NXP Semiconductors
User's Guide

Document Number: PMSMFOCRT10xxUG
Rev. 0, 12/2018

PMSM Field-Oriented Control on MIMXRT10xx EVK

1. Introduction

This user's guide provides a step-by-step guide on how to open, compile, debug, and run Permanent Magnet Synchronous Motor (PMSM) projects in most common IDEs, such as IAR Embedded Workbench®, MCUXpresso, and µVision® Keil® IDEs on MIMRT10xx EVK boards.

This user's guide also describes how to turn the NXP Freedom PMSM power stage and the i.MX RT10xx evaluation kit into a complete motor control reference design (see Section 2, "Hardware setup").

There is a description of how to initialize the FreeMASTER GUI tool for controlling motor-control applications at the end of this document.

Contents

2. Hardware setup

To run the PMSM application using the NXP hardware development platform, you need:

- One of the i.MX evaluation kits (MIMXRT1050-EVK, MIMXRT1060-EVK, or MIMXRT1020-EVK).
- 3-Phase Low-Voltage Freedom PMSM Power Stage (FRDM-MC-LVPMSM).
- PMSM motor—in this case, PMS Motor-Tecnic (Teknic M-2310P motor).

The hardware setup of each component is briefly described in the sections below.

You can order the boards from www.nxp.com (or distributors) and easily build the hardware platform for the target application. The motor used in the presented reference application can be ordered at tgdrives.cz.

2.1. MIMXRT1050-EVK

The MIMXRT1050-EVK board is a platform designed to showcase the most common features of the i.MX RT1050 processor in a small, low-cost package. The MIMXRT1050-EVK board is an entry-level development board, which helps you to become familiar with the processor before investing a large amount or resources in more specific designs. The EVK board provides various memory types, especially the 64-Mbit Quad SPI Flash and the 512-Mbit Hyper Flash.

The i.MX RT1050 processor family (on which the EVK board is based) features NXP's advanced implementation of the Arm® Cortex®-M7 core which operates at speeds of up to 600 MHz. The processor is equipped with 512 KB of on-chip RAM memory. Four Flex Pulse-Width Modulator (eFlexPWM) modules and two 12-bit 16-channel Analog-to-Digital Converters (ADCs) make this device a good choice for high-end multi-motor-control applications.

Table 1. MIMXRT1050-EVK jumper settings

Jumper	Setting	Jumper	Setting	Jumper	Setting
J1	5-6	**J13**	open	**J30**	1-2
J3	1-2	**J15**	open	**J31**	1-2
J4	1-2	**J26**	open	**J32**	1-2
J5	1-2	**J27**	1-2	**J33**	1-2
J7	1-2	**J29**	1-2	**J36**	1-2

NOTE

Jumper J1 in position 5-6 means that that the board is powered from the debug USB. You can also change J1 into position 1-2 for powering from the DC input (5 V).

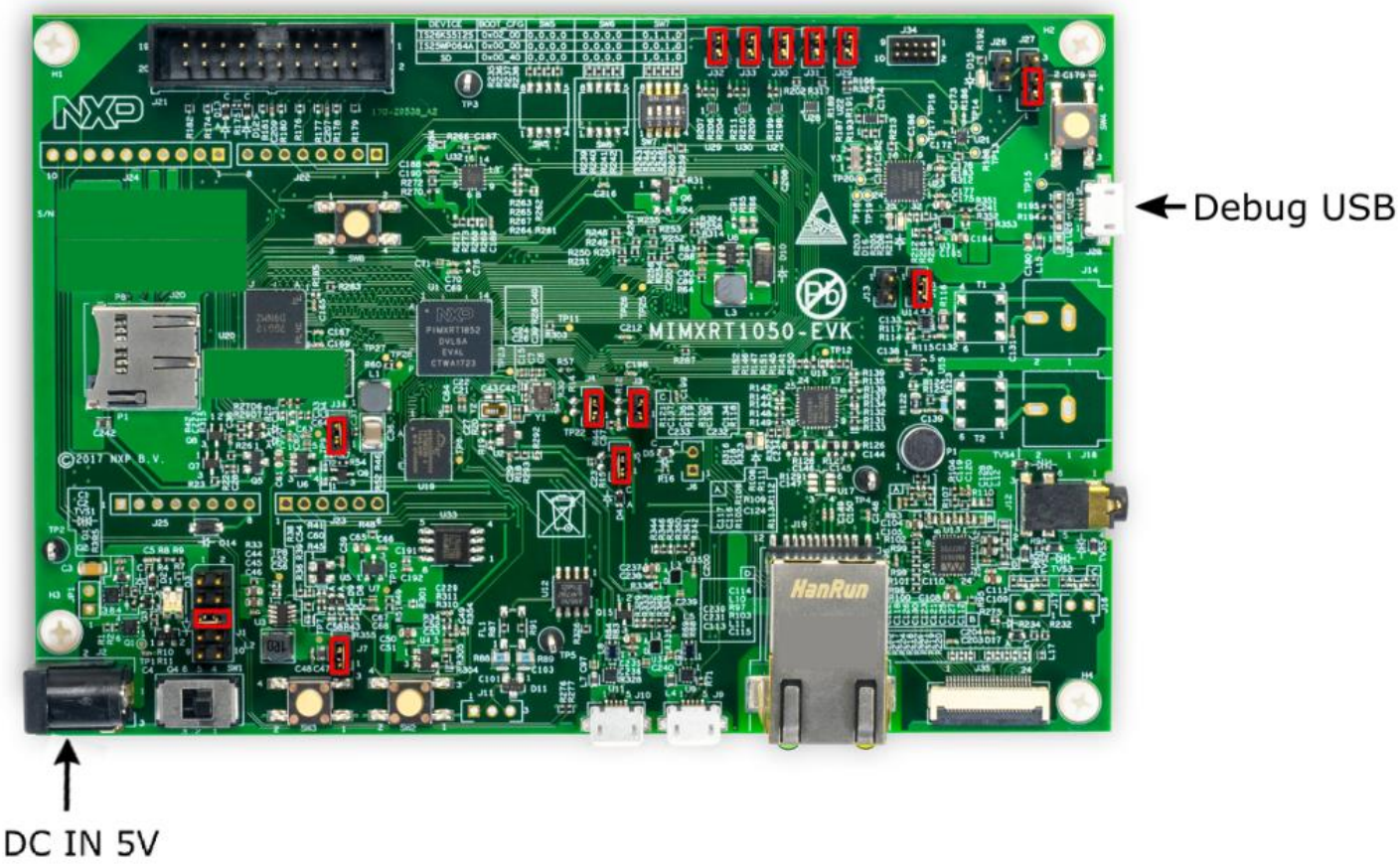

Figure 1. MIMXRT1050-EVK board with highlighted jumper settings

The PWM modules are used in the motor-control application. For this purpose, it is necessary to solder the 0-Ω resistors R278, R279, R280, and R281 to the board. These resistors are located on the underside of the EVK board. For more details, see the schematic (revision A5).

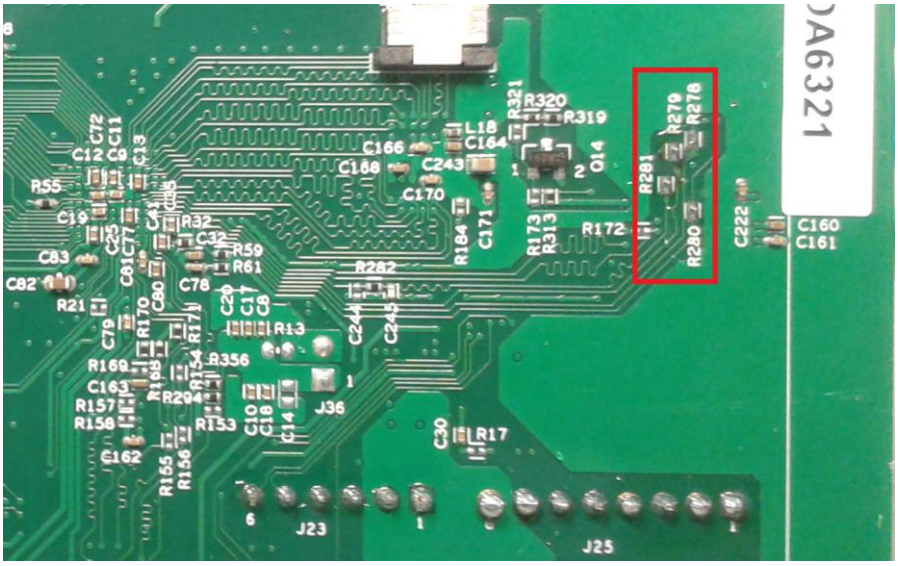

Figure 2. Resistors needed for the PWM on the underside of the EVK board

For more information about the MIMXRT1050-EVK hardware (processor, peripherals, and so on), see the *MIMXRT1050 EVK Board Hardware User's Guide* (document MIMXRT1050EVKHUG).

2.2. MIMXRT1060-EVK

i.MX RT1060 is the latest addition to the industry's first crossover processor series and expands the i.MX RT series to three scalable families. i.MX RT1060 doubles the on-chip SRAM to 1 MB while maintaining pin-to-pin compatibility with i.MX RT1050. This new MCU series introduces additional features ideal for real-time applications, such as high-speed GPIO, CAN-FD, and a synchronous parallel NAND/NOR/PSRAM controller. i.MX RT1060 features the Arm Cortex-M7 core operating at 600 MHz.

Table 2. MIMXRT1060-EVK jumper settings

Jumper	Setting	Jumper	Setting	Jumper	Setting
J1	5-6	J13	open	J44	1-2
J3	1-2	J15	open	J47	1-2
J4	1-2	J26	open	J48	1-2
J5	1-2	J36	1-2	J49	1-2
J7	1-2	J43	1-2	J50	1-2

PWM modules are used in the motor-control application. For this purpose, it is necessary to solder the 0-Ω resistors R278, R279, R280, and R281 to the board. For more details, see the schematic. For more information about the MIMXRT1060-EVK board (processor, peripherals, and so on), see www.nxp.com.

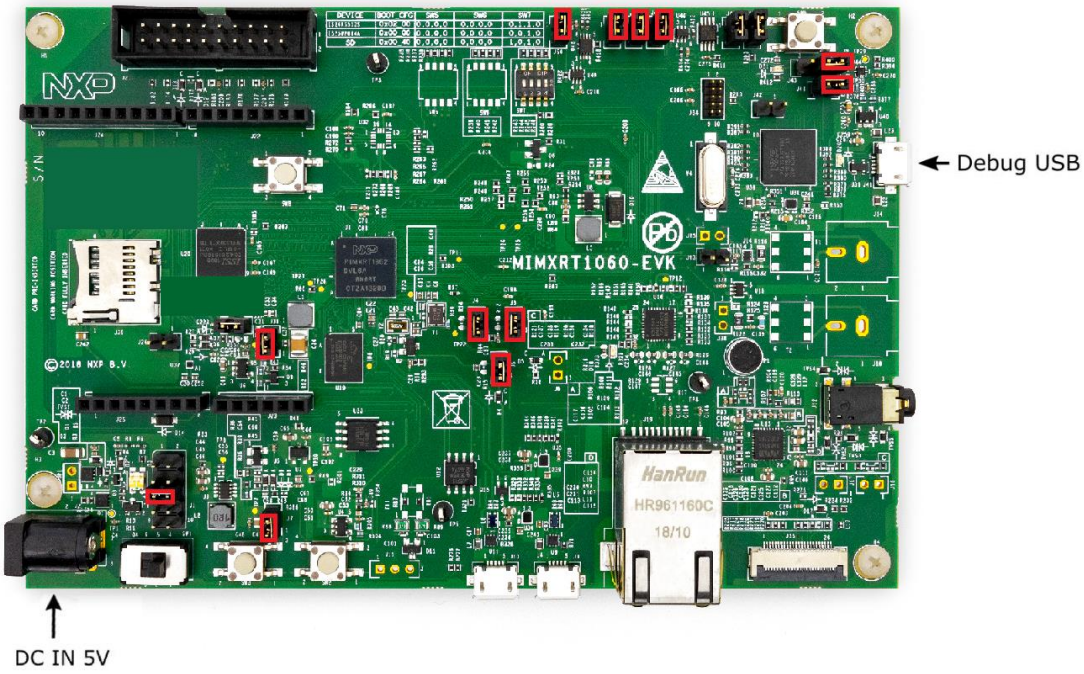

Figure 3. MIMXRT1060-EVK board with highlighted jumper settings

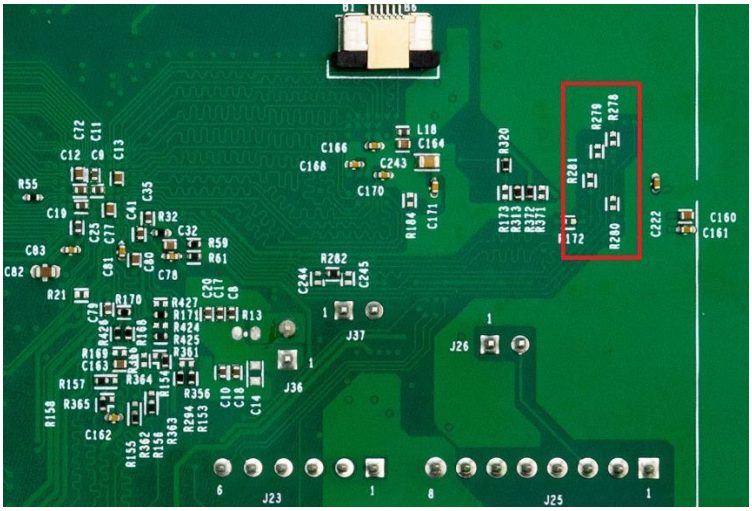

Figure 4. Resistors that must be soldered on the underside of the EVK board

2.3. MIMXRT1020-EVK

The i.MX RT1020 EVK is a 2-layer low-cost through-hole USB-powered PCB. It contains the i.MX RT1020 crossover processor in a LQFP144 package.

The i.MX RT1020 expands the i.MX RT crossover processor families by providing high-performance feature set in low-cost LQFP packages, further simplifying the board design and layout. The i.MX RT1020 features the Arm Cortex-M7 core. This core operates at a of up to 500 MHz to provide high CPU performance and the best real-time response.

