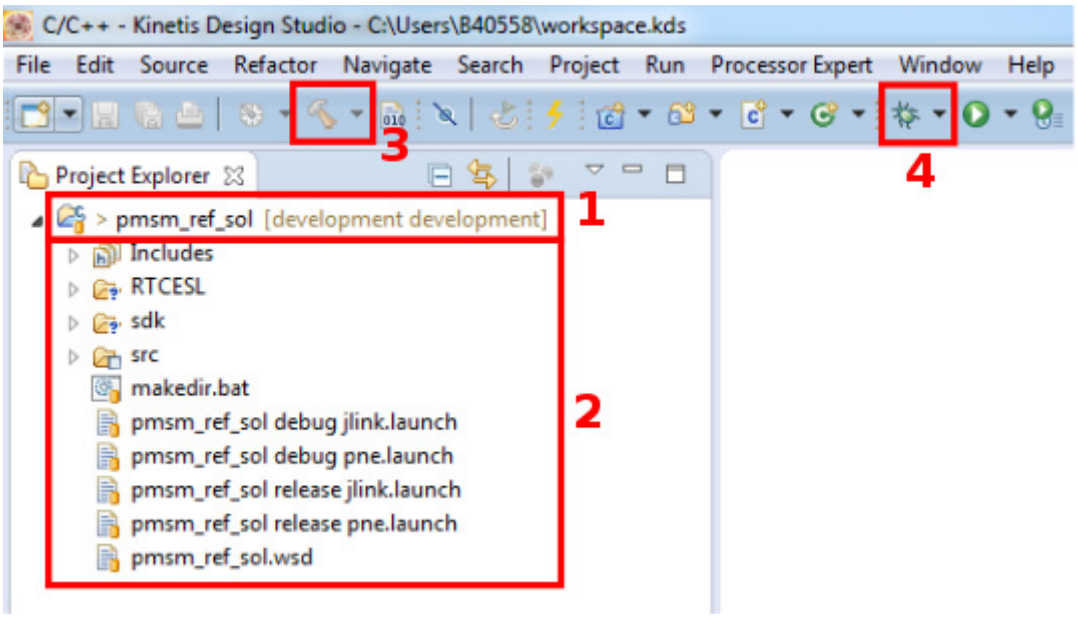

Figure 30. KDS PMSM project

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 30). You can 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).

7.3. ARM-MDK Keil uVision IDE

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 demonstration or solution projects, 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/mdk*:

- Double-click the *pmsm_demo.uvprojx* project file:

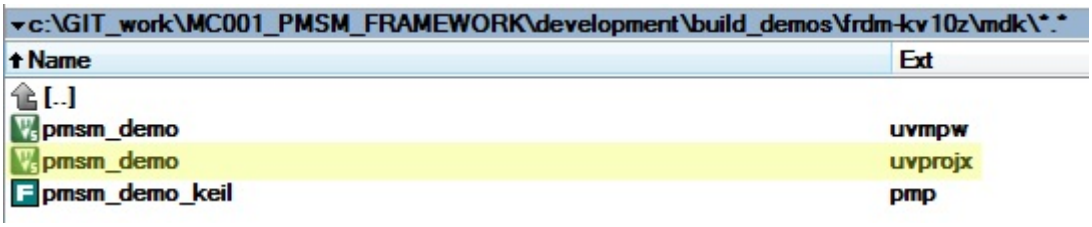

Figure 31. Keil project location

- The project opens. Click the “Build” button (Point 1) to compile the project. Then 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):

