

UG10136

Getting Started with the S32K396 MBDT Motor Control Demo

Rev. 1.0 — 23 May 2024

User guide

Document information

Information	Content
Keywords	S32K396 MBDT MATLAB FreeMASTER PIL
Abstract	This documentation describes how to run the demo project. It includes the compilation of the project and the connection between the hardware and the operation of the FreeMASTER, also introduced the PIL simulation process.



1 Introduction

The Model-Based Design Toolbox (MBDT) enables the model-based design workflow targeting NXP processors via the MATLAB and Simulink environments.

NXP MBDT integrated the system and peripheral devices interface blocks and their real-time Drivers(RTD), Automotive Math and Motor Control Library(AMMCLIB), Compilers and Toolchain. It is a Simulink-embedded target supporting NXP MCUs for direct rapid prototyping and built-in support for software-in-the-loop and processor-in-the-loop (SIL and PIL) development workflows. It also generates and deploys code automatically to start up the MCU and run complex applications, which enable control engineers and embedded developers to shorten project life cycles.

This documentation describes how to run the demo project. It includes the compilation of the project and the connection between the hardware and the operation of the FreeMASTER. The details about the demo design and migration are not included. See the related documents on www.nxp.com for more information.

This software is validated on the S32K396 Evaluation Boards (EVB), Mother Board (MB), and two of the 3-phase LV PWR board.

2 Hardware, Software, and Tools

The following is a list of hardware and tools required:

- Boards: S32K396 EVB and 3-phase LV PWR board
- Debugger: Lauterbach for Cortex-M7/Multilink PE micro debugger
- PCIE Cable
- USB Cable

The following is a list of software and tools required:

- AUTOSAR Tool: EB Tresos 29.0.0
- GUI: FreeMASTER 3.2.0.5
- MCAL: SW32K3_RTD_4.4_R21-11_3.0.0_P01_HF01
- IDE: S32 Design Studio for S32 Platform 3.5
- MATLAB R2023a
- Embedded Coder Support Package for ARM Cortex-M Processor version 23.1.0
- NXP_Support_Package_S32K3xx version 1.4.0
- NXP_MBDToolbox_S32K3xx version 1.4.0

Note: MCAL SW32K3_RTD_4.4_R21-11_3.0.0_P01_HF01 installation can cause an error that the installation path is too long. Click ignore to continue the installation.

3 Build demo

If you want to run the demo without any modifications, you don't need to build the project. The built binary (`s32k396_pmsm_mc_mbd/mbd/mbd_ebt.elf`) is provided in the release package.

Before building this project, ensure that the tools have been installed. The following steps give a guide to building the demo.

3.1 Open model and setup configuration

Add folder '`data_dictionary`' and '`modules`' into the MATLAB path. For example, see [Figure 1](#).