Table 3. MIMXRT1020-EVK jumper settings

Jumper	Setting	Jumper	Setting	Jumper	Setting
J1	5-6	J8	1-2	J26	1-2
J3	1-2	J21	open	J27	1-2
J4	1-2	J22	1-2	J28	1-2
J5	1-2	J24	1-2	J29	open
J6	1-2	J25	1-2	J37	1-2

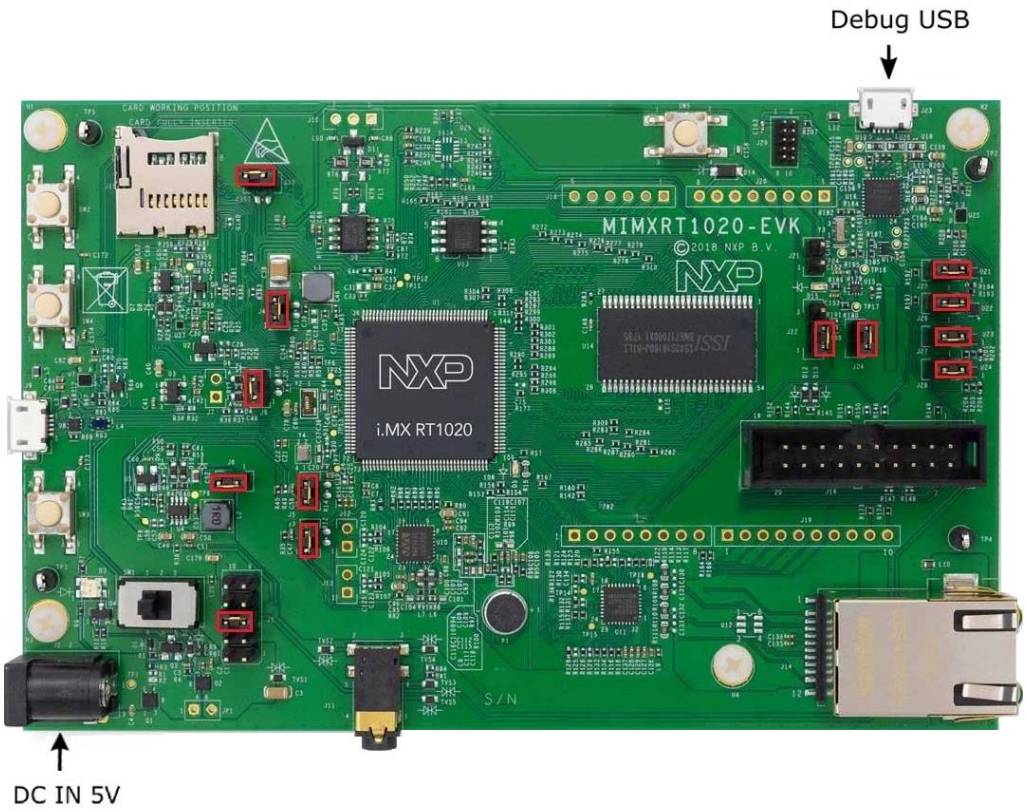

Figure 5. MIMXRT1020-EVK board with highlighted jumper settings

The ADC pins are shared with the next peripherals. Therefore, it is needed to remove resistors R65, R68, R69, and R74 from the board. These resistors are located on the top side of the EVK board. For more details, see the schematic.

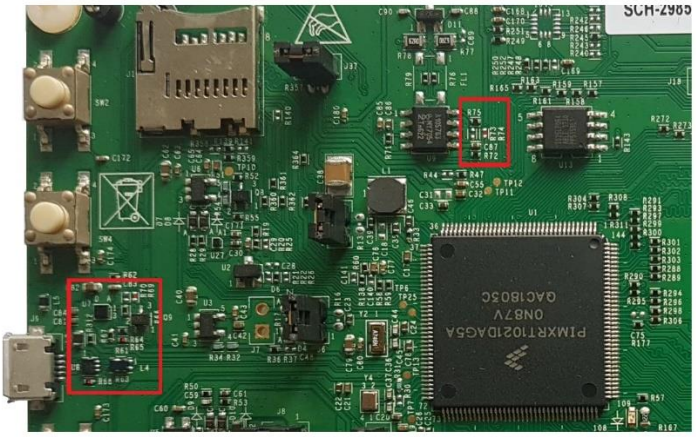

Figure 6. Removed resistor for proper operation of the ADC on the top side of the EVK board

2.4. FRDM-MC-LVPMSM

The FRDM-MC-LVPMSM low-voltage evaluation board (in a shield form factor) turns the MIMXRT10xx-EVK board into a complete motor-control reference design. There is also an interface for the speed/position sensors (Encoder, Hall).

The NXP Freedom PMSM power stage does not require any hardware configuration or jumper settings.

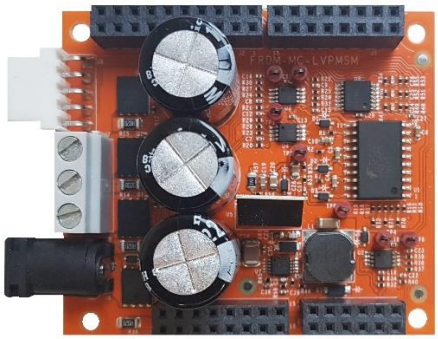

Figure 7. FRDM-MC-LVPMSM

2.5. Teknic M-2310P motor

Teknic M-2310P-LN-04K is a default low-voltage 3-phase synchronous permanent magnet servo motor used in PMSM applications. Motor parameters are summarized in Table 4. A connection diagram of the motor connector is available on the web pages of the motor manufacturer.

Table 4. MIMXRT1060-EVK jumper settings

Characteristic	Symbol	Value	Units
Rated Voltage	Vt	60V	V
Rated speed	-	6000	RPM
Rated torque	T	0.274	Nm
Continuous current	Ics	7.1	A
Number of pole pairs	Pp	4	-

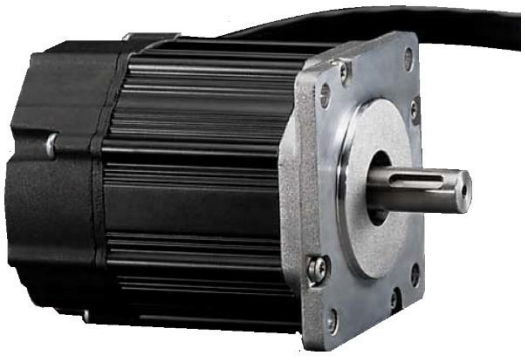

Figure 8. Teknic M-2310P motor

NOTE

For the sensorless control mode, only the power input wires are needed. In the position control mode, connect also the sensor wires to the NXP Freedom power stage. The wiring is described in Section 2.7, "Hardware assembling".

2.6. TG Drives TGT2 motor

The TG Drives TGT2-0032-30-24 (parameters in Table 5) is a second low-voltage 3-phase motor used in the PMSM application. When using this motor, rename the *M1_params_pmsm_evk-imxrt1050_TGDrive.txt* file located in the *freemaster\mcat\param_files* folder to *M1_params_pmsm_evk-imxrt1050.txt*. Rename also the *m1_pmsm_appconfig_TGdrive.h* header file located in *\src\projects\evkbimxrt10..* to *m1_pmsm_appconfig.h*. Then rebuild and download the project to the target device.

Table 5. TG Drives TGT2-0032-30-24/T0PS1KX-1M motor parameters

Characteristic	Symbol	Value	Units
Rated voltage	Vt	30	V
Rated speed	—	3000	RPM
Rated torque	T	0.32	Nm
Continuous current	Ics	5.2	A
Number of pole pairs	Pp	3	—

Figure 9. TG Drives 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 the Hall sensors' signal sensing.

NOTE

For the sensorless control mode, you need only the power input wires. In the position control mode, connect also the sensor wires to the NXP Freedom power stage. The wiring is described in Section 2.7, "Hardware assembling".

2.7. Hardware assembling

1. Connect the NXP Freedom PMSM power stage and the MIMXRT10xx-EVK board together by wires according to the pin assignment and the interconnection diagram.
2. Connect the TG Drives motor 3-phase wires into the screw terminals on the NXP Freedom PMSM power stage.
3. In case of the position motor control, connect also the five wires from the motor sensors to the NXP Freedom PMSM power stage interface. The pins on the connector are:
   1 — +5 V
   2 — GND
   3 — ENC A (Phase A)
   4 — ENC B (Phase B)
   5 — ENC I (Index)
4. Plug the USB cable from the USB host to the OpenSDA micro USB connector on the EVK board.
5. Plug the 24 V DC power supply to the DC power connector on the NXP Freedom PMSM power stage.

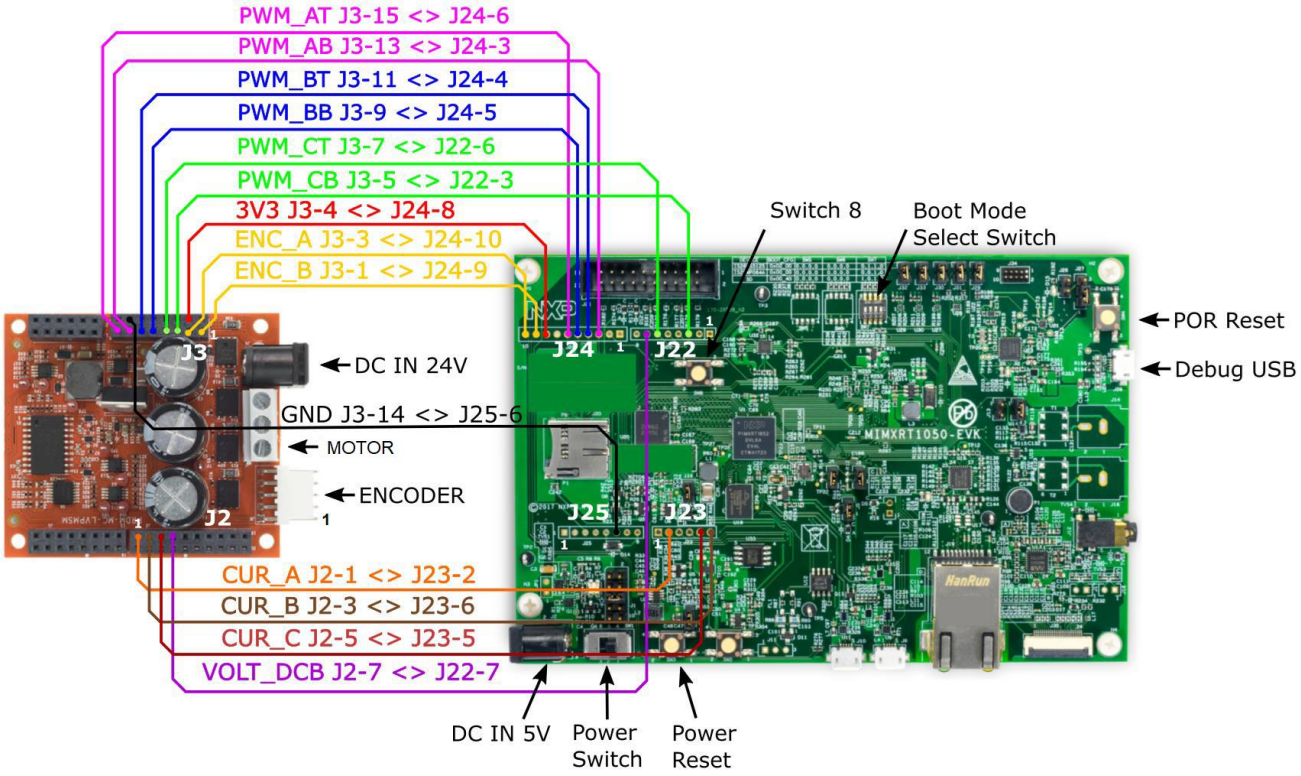

Figure 10. MIMXRT1050-EVK interconnection diagram

Table 6. MIMXRT1050-EVK pin assignment

FRDM-MC-LVPMSM	Connection	MIMXRT1050-EVK	
PWM_AT	**J3, 15 <-> J24, 6**	D13/SPI_CLK	GPIO_SD_B0_00
PWM_AB	**J3, 13 <-> J24, 3**	D10/SPI_CS	GPIO_SD_B0_01
PWM_BT	**J3, 11 <-> J24, 4**	D11/OC2A/PWM/SPI_MOSI	GPIO_SD_B0_02
PWM_BB	**J3, 9 <-> J24, 5**	D12/SPI_MISO	GPIO_SD_B0_03
PWM_CT	**J3, 7 <-> J22, 6**	D5/TI/PWM	GPIO_AD_B0_10
PWM_CB	**J3, 5 <-> J22, 3**	D2/INT0	GPIO_AD_B0_11
3V3	**J3, 4 <-> J24, 8**	3V3	3V3
ENC_A	**J3, 3 <-> J24, 10**	D15/I2C_SCL	GPIO_AD_B0_00
ENC_B	**J3, 1 <-> J24, 9**	D14/I2C_SDA	GPIO_AD_B0_01
GND	**J3, 14 <-> J25, 6**	GND	GND
CUR_A	**J2, 1 <-> J23, 2**	A1/ADC1	GPIO_AD_B1_11
CUR_B	**J2, 3 <-> J23, 6**	A5/ADC5/SCL	GPIO_AD_B1_00
CUR_C	**J2, 5 <-> J23, 5**	A4/ADC4/SDA	GPIO_AD_B1_01
VOLT_DCB	**J2, 7 <-> J22, 7**	D6/AIN0/PWM/OC0A	GPIO_AD_B1_02

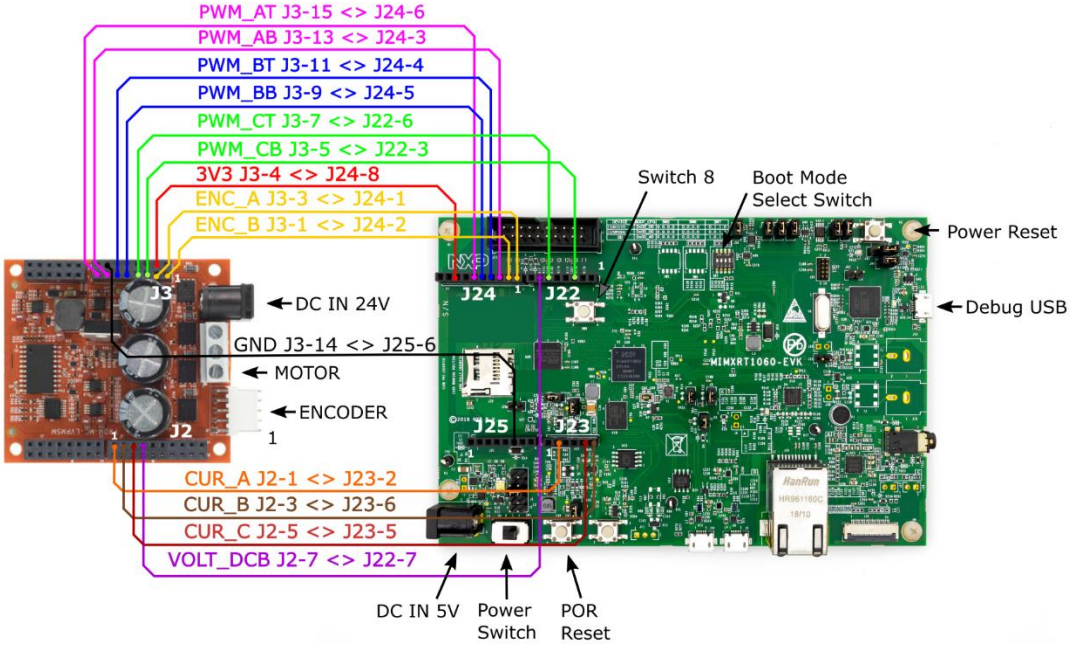

Figure 11. MIMXRT1060-EVK interconnection diagram

Table 7. MIMXRT1060-EVK pin assignment

FRDM-MC-LVPMSM	Connection	MIMXRT1060-EVK	
PWM_AT	**J3, 15 <-> J24, 6**	D13/SPI_CLK	GPIO_SD_B0_00
PWM_AB	**J3, 13 <-> J24, 3**	D10/SPI_CS	GPIO_SD_B0_01
PWM_BT	**J3, 11 <-> J24, 4**	D11/OC2A/PWM/SPI_MOSI	GPIO_SD_B0_02
PWM_BB	**J3, 9 <-> J24, 5**	D12/SPI_MISO	GPIO_SD_B0_03
PWM_CT	**J3, 7 <-> J22, 6**	D5/TI/PWM	GPIO_AD_B0_10
PWM_CB	**J3, 5 <-> J22, 3**	D2/INT0	GPIO_AD_B0_11
3V3	**J3, 4 <-> J24, 8**	3V3	3V3
ENC_A	**J3, 3 <-> J24, 1**	D15/I2C_SCL	GPIO_AD_B1_00
ENC_B	**J3, 1 <-> J24, 2**	D14/I2C_SDA	GPIO_AD_B1_01
GND	**J3, 14 <-> J25, 6**	GND	GND
CUR_A	**J2, 1 <-> J23, 2**	A1/ADC1	GPIO_AD_B1_11
CUR_B	**J2, 3 <-> J23, 6**	A5/ADC5/SCL	GPIO_AD_B1_00
CUR_C	**J2, 5 <-> J23, 5**	A4/ADC4/SDA	GPIO_AD_B1_01
VOLT_DCB	**J2, 7 <-> J22, 7**	D6/AIN0/PWM/OC0A	GPIO_AD_B1_02