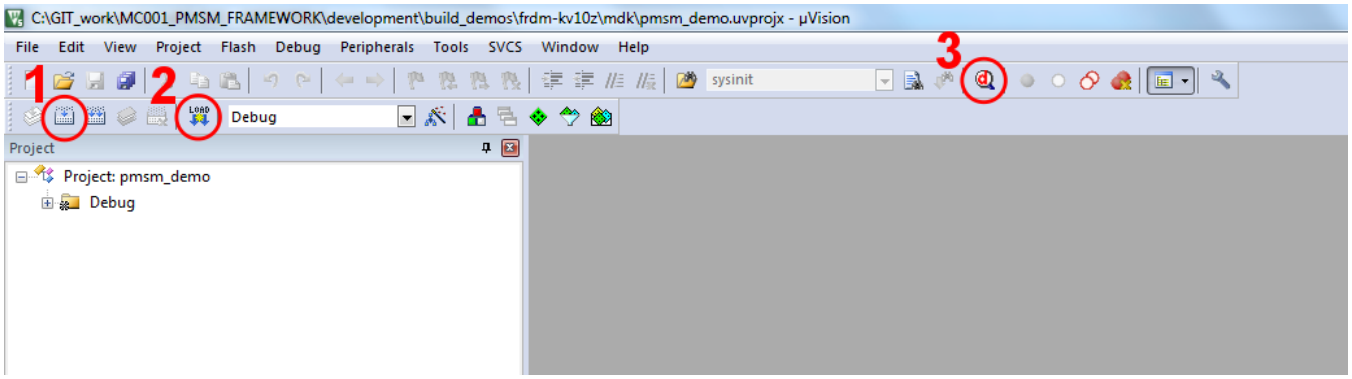

Figure 32. Keil PMSM project

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

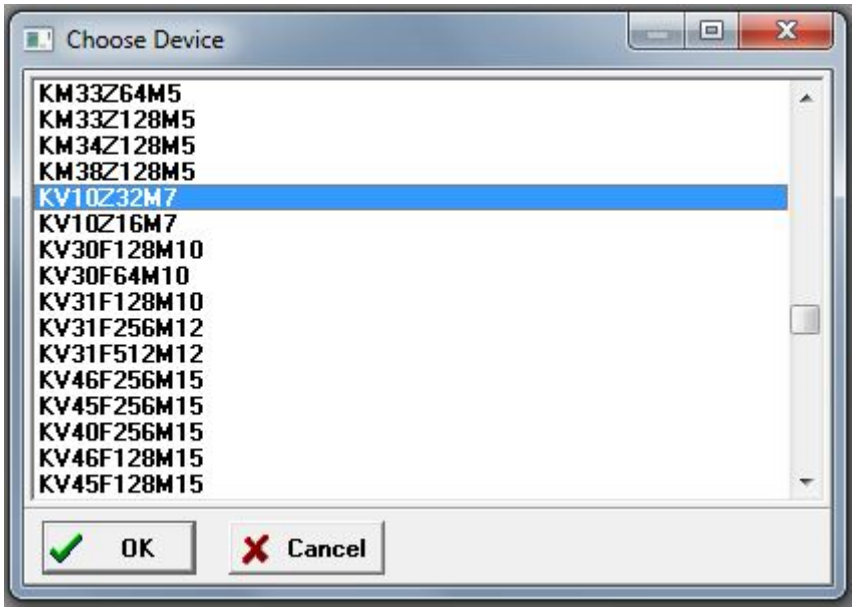

Figure 33. P&E debugger—target selection

- 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).

7.4. OpenSDA debugger

NXP 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 Motorola S-Record or binary 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.