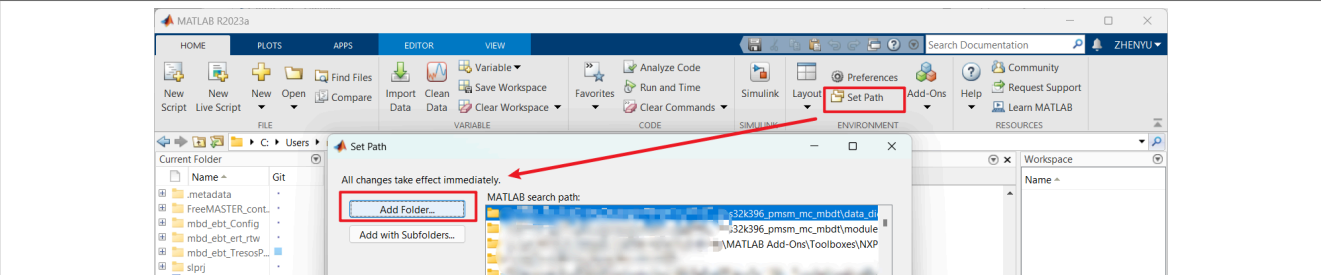
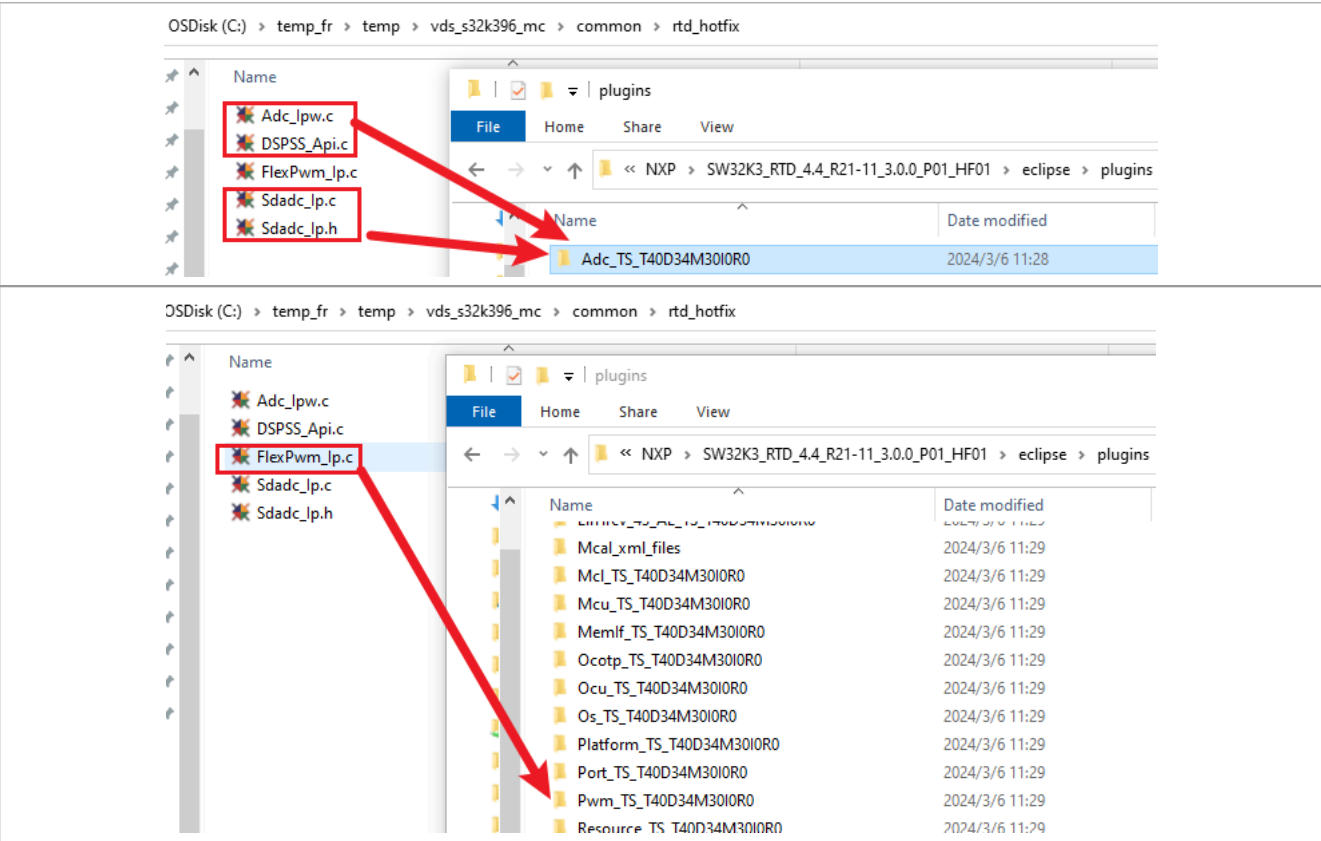


Figure 1. Add folder to MATLAB path

Hotfix replaces RTD drivers: copy files from "common/rtd_hotfix" to the RTD installation path “NXP/SW32 K3_RTD_4.4_R21-11_3.0.0_P01_HF01/xxxx” and replace each module driver file(including ADC, DSP, and eflexpwm).



Change the workspace to the project folder `s32k396_pmsm_mc_mbdt/mbd`. Then open the project model `mbd_ebt.slx` in MATLAB.