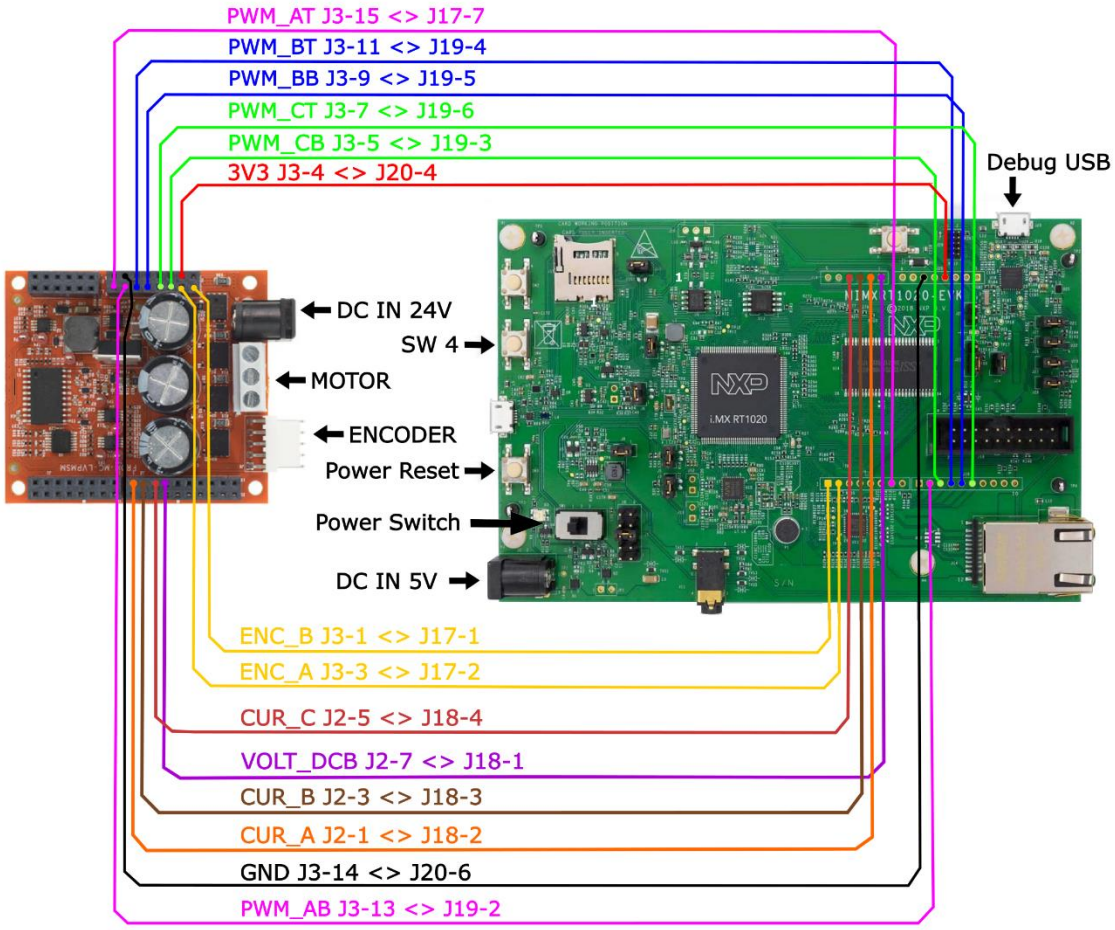

Figure 12. MIMXRT1020-EVK interconnection diagram

Table 8. MIMXRT1020-EVK pin assignment

FRDM-MC-LVPMSM	Connection	MIMXRT1020-EVK	
PWM_AT	**J3, 15 <-> J17, 7**	D6/AIN0/PWM/OC0A	GPIO_AD_B0_14
PWM_AB	**J3, 13 <-> J19, 2**	D9/OC1A/PWM	GPIO_AD_B0_15
PWM_BT	**J3, 11 <-> J19, 4**	D11/OC2A/PWM/SPI_MOSI	GPIO_AD_B0_12
PWM_BB	**J3, 9 <-> J19, 5**	D12/SPI_MISO	GPIO_AD_B0_13
PWM_CT	**J3, 7 <-> J19, 6**	D13/SPI_CLK	GPIO_AD_B0_10
PWM_CB	**J3, 5 <-> J19, 3**	D10/SPI_CS	GPIO_AD_B0_11
3V3	**J3, 4 <-> J20, 4**	3V3	3V3
ENC_A	**J3, 3 <-> J17, 1**	D0/UART_RX	GPIO_AD_B1_09
ENC_B	**J3, 1 <-> J17, 2**	D1/UART_TX	GPIO_AD_B1_08
GND	**3, 14 <-> J20, 6**	GND	GND
CUR_A	**J2, 1 <-> J18, 2**	A1/ADC1	GPIO_AD_B1_11
CUR_B	**J2, 3 <-> J18, 3**	A0/ADC0	GPIO_AD_B1_12
CUR_C	**J2, 5 <-> J18, 4**	A3/ADC3	GPIO_AD_B1_13
VOLT_DCB	**J2, 7 <-> J18, 1**	A4/ADC4/SDA	GPIO_AD_B1_10

3. Motor-control drivers vs. MCUXpresso SDK

The motor-control examples use the MCUXpreso SDK peripheral drivers to configure the general peripherals, such as the clocks, SPI, SIM, and ports. However, 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).

It is highly recommended not to modify the default configuration of the allocated MC peripherals due to a possible application timing conflict. The *mcdrv_evkbimxrt10xx.c* source file contains the configuration functions of the allocated peripherals.

For more information about the peripherals' configuration in the motor-control application on the MIMXRT10xx-EVK board, see *PMSM Field-Oriented Control on MIMXRT10xx EVK* (document AN12214).

4. Project file structure

All the 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 organized in a logical manner. The folder structure used in the IDE is different from the structure of the PMSM package installation and described in separate sections.

4.1. PMSM project structure

The directory tree of the PMSM project is shown in Figure 13.

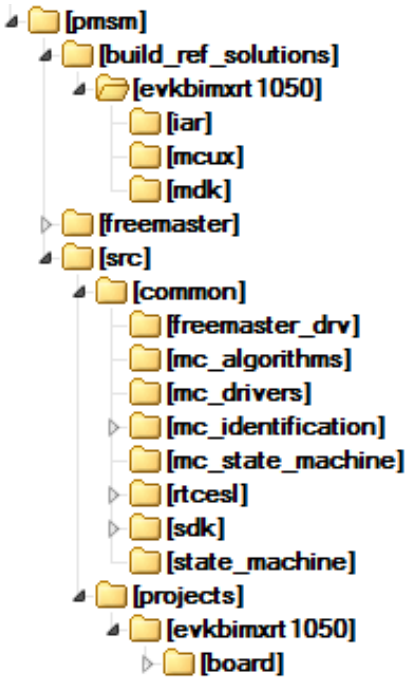

Figure 13. Directory tree

The PMSM package consists of these folders:

- *build_ref_solutions.*
- *freemaster.*
- *src.*

The main project folder */build_ref_solutions/build_single/evkbimxrt10xx/* contains these folders:

- *iar*—for the IAR Embedded Workbench IDE.
- *mcux*—for the MCUXpresso IDE.
- *mdk*—for the μVision Keil IDE.

Each folder contains IDE-related configuration files for the PMSM project—project files, output executable, linker files, and so on. See the step-by-step tutorial on how to open, run, and debug the PMSM project in Section 6, “Building and debugging the application”.

The */freemaster/* folder contains the auxiliary files for the MCAT tool. This folder contains also the FreeMASTER project file *pmsm_ref_sol_single.pmp*. Open this file in the FreeMASTER tool and use it to control the application. This PMSM reference project contains also the motor-identification algorithms and the scalar-control algorithm which can be used for motor control tuning. See *PMSM Field-Oriented Control on MIMXRT10xx EVK* (document AN12214) for more information about the Motor Control Application Tuning (MCAT) 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 other motor-control projects.
- *projects*—contains the MCU-dependent source code.

The */src/common/* folder contains these subfolders common to the other motor-control projects:

- *freemaster_drv*—contains the FreeMASTER header and source files.
- *rtcesl*—contains the 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 (each for a different IDE) that contain precompiled library files for the Arm Cortex-CM7 cores. The *rtcesl* folder is taken from the RTCESL release 4.5, and fully compatible with the official release. See www.nxp.com/rtcesl 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 the automated parameter-identification routines of the motor. See the "Tuning and controlling the application" section of *PMSM Field-Oriented Control on MIMXRT10xx EVK* (document AN12214) for more information.
- *mc_state_machine*—contains the software routines that are executed when the application is in a particular state or state transition.
- *sdk*—contains the functions for the startup routines, header files, core specific functions, and linker files used by the IDEs available in this package and the routines for the core clock settings.
- *state_machine*—contains the state machine functions for the FAULT, INITIALIZATION, STOP, and RUN states.
- *freemaster_cfg.h*—the FreeMASTER configuration file containing the FreeMASTER communication and features setup.

The */src/projects/* folder contains the reference solution software files. They specify the peripheral initialization routines, FreeMASTER initialization, motor definitions, and state machines. The source code contains a lot of comments. The functions of the files are explained in this list:

- *board/m1_pmsm_appconfig.h*—contains the 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 (MCAT) tool, the tool generates this file at the end of the tuning process. The "*m1_*" (meaning motor 1) process is used in single-motor applications. This number designates the motor number in multi-motor applications.
- *board/main.c*—contains the 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 the functions for the UART, GPIO, and SysTick initialization.
- *board/board.h*—contains the definitions of the board LEDs, buttons, UART instance used for FreeMASTER, and so on.
- *board/clock_config*—contains the CPU clock setup functions. These files are going to be generated by the clock tool in the future.

- *board/mcdrv.h*—this file ensures the abstraction of the *mcdrv_evkbimxrt10xx.h* file inclusion.
- *board/mcdrv_evkbimxrt10xx.c*—contains the motor-control driver peripherals initialization functions that are specific for the board and MCU used.
- *board/mcdrv_evkbimxrt10xx.h*—header file for *board/mcdrv_evkbimxrt10xx.c*. This file contains the macros for changing the PWM period and the ADC channels assigned to the phase currents and board voltage.
- *board/pin_mux*—port configuration files. It is recommended to generate these files in the pin tool.

4.2. IDE workspaces structure

The structure of workspaces 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.

The “build_ref_solutions“ project includes all the 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.

5. Tools

Install the FreeMASTER Run-Time Debugging Tool 2.0 and one of the following IDEs on your PC to run and control the PMSM application properly:

- IAR Embedded Workbench IDE v8.30.1 or higher.
- MCUXpresso v10.2.1.
- ARM-MDK - Keil µVision version 5.25.2.
- FreeMASTER Run-Time Debugging Tool 2.0.

6. Building and debugging the application

The package contains the PMSM reference project for the IAR Embedded Workbench, MCUXpresso, and µVision Keil IDEs. The following sections describe how to open, run (Debug and Release configuration), and debug the project in a particular IDE.

The Debug and Release configurations differ in the level of optimizations and the memory location, from which the application is executed (RAM or FLASH). That is the reason why both configurations are described separately in each IDE’s subsection.

If using the release FLASH configuration, make sure that switch SW7 is set to the Hyper Flash boot configuration (SW7-1 OFF, SW7-2 ON, SW7-3 ON, SW7-4 OFF). For more information about this setting, see *How to Enable Debugging for FLEXSPI NOR Flash* (document AN12183), Chapter 3, “XIP boot flow”.

NOTE

The CMSIS-DAP is used as a debugger in the PMSM project. To install the required drivers, go to www.nxp.com/opensda, choose the "MIMXRT10xx-EVK" board and follow the instructions.

6.1. IAR Embedded Workbench IDE

The first step is to open the project file. It is in the *build_ref_solutions/evkbimxrt10xx/iar* folder, which contains all necessary files. Double-click "*pmsm_ref_sol.eww*" to open this project. The project opened in the IAR Embedded Workbench IDE is fully configured and includes all necessary source and header files required by the application, such as startup code, clock configuration, and peripherals configuration.