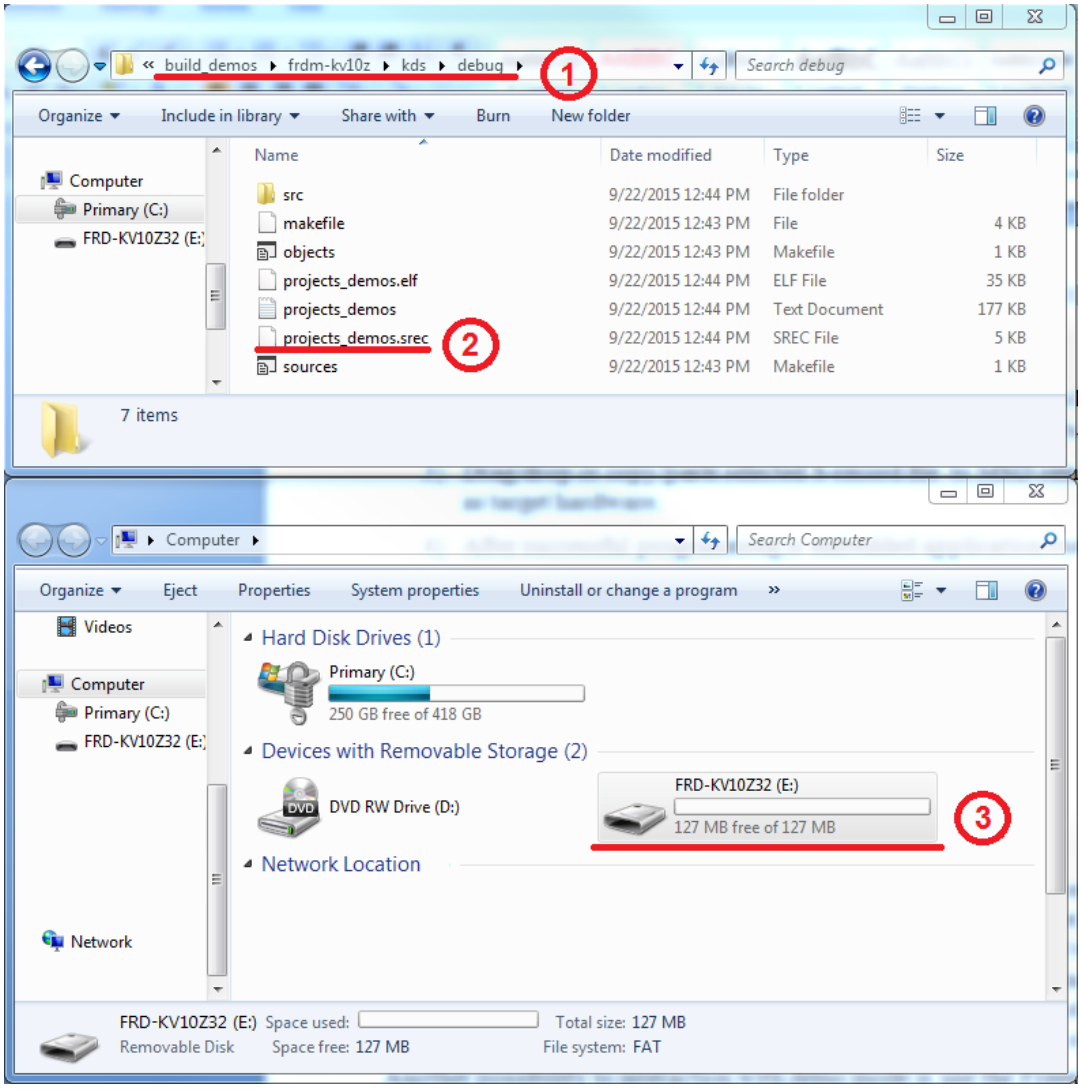

Figure 34. OpenSDA MSD

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

1. Open Windows® OS Explorer.
2. Locate the generated binary file (*.srec*, *.s19*, or *.bin* file extension) (for example *pmsm_package/build_demos/<board_MCU>/<tool>/debug/projects_demos.srec*, as shown in Figure 34, Point 1).
3. Drag/drop or copy/paste the selected binary file (Figure 34, Point 2) to the MSD removable drive with the volume labelled as the target hardware (Figure 34, Point 3).
4. After successful programming, the embedded application executes automatically.
5. Reconnect the target device.

7.5. 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 (in IDEs), some warnings are suppressed. These warnings are mentioned below, and the other warnings are set to a standard configuration.

IAR Embedded Workbench IDE suppresses these warnings:

- Pa082—undefined behavior; the order of volatile accesses is not defined in this statement.

ARM-MDK Keil µVision IDE suppresses these warnings:

- 66—enumeration value is out of "int" range.

By default, there are no other warnings shown during the compiling process.

8. User Interface

The application contains the demo application mode to demonstrate the motor rotation. 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 buttons used are listed in Table 12. 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. The LEDs used are listed in Table 12.

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 serial 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 PMSM.

Control the application using these two interfaces:

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

8.1. 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 12. Control button assignment

Board name	Control button	LED state indication
FRDM-KV10Z	SW2	D4
FRDM-KV31F	SW2	D4
FRDM-KE15Z	SW2	D4
TWR-KV10Z32	SW2	D5
TWR-KV11Z75M	SW2	D5
TWR-KV31F120M	SW2	D5
TWR-KV46F150M	SW2	D5
TWR-KV58F220M	SW2	D5
TWR-KE18F	SW2	D5
HVP-KV10Z	—	D20
HVP-KV31F	—	D20
HVP-KV46F	—	D20
HVP-KV58F	—	D20
HVP-KE18F	—	D20

8.2. 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 nxp.com/freemaster.