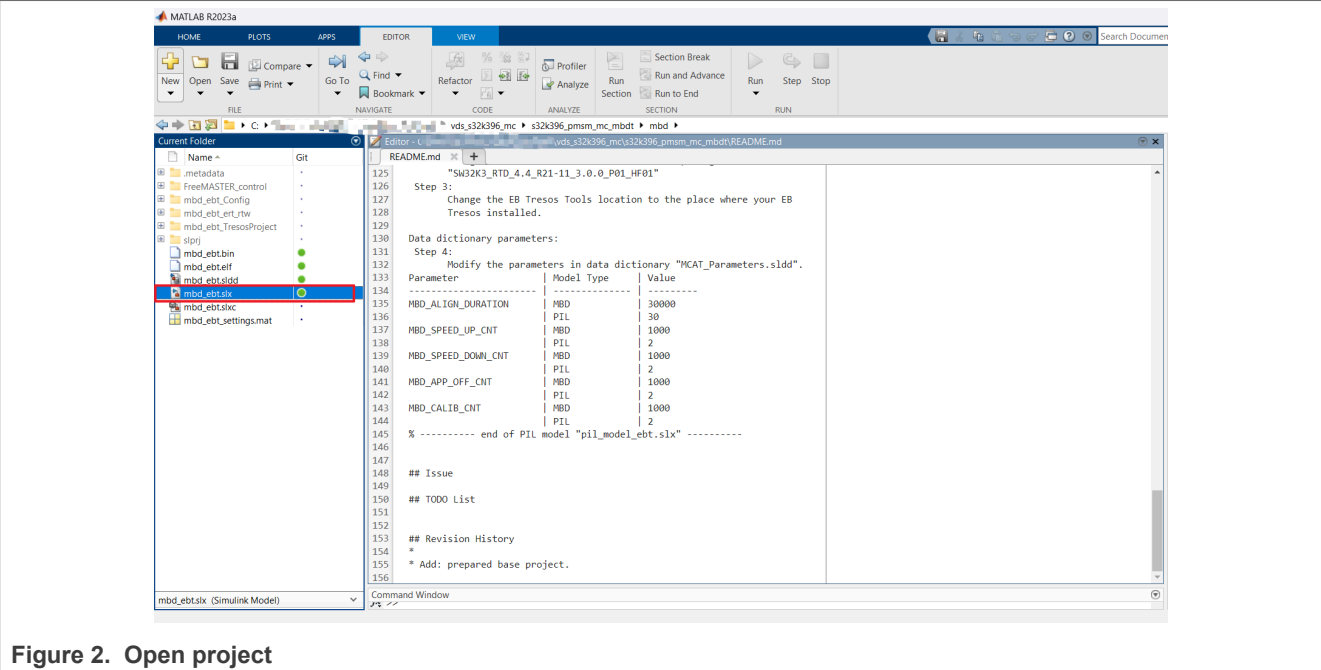


Figure 2. Open project

Change Project Configurations:

In the model, click 'Hardware Settings'.