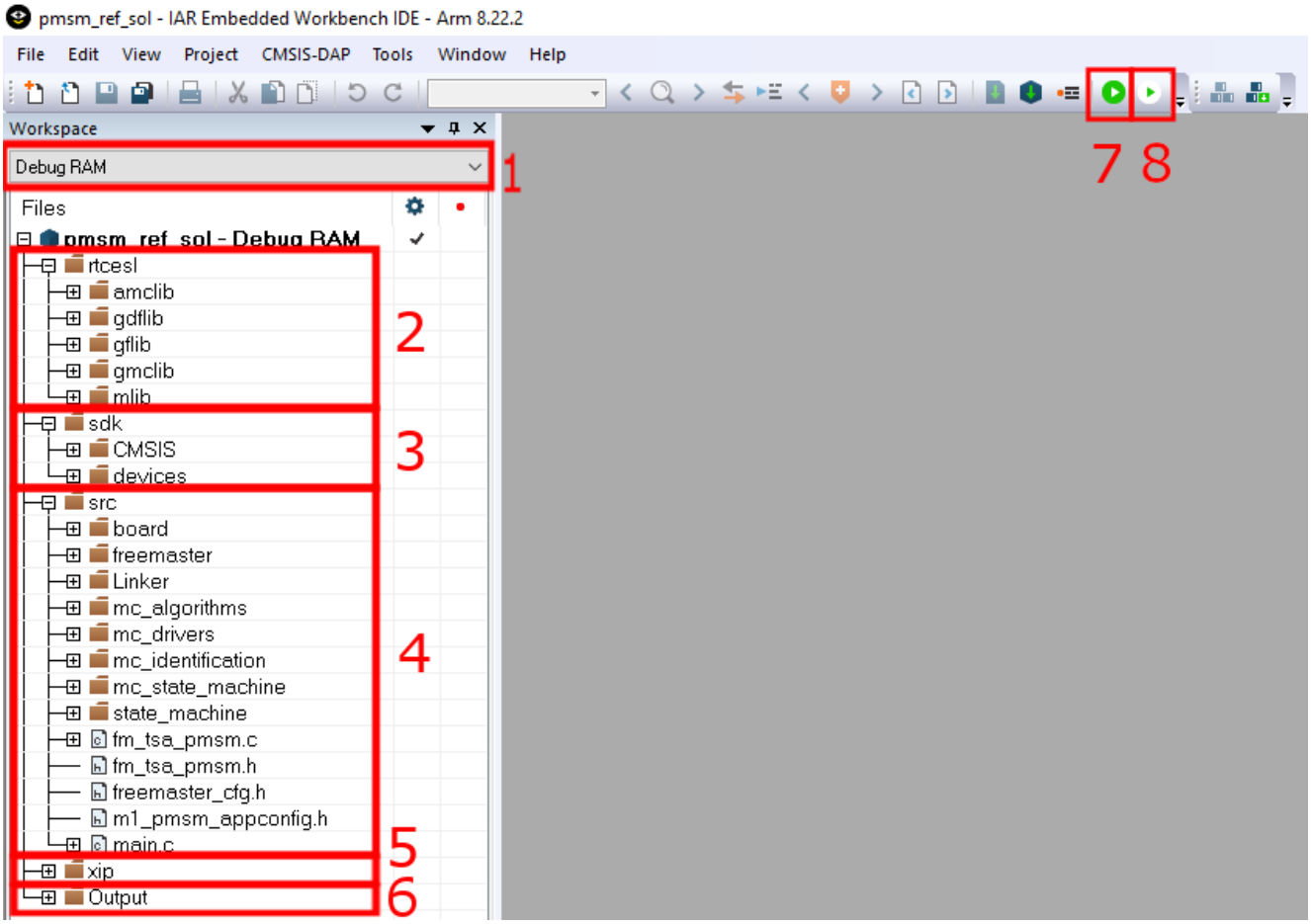

Figure 14. IAR IDE workspace

- Point 1 in Figure 14 shows the current build configuration. You can choose between the Debug RAM and Release FLASH configurations. Each of the two conditions has its own settings (described below).
- Point 2—the */rtcesl/* library source folder contains the header files for the mathematical and control functions used in this project. See www.nxp.com/rtcesl for more information about RTCESL.

- Point 3—the */sdk/* folder contains the startup routines, system initialization, clock definition, and header file for the MCU. It also contains the basic CMSIS routines for interrupt handling.
- Point 4—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 organized to fit into the IDE workspace.
- Point 5—the */xip/* folder contains files for the XIP (eXecution In Place) support. See *How to Enable Debugging for FLEXSPI NOR Flash* (document AN12183) for more information.
- Point 6—shows the output file generated by the compiler and ready to be used by the debugger.
- Point 7—"Download and Debug" button. This button downloads the compiled project into the MIMXRT10xx-EVK's RAM/FLASH.
- Point 8—"Debug without downloading" button (useful for debugging the Release FLASH configuration). This button connects to the running target application. The second step is to select the build configuration.

6.1.1. Debug RAM

This configuration links the application to the RAM memory. The optimization is disabled. To build, run, and debug this configuration, follow these instructions:

1. Select the "Debug RAM" build configuration (Point 1 in Figure 14).
2. Choose CMSIS-DAP as the debugger – go to "Project->Options->Debugger" and click the "Setup" tab. Choose CMSIS-DAP as the driver. Leave the other settings in the "Debugger" category as they are. Then click "CMSIS DAP" (in the left panel under "Debugger"), choose the "Interface" tab, and select "SWD" in the interface settings.
3. Press the "F7" key or click "Project->Make" to build the project.
4. After a successful build, the application is ready to be downloaded to the RAM. To perform this action, just press "CTRL+D" or click the "Download and Debug" button (Point 7 in Figure 14).
5. As shown in Figure 15, the layout is switched to the debugging mode.
    - To run the application, press the "Go" button (Point 1 in Figure 15).
    - To pause the running application, press the "Pause" button (Point 2 in Figure 15).
    - To stop debugging, press the "Stop" button (Point 3 in Figure 15).

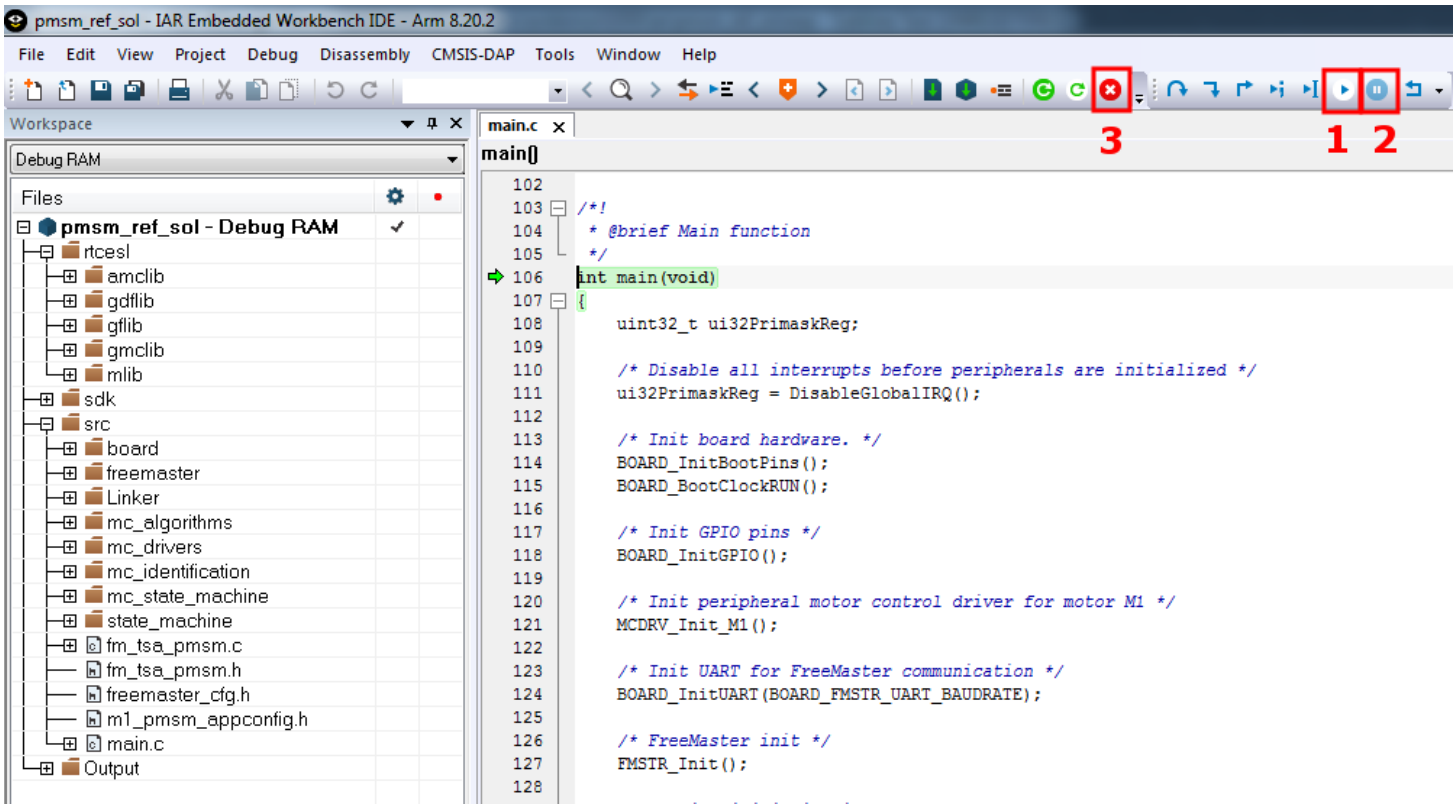

Figure 15. IAR debug workspace

6.1.2. Release FLASH

This configuration links the application to the hyper flash memory. The optimization is set to the highest level for speed. The significant difference (opposite to the Debug configuration) is the ability to choose the functions which are going to be executed from the RAM. This feature is achieved by defining one of the following appropriate symbols in “Preprocessor” (“Project->Options->C/C++ Compiler->Preprocessor tab, Defined symbols” box):

- RAM_OPTIM_LOW—the Real Time Control Embedded Software Libraries (RTCESL www.nxp.com/rtcesl) are going to be executed from the RAM.
- RAM_OPTIM_MEDIUM—the motor-control fast-loop algorithms are going to be executed from the RAM. These algorithms are executed periodically in the ADC conversion complete interrupt. This interrupt has the highest priority level. When the interrupt is entered, the DC-bus voltage and motor phase currents are read and the PMSM FOC algorithm is calculated. The RAM_OPTIM_MEDIUM also includes the RAM_OPTIM_LOW symbol.
- RAM_OPTIM_HIGH—the slow-loop interrupt service routine (basically speed control, user button, and so on) is going to be executed from the RAM. The RAM_OPTIM_HIGH includes both the above-mentioned symbols.

You can also choose whether the Interrupt Vector Table (IVT) is going to be placed into the RAM or Hyper Flash. To place the IVT into the RAM, define the following symbol in “Preprocessor” and “Linker” (“Project->Options->Linker->Configuration file symbol definitions”):

- ENABLE_RAM_VECTOR_TABLE=1—the linker reserves an empty space from address 0x0 to 0x400 (RAM). With the reset signal, the IVT is copied from 0x60002000 to 0x0 and the pointer to the IVT (VTOR register of the "System Control Block") is set to address 0x0.
- If you do not want to place the IVT into the RAM, undefine the ENABLE_RAM_VECTOR_TABLE symbol from "Preprocessor" and "Linker".

You may combine the RAM_OPTIM* symbols with ENABLE_RAM_VECTOR_TABLE as you want, but some of the combinations do not make sense. It is recommended to use either both or none. If you do not care about the location of the IVT or function placement, leave the configuration as is. By default, the IVT in the RAM is enabled and RAM_OPTIM_HIGH is defined.

To build, run, and debug this configuration, follow these instructions:

1. Make sure that switch SW7 is set to the Hyper Flash boot configuration (SW7-1 OFF, SW7-2 ON, SW7-3 ON, SW7-4 OFF).
2. Select the "Release FLASH" build configuration (Point 1 in Figure 14).
3. Select CMSIS-DAP as the debugger—go to "Project->Options->Debugger" and click the "Setup" tab. Choose CMSIS-DAP as the driver. Leave the other settings in the "Debugger" category as they are. Click "CMSIS DAP" (in the left panel under "Debugger"), choose the "Interface" tab, and select "SWD" in the interface settings. Go to the "Setup" tab and set "Reset" to "Disabled (no reset)".
4. Press the "F7" key or click "Project->Make" to build the project.
5. After a successful build, the application is ready to be downloaded to the Hyper Flash. To perform this action, press "CTRL+D" or click the "Download and Debug" button (Point 7 in Figure 14).
6. As shown in Figure 15, the layout is switched to the debugging mode. The debugger does not perform reset. It just connects to the running target.
    - To pause the running application, press the "Pause" button (Point 2 in Figure 15).
    - To run the application, press the "Go" button (Point 1 in Figure 15).
    - To stop the debugging, press the "Stop" button (Point 3 in Figure 15).

NOTE

To use the on-board MIMXRT1060-EVK hyper flash, reconfigure the hardware board according to the schematic. The default configuration of MIMXRT1060-EVK is set to use the on-board QSPI flash. To build, run, and debug this configuration, follow these steps:

1. Set the SW7: 1 off, 2 off, 3 on 4 off. Power on the board and connect a USB cable to J23.

2. Follow steps 2–5 from the above-mentioned Release FLASH configuration for the hyper flash.

To download the binary into the QSPI flash and boot directly from the QSPI flash, follow these steps:

1. Compile the flash target of the project and get the *pmsm_ref_sol.bin* binary file.

2. Set the SW7: 1 off, 2 off, 3 on 4 off. Power on the board and connect a USB cable to J23.

3. Drop the binary into the “RT1060-EVK” drive on the PC.

4. Wait for the disk to disappear and appear again, which takes a few seconds.

5. Reset the board by pressing SW3, or power the board off and on.

6.2. MCUXpresso IDE

Firstly, the MCUXpresso SDK must be imported into the MCUXpresso IDE:

1. Go to mcuxpresso.nxp.com/en/welcome and click “Select Development Board”.
2. Select “EVK-MIMXRT10xx”. On the right-hand side, click “Build MCUXpresso SDK”. You are going to see something like this (Figure 16):

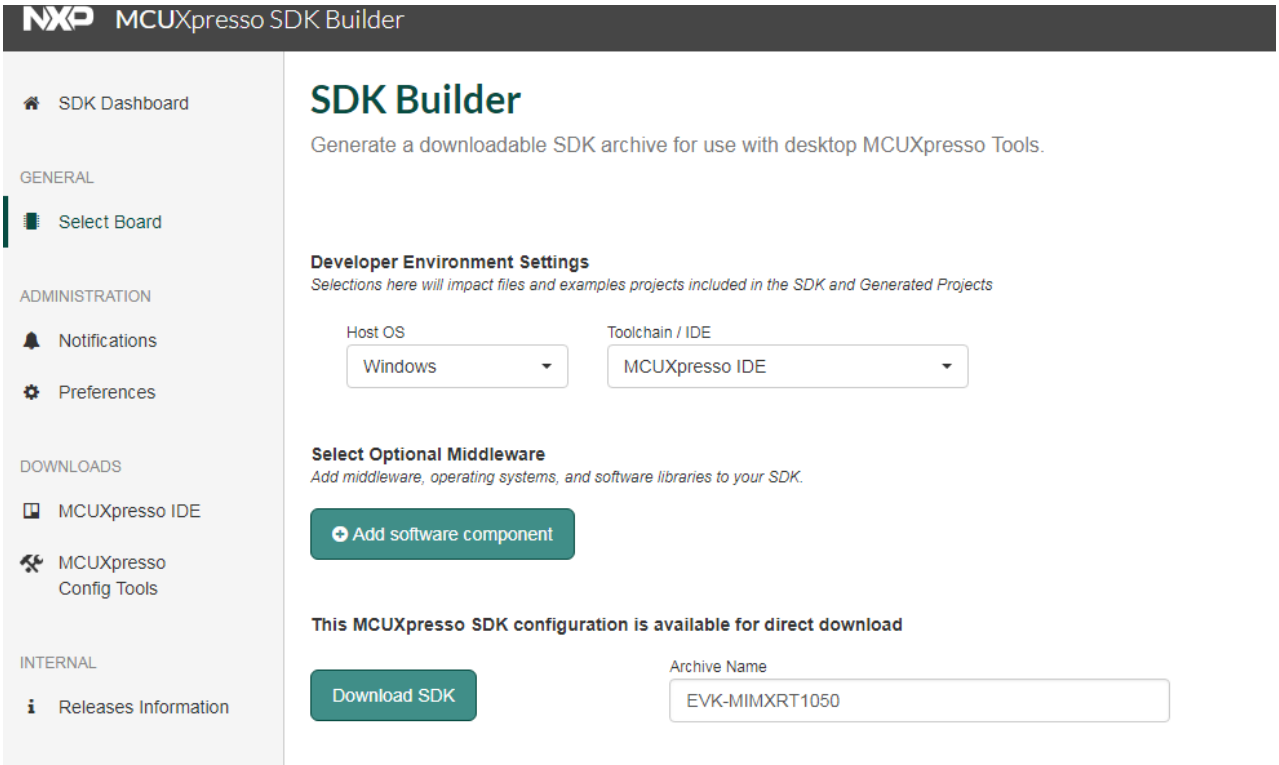

Figure 16. MCUXpresso SDK download

3. Click the “Download SDK” button.
4. Open the MCUXpresso IDE. On the bottom of the workspace, there is a panel with some tools. Click “Installed SDKs” (Figure 17) and drag and drop the downloaded file into the “SDKs” field.