Perform these steps to control a PMSM motor using FreeMASTER:

1. Open the FreeMASTER file located in */freemaster/* (*pmsm_ref_sol.pmp* or *pmsm_demo.pmp*) corresponding to your application. All projects except those for KV10 use TSA by default so it is not necessary to select a Symbol file for FreeMASTER (see Section 8.2.1, “FreeMASTER TSA and user variables addition to FreeMASTER watch”).
2. Click the communication button (the red STOP button in the top left-hand corner) to establish the communication.

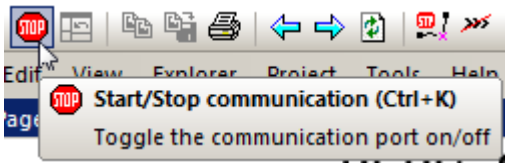

Figure 35. Red STOP button placed in top left-hand corner

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=115200”. Otherwise, the FreeMASTER warning popup window appears.

Figure 36. FreeMASTER—communication is established successfully

3. Control the PMSM motor using the control page.
4. If you rebuild and download a new code to the target, turn the application off and on.

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 115200 bps.

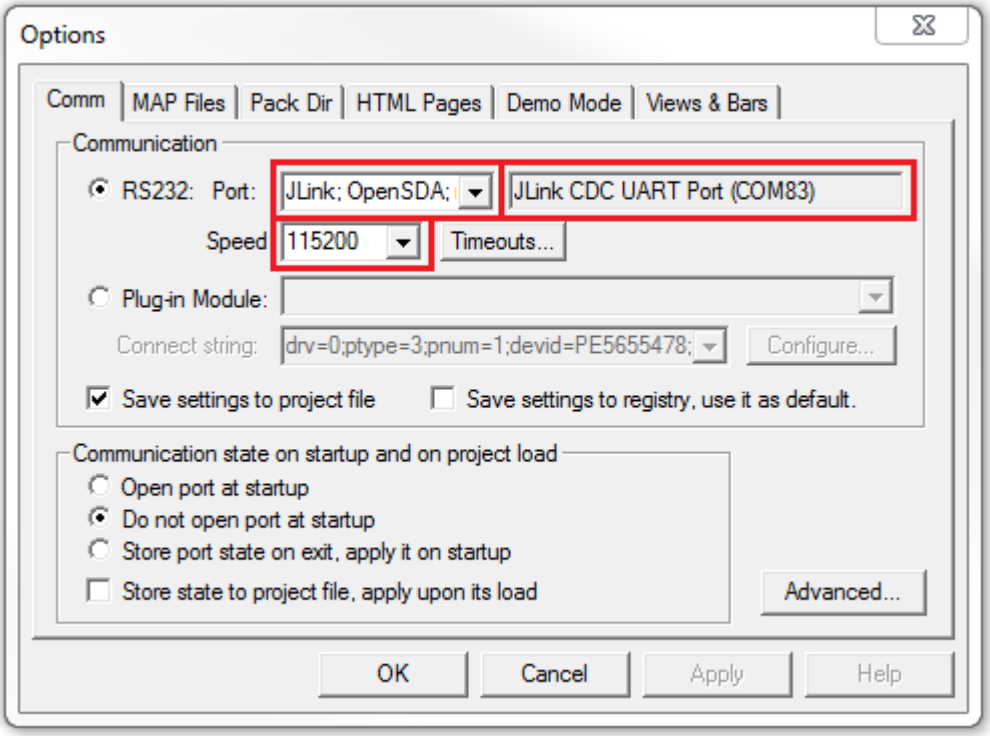

Figure 37. FreeMASTER communication setup window

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.

8.2.1. FreeMASTER TSA and user variables addition to FreeMASTER watch

By default, all projects use the TSA (Target Side Addressing). This means that the information about the variables' address and size are stored in the MCU flash memory. The KV10 projects don't have enough flash memory so the TSA is disabled by default for these projects (to use FreeMASTER, use a symbol file—see below). The TSA feature may be enabled or disabled by the BOARD_FMSTR_USE_TSA macro in the *src/projects/<board>/board.h* file. Only the variables necessary for the MCAT functionality are stored in the TSA. Only these variables are visible in FreeMASTER. If you want to monitor your own variables, you can provide a symbol file which contains the information about the addresses of all variables in the project to FreeMASTER. The symbol files are generated during the build process to the */build_<demo or ref_sol>/<board>/<compiler>/<debug or release>* folder. The symbol files have different extensions for different compilers (IAR generates **.out*, KDS generates **.elf* and Keil generates the **.axf* file). For more information about the TSA, see *FreeMASTER Serial Communication Driver* (document FMSTERSCIDRVUG).