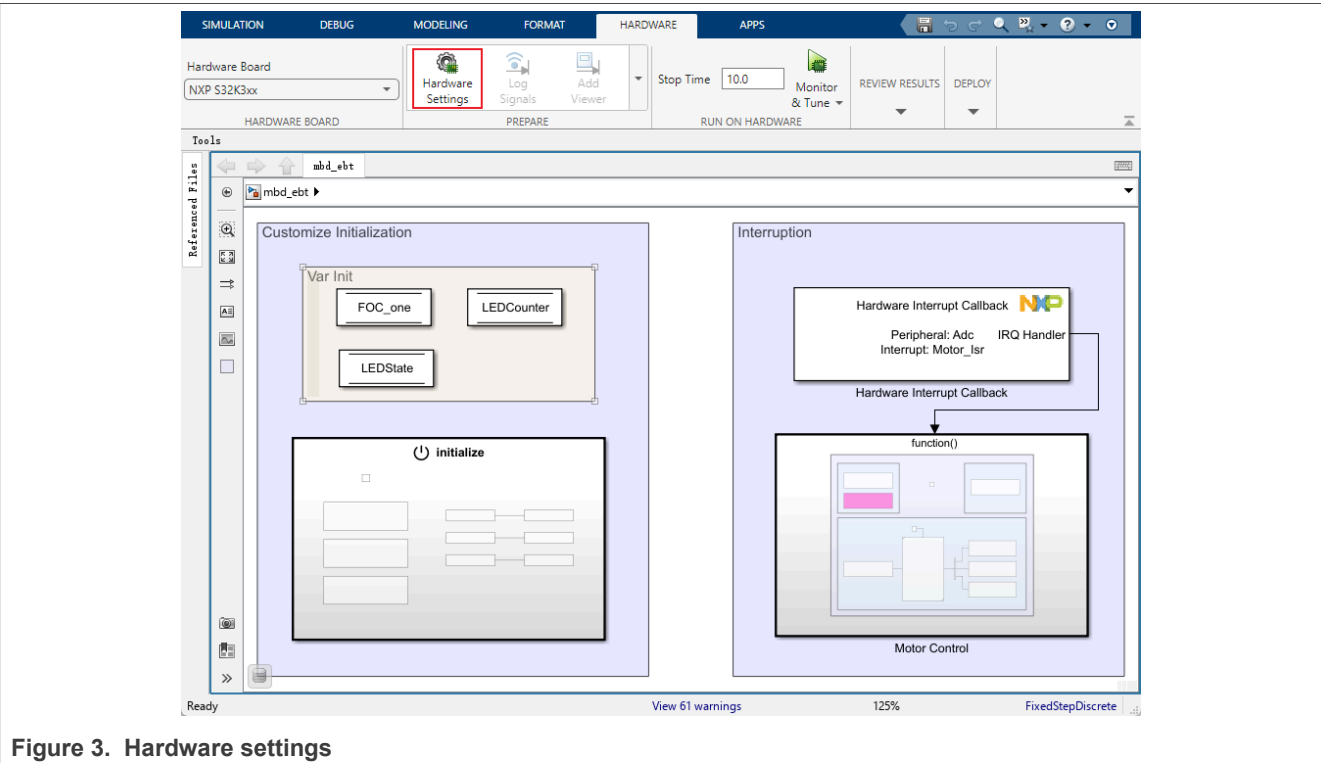


Figure 3. Hardware settings

1. Change the EB project Template.
In the Configuration Parameters options, click *Hardware implementation -> Hardware board settings -> Hardware*, and configure the Configuration Template as 'mbd_TresosProject', which is in the folder *s32k396_pmsm_mc_mbd/ebt_default_project_config*.

In the dialog box, select '*Overwrite the existing folder*'. After that, click 'Apply' to save this configuration. [Figure 4](#) shows the steps.

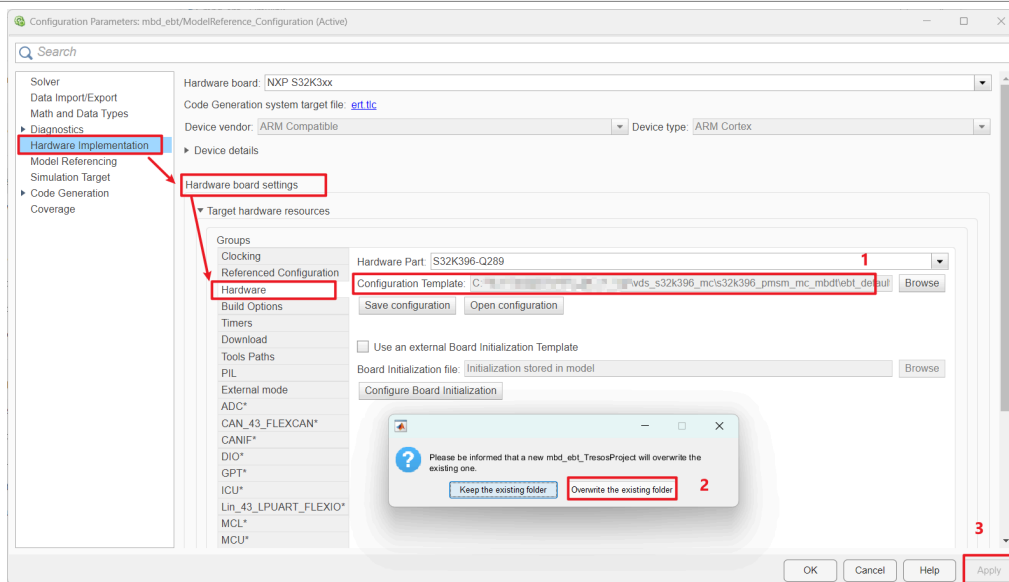


Figure 4. Change the configuration template

2. Change EB tool path and RTD location.

In the Configuration Parameters options, click *hardware implementation* -> *Hardware board settings* -> *Tool Paths*, and change the '**EB Tresos Tools location**' to the location where the tools are installed in the computer.

Then, change the RTD location to the place where the "SW32K3_RTD_4.4_R21-11_3.0.0_P01_HF01" version is installed. In the dialog box, select '*Overwrite the existing folder*'. After that, click 'Apply' to save this configuration. [Figure 5](#) shows the steps.