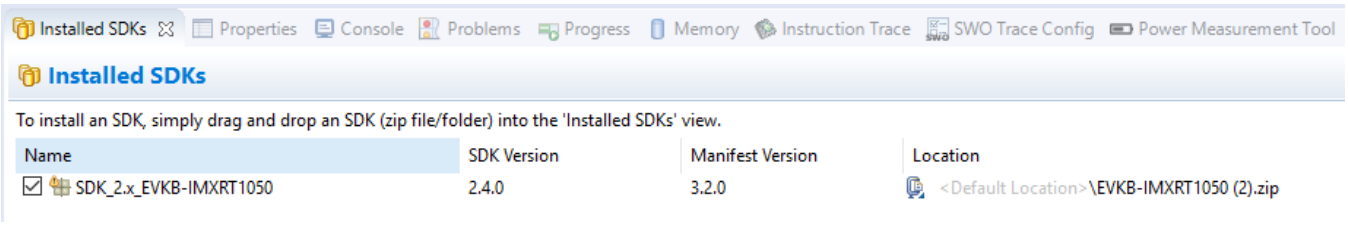

Figure 17. MCUXpresso installed SDKs

5. After waiting a moment to import, the SDK successfully installs into the MCUXpresso IDE.

The second step is to import the project to the IDE. Open the MCUXpresso IDE and click "File->Import". When the new dialog box appears, select "Existing Projects into Workspace" (Figure 18).

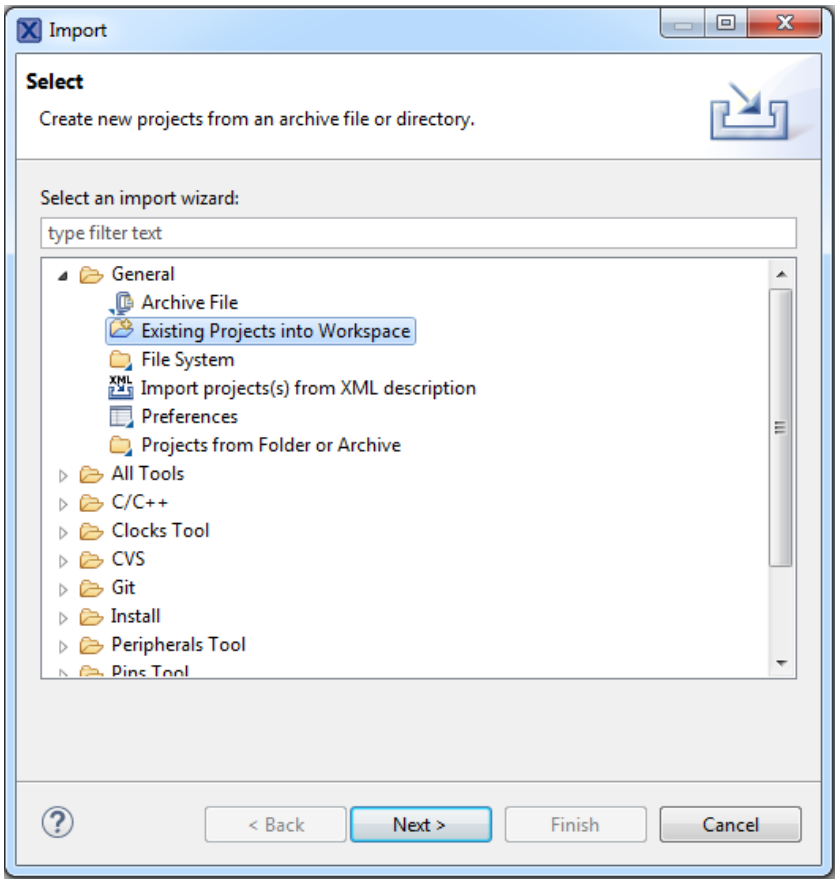

Figure 18. MCUXpresso import

The next step is to select the folder containing the project files. Click the "Browse" button and find the *mcux* folder on your hard disk. In the PMSM project, it is in *build_ref_solutions/evkbimxrt10xx/mcux.* (Figure 19). Confirm the import by clicking the "Finish" button in the corner on the lower right-hand side.

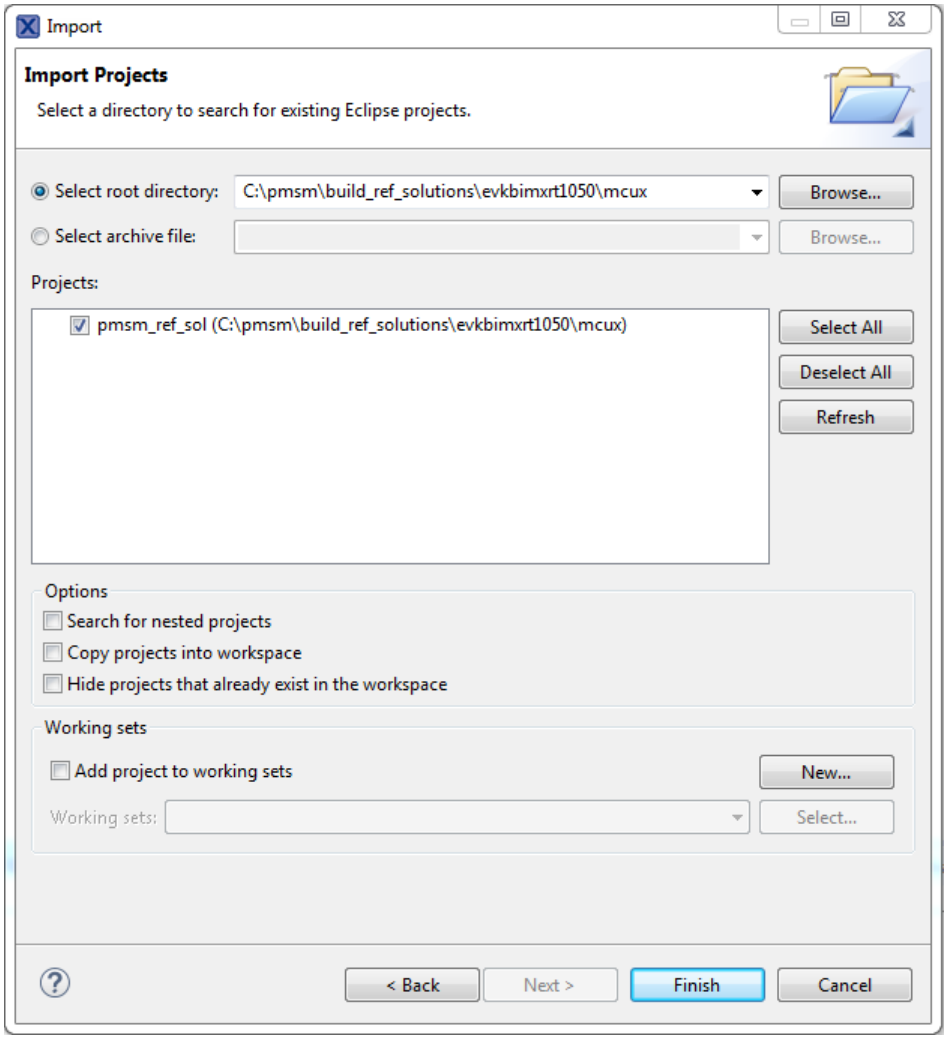

Figure 19. MCUXpresso import

Now you see the MCUXpresso's workspace with the PMSM project imported (Figure 20). The project in the MCUXpresso IDE is fully configured and includes all necessary source and header files required by the application, such as the startup code, clock configuration, and peripherals configuration.

- Point 1—shows the imported project's file structure.
- Point 2—"Build" button. You may also choose the build configuration with a little arrow on this button. There are two build configurations available (Debug RAM and Release FLASH). Each of the two conditions has its own settings (described below).
- Point 3—"Debug" button. This button downloads the compiled project into the MIMXRT10xx-EVK's RAM/FLASH. It also creates the debug configuration (if it does not exist).

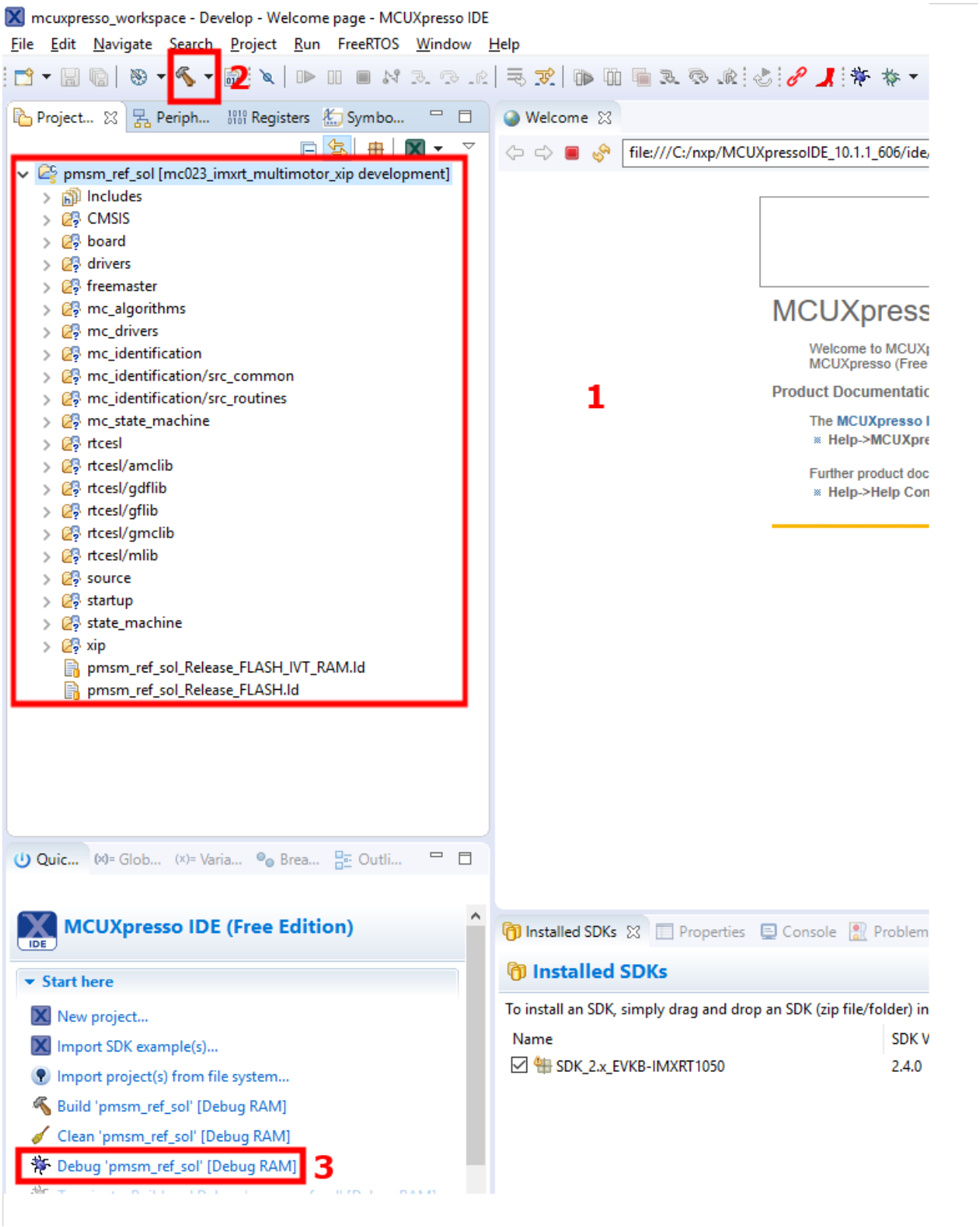

Figure 20. MCUXpresso workspace

The next step is to choose the build configuration.

6.2.1. Debug RAM

This configuration links the application to the RAM memory. The optimization is disabled. To build, run, and debug this configuration, follow these instructions:

1. Select the “Debug RAM” build configuration (Point 2 in Figure 20).
2. Click the “Build” button (Point 2 in Figure 20).
3. Create a new debug configuration by clicking the “Debug” button (Point 3 in Figure 20). A new window with the discovered probes appears (see Figure 21). Choose “DAPLink CMSIS-DAP” and click the “OK” button.