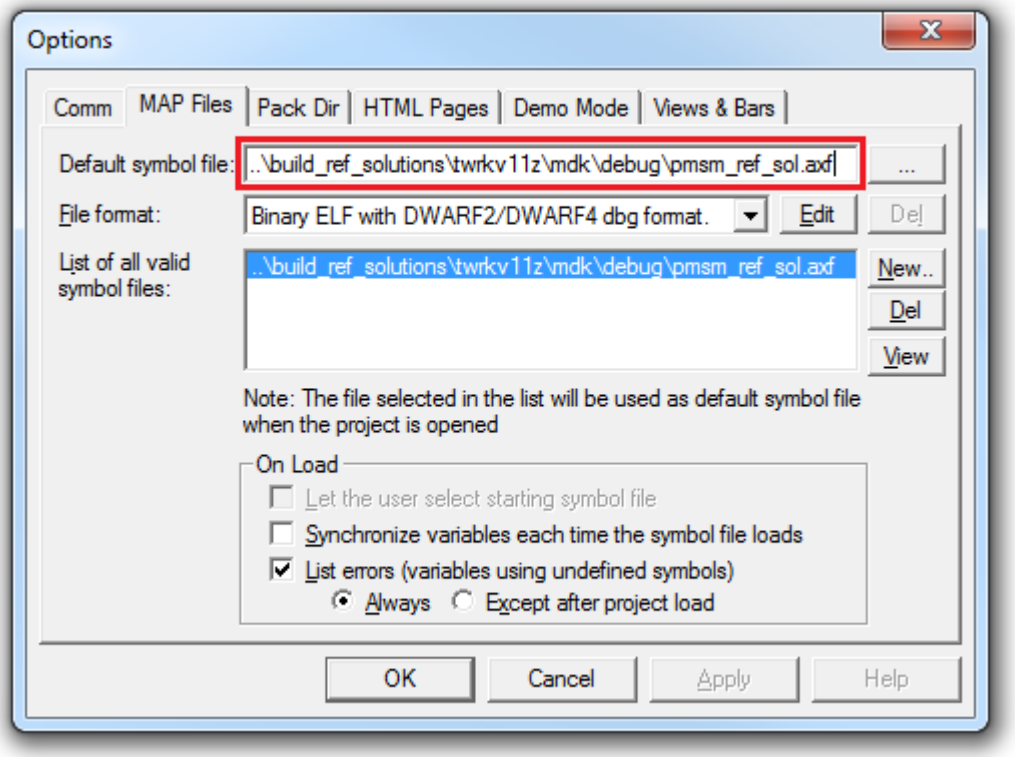

Figure 38. FreeMASTER symbol file selection

8.2.2. 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 motor current—shows the actual torque-producing 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).