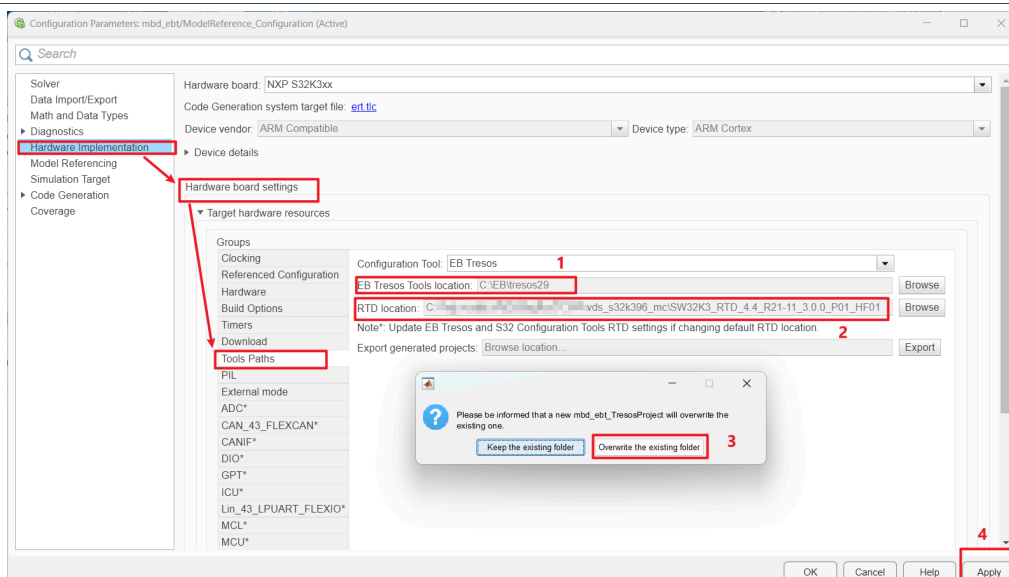


Figure 5. Change tool path and RTD location

3.2 Build the project

Click the '**Build, Deploy, and Start**' button. Then the project starts to generate codes and build. When the build is finished, an executable file '**mbd_ebt.elf**' can be found in the folder **s32k396_pmsm_mc_mbd/mbd**. Also, a binary file is generated in the same folder '**mbd_ebt.bin**'.

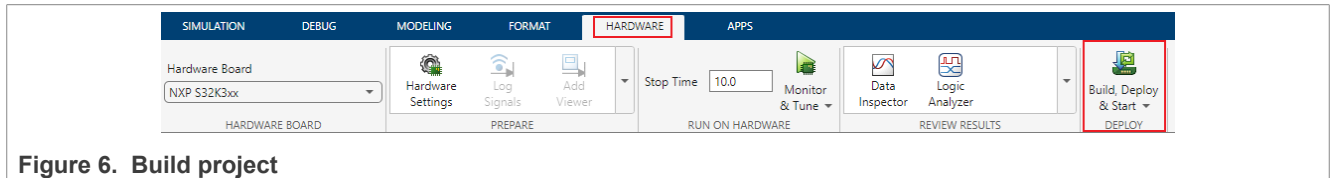


Figure 6. Build project

Also, there is another way to build the project. Open the '*Embedded Coder*' app in the APPS gallery. Then, click the '**Build**' button.