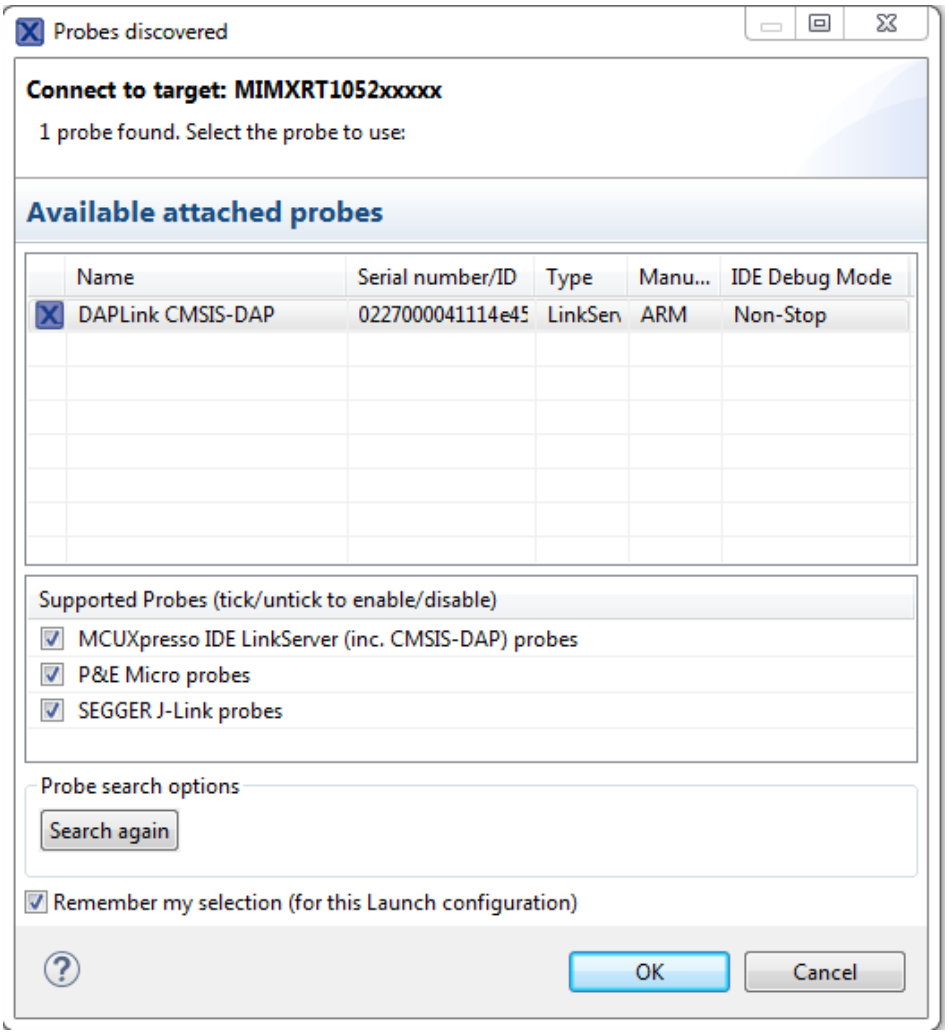

Figure 21. MCUXpresso discovered probes

4. In a few seconds, the compiled application is downloaded to the MIXMRT10xx-EVK’s RAM memory. As shown in Figure 22, the layout is switched to the debugging mode. The program execution stops at the “main” function.
    - To run the application, press the “Resume” button (Point 1 in Figure 22).
    - To pause the running application, press the “Suspend” button (Point 2 in Figure 22).

- To stop debugging, press the “Terminate” button (Point 3 in Figure 22).

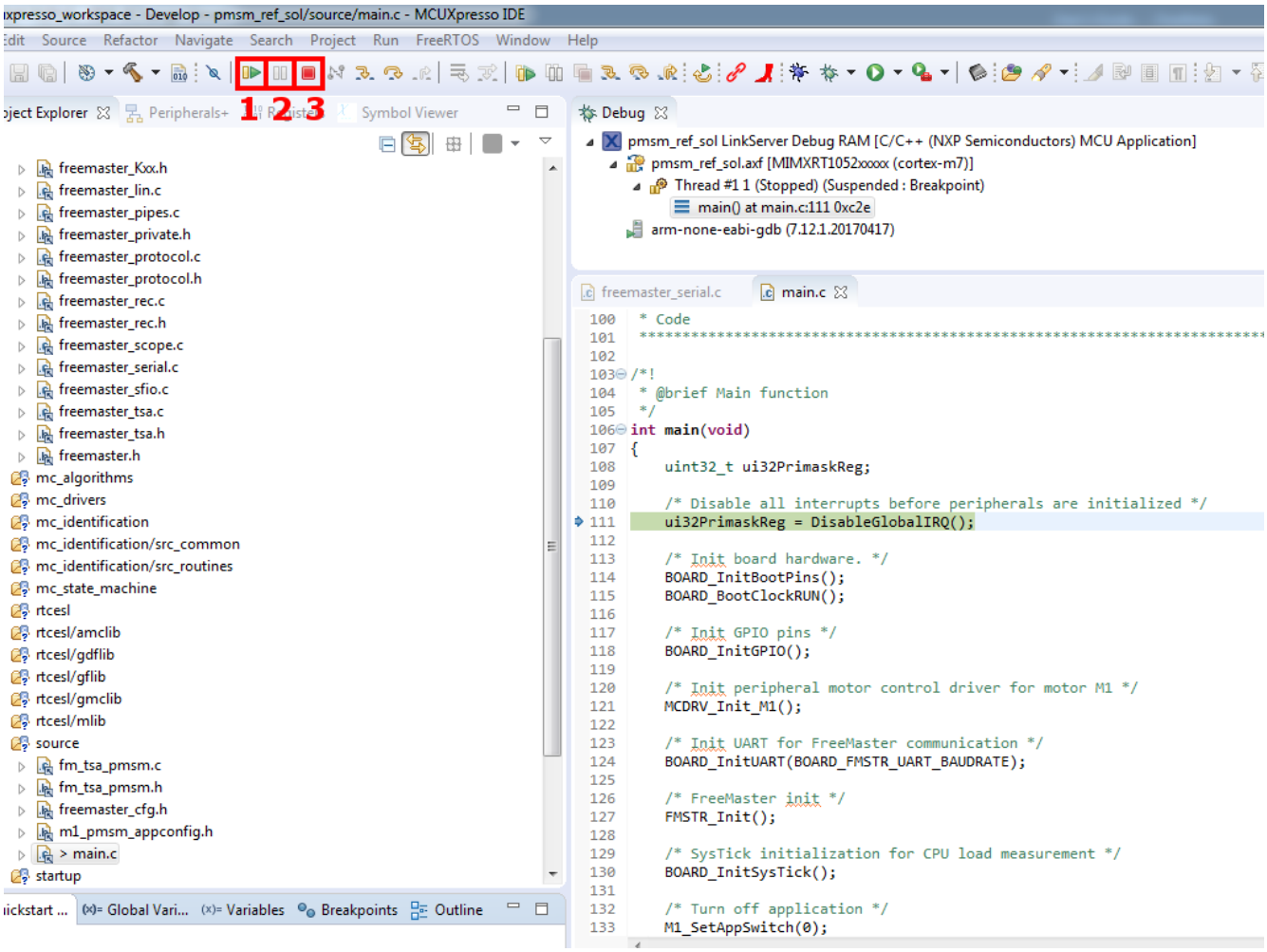

Figure 22. MCUXpresso debug workspace

6.2.2. Release FLASH

This configuration links the application to the Hyper Flash memory. The optimization is set to the “O1” level. The significant difference (opposite to the debug configuration) is the ability to choose the functions which are going to be executed from the RAM. This feature is achieved by defining one of the appropriate symbols in “Preprocessor” (“Project->Properties->C/C++ Build->Settings->Tool Settings”). Select the “Preprocessor” option in the “MCU C Compiler list” settings.

You can also choose whether the Interrupt Vector Table (IVT) is going to be placed into the RAM or Hyper Flash. To place the IVT into the RAM, perform these steps:

- Define the ENABLE_RAM_VECTOR_TABLE symbol in “Preprocessor”. The linker is going to reserve the empty space from address 0x0 to 0x400 (RAM memory). With the reset signal, the IVT is copied from 0x60002000 to 0x0 and the pointer to the IVT (VTOR register of System Control Block) is set to address 0x0.

- Change the linker file path. Go to the “Project->Properties->C/C++ Build->Settings->Tool Settings” tab. Select “Managed Linker Script” in the list of the “MCU Linker” settings and find the “Linker script” text box. Change its value to “pmsm_ref_sol_Release_FLASH_IVT_RAM.ld” (see Figure 23).
- If you do not want to place the IVT into the RAM, undefine the ENABLE_RAM_VECTOR_TABLE symbol and change the file path to “pmsm_ref_sol_Release_FLASH.ld”.

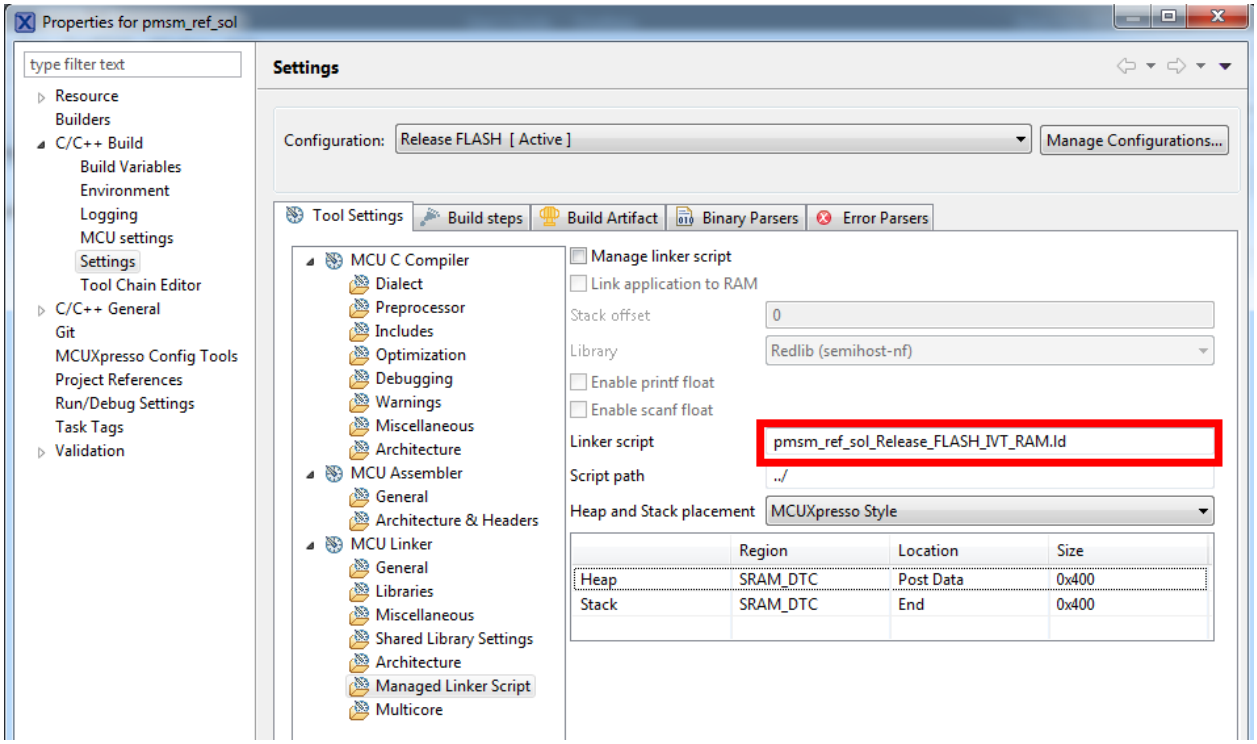

Figure 23. MCUXpresso linker file path

You may combine the RAM_OPTIM* symbols with ENABLE_RAM_VECTOR_TABLE in any way, but it is recommended to use either both or none. If you do not care about the location of the IVT or function, leave the configuration as it is. By default, the IVT in the RAM is enabled and RAM_OPTIM_HIGH is defined.

To build, run, and debug this configuration, follow these instructions:

1. Make sure that switch SW7 is set to the Hyper Flash boot configuration (SW7-1 OFF, SW7-2 ON, SW7-3 ON, SW7-4 OFF).
2. Select the “Release FLASH” build configuration (Point 2 in Figure 20).
3. Click the “Build” button (Point 2 in Figure 20).
4. Create a new debug configuration by clicking the “Debug” button (Point 3 in Figure 14). A new window with the discovered probes appears (see Figure 15). Choose “DAPLink CMSIS-DAP” and click the “OK” button.
5. After a few seconds, the compiled application is downloaded to the MIXMRT10xx-EVK’s Hyper Flash memory. As shown in Figure 22, the layout is switched to the debugging mode. The

debugger does not perform reset; it just connects to the running target.

- To pause the running application, press the “Suspend” button (Point 2 in Figure 22).
- To run the application, press the “Resume” button (Point 1 in Figure 22).
- To stop debugging, press the “Terminate” button (Point 3 in Figure 22).

NOTE

To use the on-board MIMXRT1060-EVK hyper flash, reconfigure the hardware board according to the schematic. The default configuration of MIMXRT1060-EVK is set to use the on-board QSPI flash. To build, run, and debug this configuration, follow these steps:

1. Set the SW7: 1 off, 2 off, 3 on 4 off. Power on the board and connect a USB cable to J23.

2. Follow steps 2–5 from the above-mentioned Release FLASH configuration for the hyper flash.

To download the binary into the QSPI flash and boot directly from the QSPI flash, follow these steps:

1. Compile the flash target of the project and get the *pmsm_ref_sol.bin* binary file.

2. Set the SW7: 1 off, 2 off, 3 on 4 off. Power on the board and connect a USB cable to J23.

3. Drop the binary into the “RT1060-EVK” drive on the PC.

4. Wait for the disk to disappear and appear again, which takes a few seconds.

5. Reset the board by pressing SW3, or power the board off and on.

6.3. Arm-MDK Keil µVision IDE

Open the project file. The *build_ref_solutions/evkbimxrt10xx/mdk* folder contains all necessary files. Double-click *pmsm_ref_sol.uvprojx* to open this project. The project opened in the Keil µVision IDE is fully configured and includes all the source and header files required by the application, such as the startup code, clock configuration, and peripherals configuration.