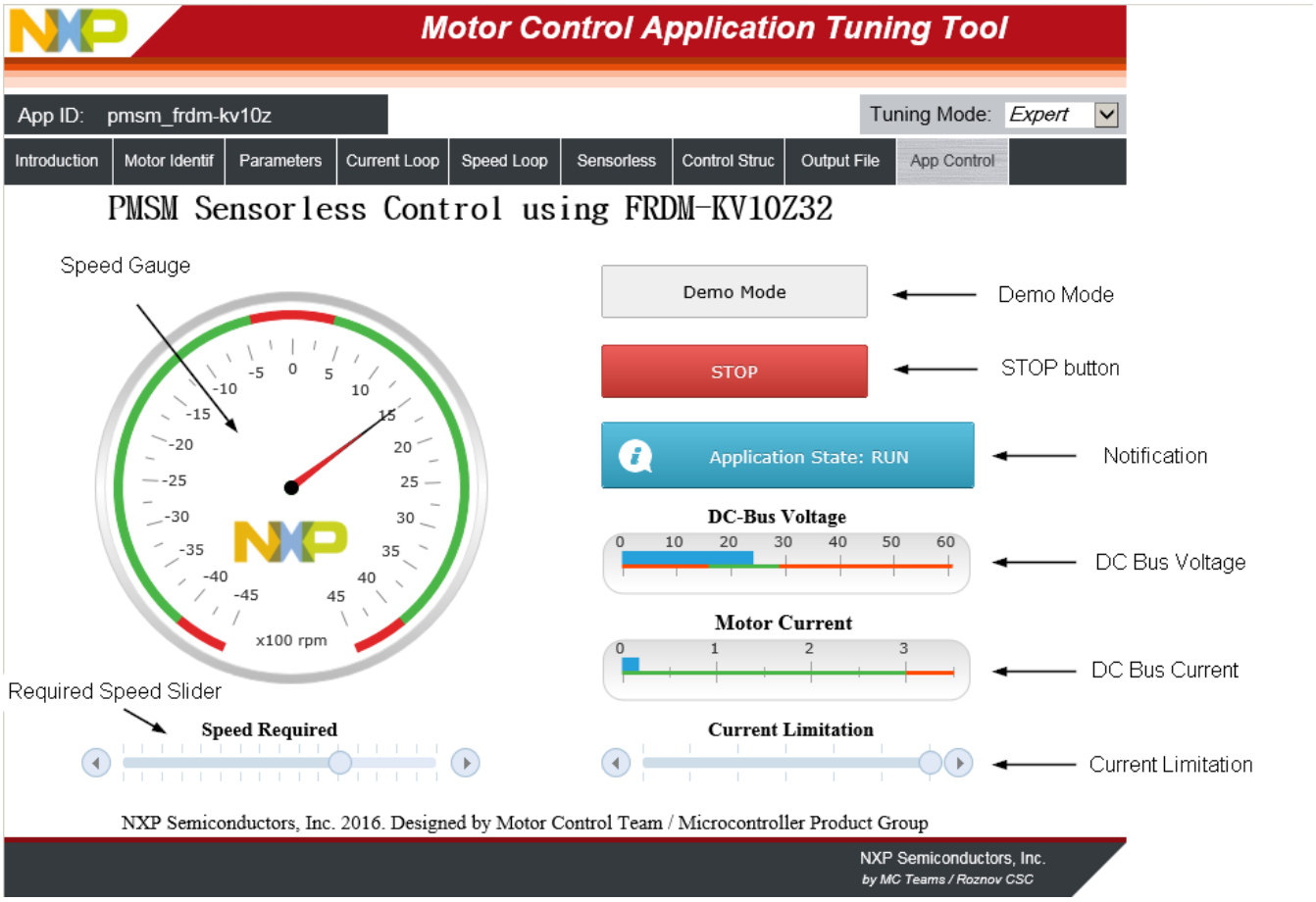

Figure 39. FreeMASTER control page

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.

9. Performing Basic Tasks

9.1. Running the motor

1. Assembly the NXP hardware according to the instructions in Section 4, “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 8.2, “Remote control using FreeMASTER”.
4. Set up the required motor speed using the speed slider, as shown in Figure 39.

9.2. Stopping the motor

1. Click the “STOP” button in the FreeMASTER control page (Figure 39).
2. Set the speed to zero using the speed slider.
3. In case of emergency, turn off the power supply.

9.3. Clearing the fault

To clear the fault, click the fault notification in the control page (Figure 39).

9.4. 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 8.1, “Button control”.

10. Acronyms and Abbreviations

Table 13. 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
TSA	Target Side Addressing

11. References

The following references are available on nxp.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).

12. Revision History

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

Table 14. Revision history

Revision number	Date	Substantive changes
0	10/2015	Initial release.
1	11/2015	Updated subsections to Section 4.3, “Running PMSM application on Tower System”. Updated Table 1. Added Section 4.5.4 “HVP-KV46F150M daughter card”. Updated Figure 22. Updated Figure 23. Updated Table 8.
2	02/2016	Added Section 7.3, “ARM-MDK Keil uVision”. Updated Section 7.1, “IAR Embedded Workbench”.
3	06/2016	Added Section 4.5.5, “HVP-KV58F daughter card”. Updated the references to RTCESL. Bug fixes.
4	09/2016	Added Section 4.3.7., “TWR-KE18F Tower System module” Added Section 4.4.4., “FRDM-KE15Z” Added Section 4.5.6., “HVP-KE18F daughter card”. Added FreeMASTER TSA feature.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: PMSMCRSPUG
Rev. 4
09/2016