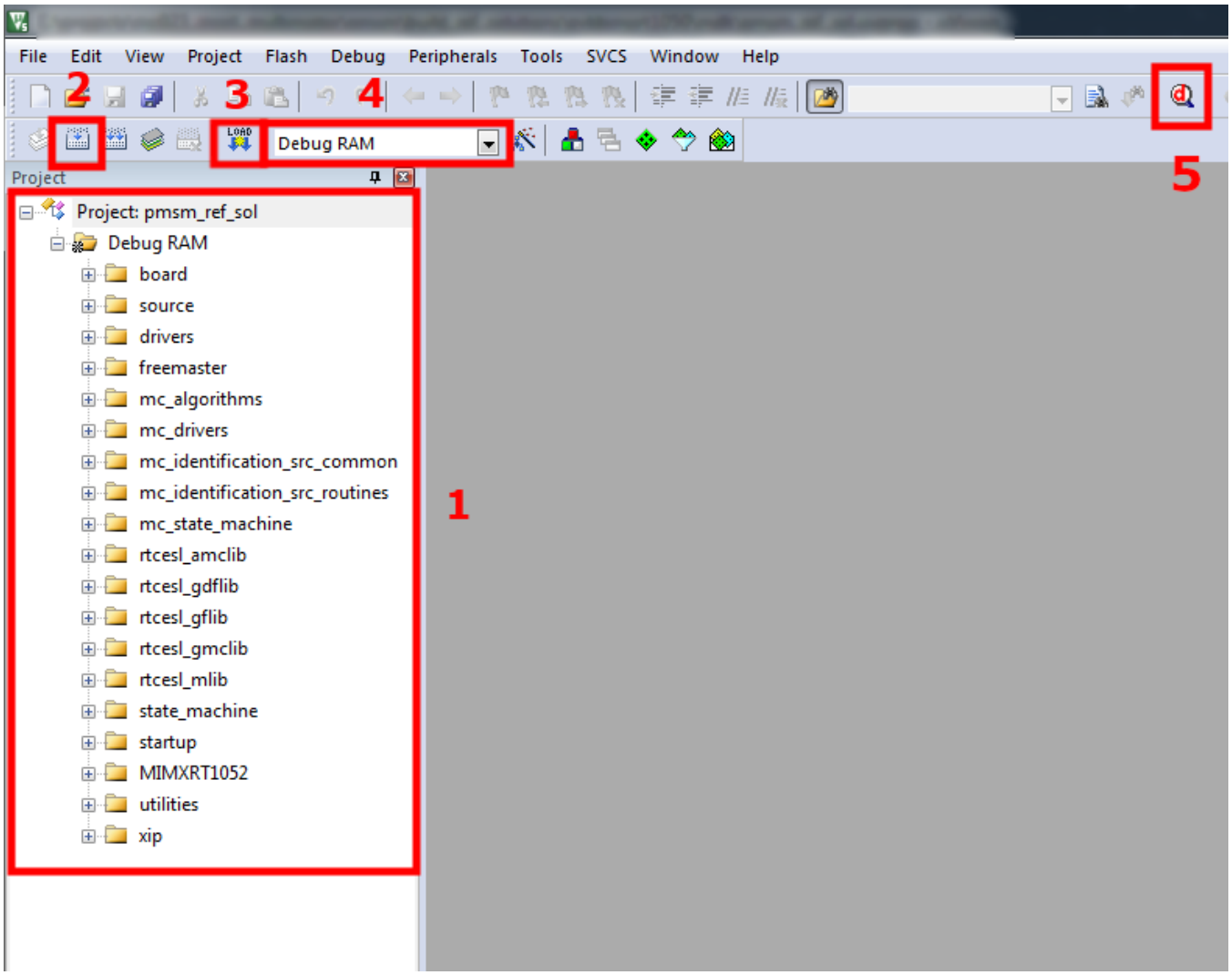

Figure 24. Keil workspace

- Point 1 in Figure 24 shows the imported project's file structure.
- Point 2—the "Build" button.
- Point 3—the "Download" button. This button downloads the compiled project into the MIMXRT10xx-EVK's FLASH.
- Point 4—the "Build configuration" selection. You can switch between the "Debug RAM" and "Release FLASH" build configurations.
- Point 5—the "Debug" button. This button is used to connect to the target application for debugging purposes.

The second step is to choose the build configuration.

6.3.1. Debug RAM

This configuration links the application to the RAM memory. The optimization is disabled. To build, run, and debug this configuration, follow these instructions:

1. Select the “Debug RAM” build configuration (Point 4 in Figure 24).
2. Choose CMSIS-DAP as the debugger. Go to the “Project->Options For Target->Debug” tab. Select “CMSIS-DAP Debugger” in the “Use” option. Switch to the “Utilities” tab and uncheck “Update Target before Debugging”. Leave the other settings as they are, they should be configured.
3. Press the “Build” button (Point 2 in Figure 24).
4. After a successful build, the application is ready for download to the RAM. To perform this action, press CTRL+F5 or click the “Debug” button (Point 5 in Figure 24).
5. As shown in Figure 25, the layout is switched to the debugging mode.
    - To run the application, press the “Run” button (Point 1 in Figure 25).
    - To pause the running application, press the “Pause” button (Point 2 in Figure 25).
    - To stop debugging, press the “Debug” button (Point 5 in Figure 24).

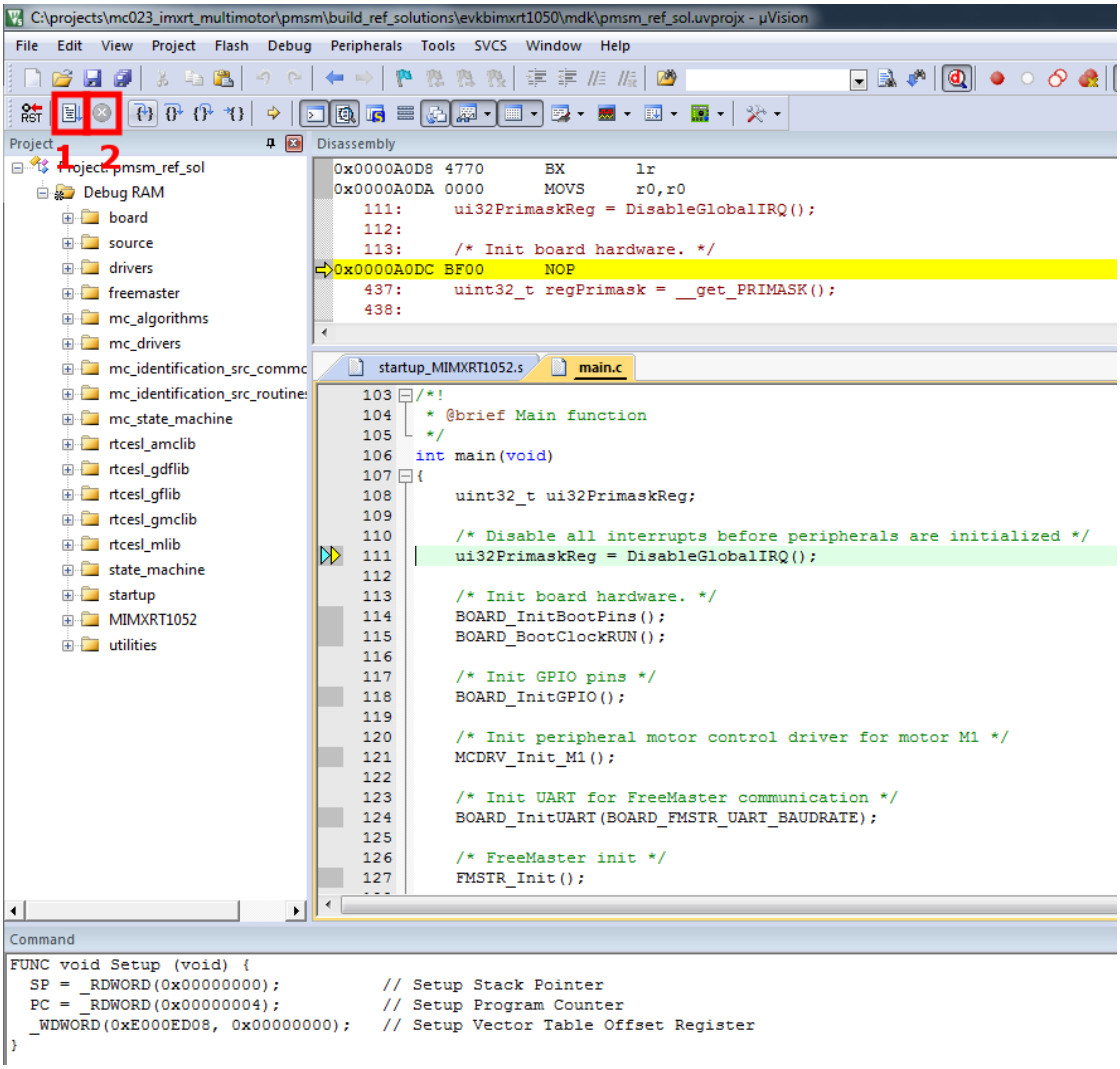

Figure 25. Keil debug workspace

6.3.2. Release FLASH

This configuration links the application to the Hyper Flash memory. The optimization is set to the highest level. The significant difference (opposite to the debug configuration) is the ability to choose the functions which are executed from the RAM. This feature is achieved by defining one of the appropriate symbols in “Preprocessor” (“Project->Options for Target->C/C++” tab, “Define” box).

You can also choose, whether the Interrupt Vector Table (IVT) is going be placed into the RAM or Hyper Flash. To place the IVT into the RAM, perform these steps:

- Define the ENABLE_RAM_VECTOR_TABLE symbol in “Preprocessor”. The linker reserves an empty space from address 0x0 to 0x400 (RAM). With the reset signal, the IVT is copied from 0x60002000 to 0x0 and the pointer to the IVT (VTOR register of the system control block) is set to address 0x0.
- Change the linker file path. Go to the “Project->Options for Target->Linker” tab. Find the “Scatter file” text box. Change its value to *“.\MIMXRT1052xxxxx_flexspi_nor_ivt_ram.scf*” (see Figure 26).
- If you do not want to place the IVT into the RAM, undefine the ENABLE_RAM_VECTOR_TABLE symbol and change the file path to “*.\MIMXRT1052xxxxx_flexspi_nor.scf*”.

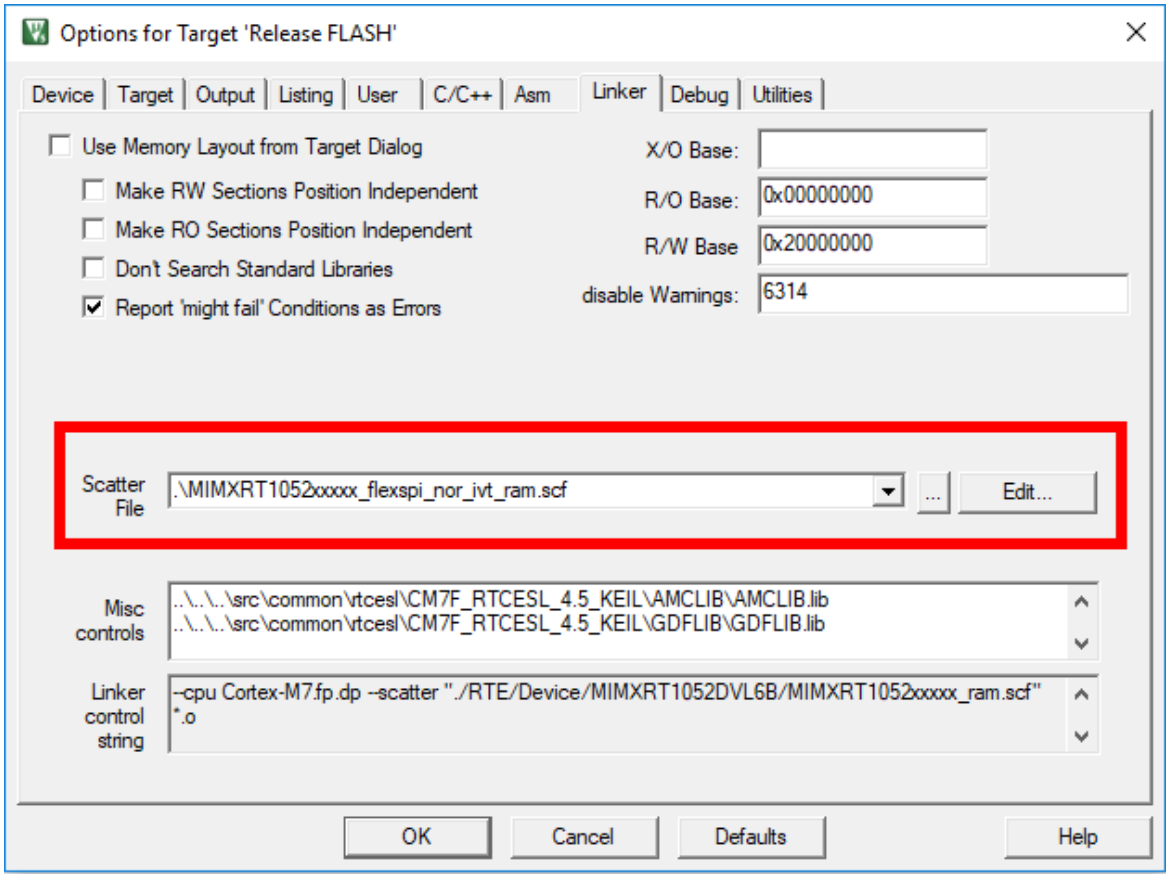

Figure 26. Keil linker settings

You may combine the RAM_OPTIM* symbols with ENABLE_RAM_VECTOR_TABLE as you want, but it is recommended to use either both or none. If you do not care about the IVT location or function placement, leave the configuration as it is. By default, the IVT in RAM is enabled and RAM_OPTIM_HIGH defined.

To build, run, and debug this configuration, follow these instructions:

1. Make sure that switch SW7 is set to the Hyper Flash boot configuration (SW7-1 OFF, SW7-2 ON, SW7-3 ON, SW7-4 OFF).
2. Select the “Release FLASH” build configuration (Point 4 in Figure 24).
3. Select CMSIS-DAP as the debugger. Go to “Project->Options For Target->Utilities” tab and uncheck “Update Target before Debugging”. Switch to the “Debug” tab and set the options as shown in Figure 27.

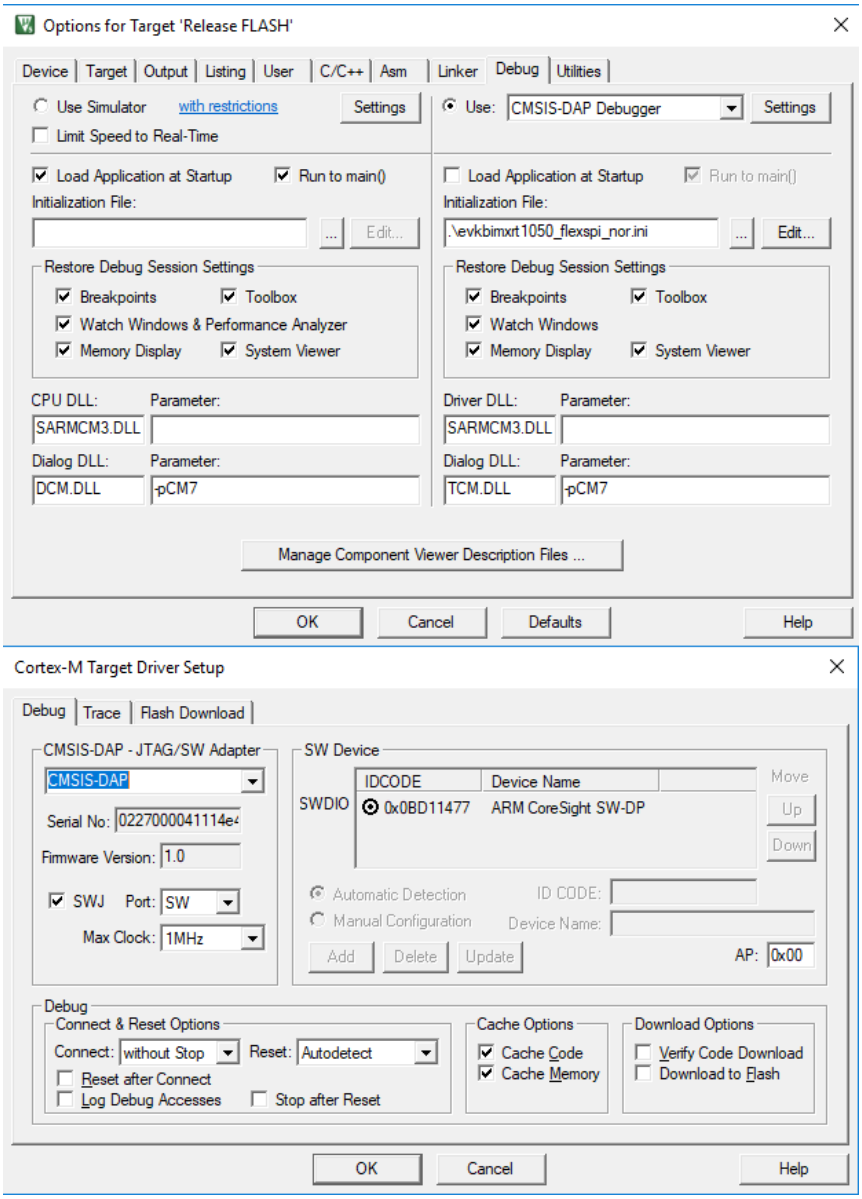

Figure 27. Keil—"Release FLASH” debugger settings

4. Press the “Build” button (Point 2 in Figure 24).
5. After a successful build, the application is ready to be downloaded to the Hyper Flash. To perform this action, click the “Download” button (Point 3 in Figure 24). If any error occurs, perform a mass erase (click “Flash->Erase”) and repeat this step.
6. After the compiled project is flashed, press “CTRL+F5” or click the “Debug” button (Point 5 in Figure 24).
7. As shown in Figure 25, the layout switches to the debugging mode. The debugger does not perform reset; it just connects to the running target.
   - To pause the running application, press the “Pause” button (Point 2 in Figure 26).
   - To run the application, press the “Run” button (Point 1 in Figure 26).
   - To stop debugging, press the “Debug” button (Point 4 in Figure 25).

NOTE

To use the on-board MIMXRT1060-EVK hyper flash, reconfigure the hardware board according to the schematic. The default configuration of MIMXRT1060-EVK is set to use the on-board QSPI flash. To build, run, and debug this configuration, follow these steps:

1. Set the SW7: 1 off, 2 off, 3 on 4 off. Power on the board and connect a USB cable to J23.

2. Follow steps 2–5 from the above-mentioned Release FLASH configuration for the hyper flash.

To download the binary into the QSPI flash and boot directly from the QSPI flash, follow these steps:

1. Compile the flash target of the project and get the *pmsm_ref_sol.bin* binary file.

2. Set the SW7: 1 off, 2 off, 3 on 4 off. Power on the board and connect a USB cable to J23.

3. Drop the binary into the “RT1060-EVK” drive on the PC.

4. Wait for the disk to disappear and appear again, which takes a few seconds.

5. Reset the board by pressing SW3, or power the board off and on.

6.4. Replacing the firmware of the OpenSDA

To replace the firmware of the OpenSDA to support downloading to the Hyper Flash, follow these instructions:

1. Download the latest firmware application from www.nxp.com/opensda (choose DAPLink vxxxx - Supports on board hyper-flash).
2. Power down the board. Jumper J27 should be set to position 1-2 for the bootloader mode.
3. Press and hold the SW4 button and power on the board simultaneously. Release the SW4 button after one second.

4. After several seconds, a new drive with the “MAINTENANCE” label appears. Drag and drop or copy and paste the downloaded firmware file to that drive. After a few seconds, the drive disappears.
5. Reset the board.
6. The firmware is now replaced and you can flash your application to the Hyper Flash.

6.5. 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 occurs in situations where you implement the function as a supporting function for better usability, but you do not use the function for any special purposes for a while.

The IAR Embedded Workbench IDE suppresses these warnings:

- Pa082—undefined behavior; the order of volatile accesses is not defined in this statement.
- Ta022—possible ROM access (<*ptr*>) from within a *__ramfunc* function.
- Ta023—call to a non *__ramfunc* function (*somefunction*) from within a *__ramfunc* function.

The Arm-MDK Keil µVision IDE suppresses these warnings:

- 66—enumeration value is out of the “int” range.
- 1035—a single-precision operand is implicitly converted to a double-precision operand.
- 1296—extended constant initializer used.

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

7. User interface

The application contains the demo application mode to demonstrate the motor rotation. You may operate it either using the user button or FreeMASTER. The MIXMRT10xx-EVK board includes the user button associated with a port interrupt (generated whenever the button 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. There is also an LED indication of the current application state. A continuous green LED indicates that the application is in the RUN state, a flashing LED indicates the FAULT state, and the LED off indicates 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 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 *pmsm_ref_sol.pmp* file located in the */freemaster/* folder. The FreeMASTER application starts, and the environment is created automatically.

You can control the application using these two interfaces:

- Buttons on the NXP MIMXRT10xx-EVK boards.
- Remote control using FreeMASTER.

7.1. 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.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*. The PMSM project uses the TSA by default, so it is not necessary to select a symbol file for FreeMASTER (see Section 7.1.1, "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.

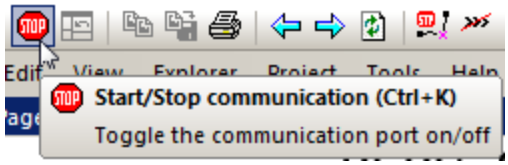

Figure 28. Red "STOP" button placed in top left-hand corner

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 pop-up window appears.

Figure 29. FreeMASTER—communication is established successfully

3. Control the PMSM motor using the Control page (Section 7.1.2, "Control page").
4. If you rebuild and download new code to the target, turn the application off and on.

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

5. Go to the "Project->Options->Comm" tab and make sure that "SDA" is set in the "Port" option and the communication speed is set to 115200 bit/s.

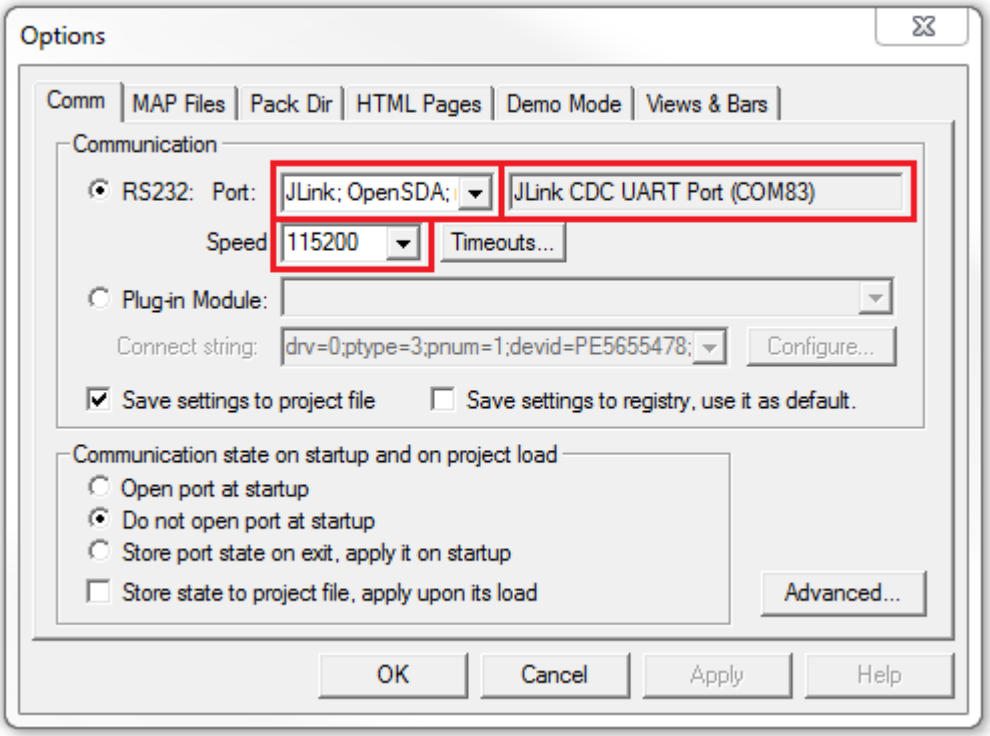

Figure 30. FreeMASTER communication setup window

6. 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.
7. Make sure to supply your development board from a sufficient energy source. Sometimes the PC USB port is not sufficient to supply the development board.

7.1.1. FreeMASTER TSA and user variables addition to FreeMASTER watch

By default, all projects use the Target Side Addressing (TSA). This means that the information about the variables' address and size is stored in the MCU flash memory. The TSA feature may be enabled or disabled by the BOARD_FMSTR_USE_TSA macro in the *src/projects/evkbimxrt10xx/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_ref_solutions/evkbimxrt10xx<compiler>/<Debug RAM or Release FLASH>* folder. The symbol files have different extensions for different compilers (IAR generates **.out*, MCUXpresso generates **.axf*, and Keil generates **.axf* file). For more information about the TSA, see *FreeMASTER Serial Communication Driver* (document FMSTERSCIDRVUG).

7.1.2. 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).

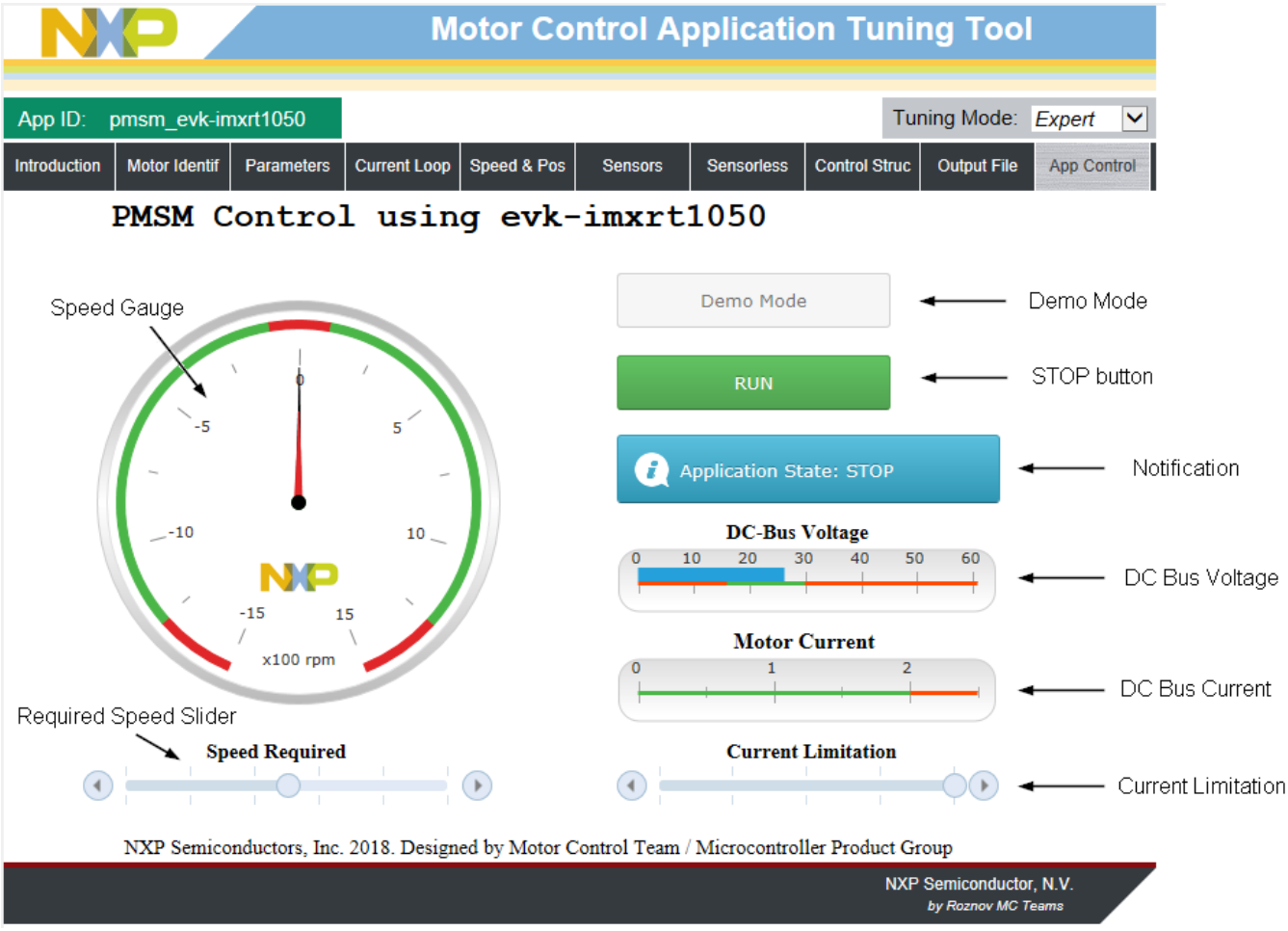

Figure 31. FreeMASTER control page

The MCAT tool is targeted for 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 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 NXP hardware according to the instructions in Section 2, “Hardware setup”.
2. Download the project into the EVK’s memory 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 7.1, “Remote control using FreeMASTER”.
4. Set the required motor speed using the speed slider, as shown in Figure 31.

8.2. Stopping the motor

1. Click the “STOP” button in the FreeMASTER control page (Figure 31).
2. Set the speed to zero using the speed slider.
3. In case of an emergency, turn off the power supply.

8.3. Clearing the fault

To clear the fault, click the fault notification in the control page (Figure 31).

8.4. Turning the demonstration mode on/off

1. If you use the FreeMASTER control page, click the “Demo” button.
2. If you do not use the FreeMASTER control page, push the corresponding button on the EVK board to turn the demonstration mode on/off, as described in Section 7, “User interface”.

9. Acronyms and abbreviations

Table 9. Acronyms and abbreviations

Term	Meaning
AC	Alternating Current
AN	Application Note
DRM	Design Reference manual
EVK	Evaluation Kit
FOC	Field-Oriented Control
IVT	Interrupt Vector Table
MCAT	Motor Control Application Tuning tool
MCU	Microcontroller
MSD	Mass Storage Device
PMSM	Permanent Magnet Synchronous Motor
TSA	Target Side Addressing

10. References

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

- *i.MX RT1020 Processor Reference Manual* (document IMXRT1020RM).
- *i.MX RT1050 Processor Reference Manual* (document IMXRT1050RM).
- *PMSM Field-Oriented Control on MIMXRT10xx EVK* (document AN12214).
- *MIMXRT1050 EVK Board Hardware User's Guide* (document MIMXRT1050EVKHUG).
- *Sensorless PMSM Field-Oriented Control* (document DRM148).
- *Tuning Three-Phase PMSM Sensorless Control Application Using MCAT Tool* (document AN4912).
- *Motor Control Application Tuning (MCAT) tool for 3-Phase PMSM* (document AN4642).
- *How to Enable Debugging for FLEXSPI NOR Flash* (document AN12183).
- *i.MXRT1060 Processor Reference Manual* (document IMXRT1060RM).

How to Reach Us:

Home Page:
www.nxp.com

Web Support:
www.nxp.com/support

Document Number: PMSMFOCRT10xxUG
Rev. 0
12/2018