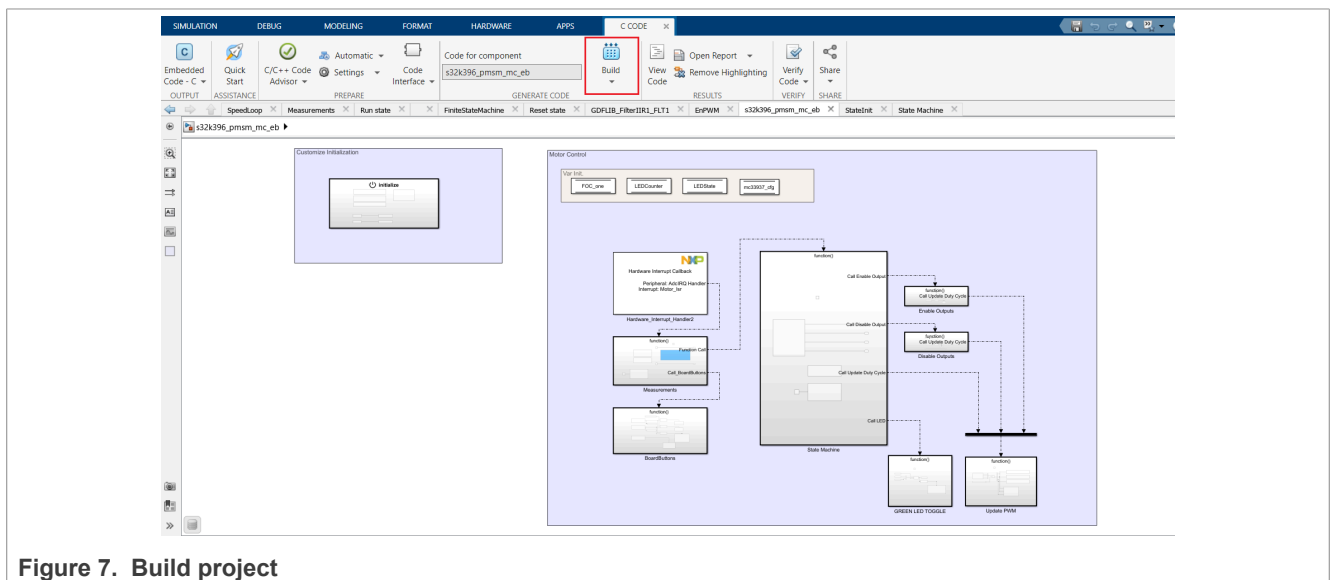


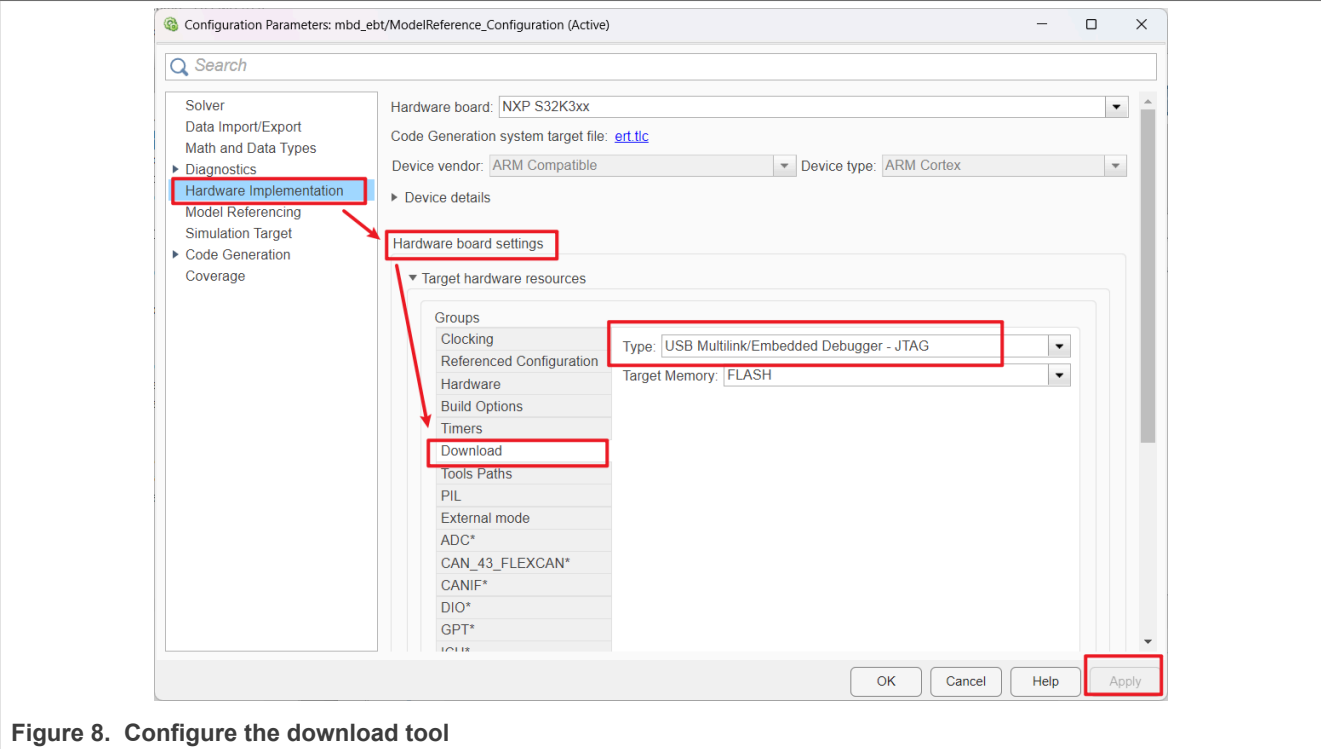
Figure 7. Build project

3.3 Download elf file into board

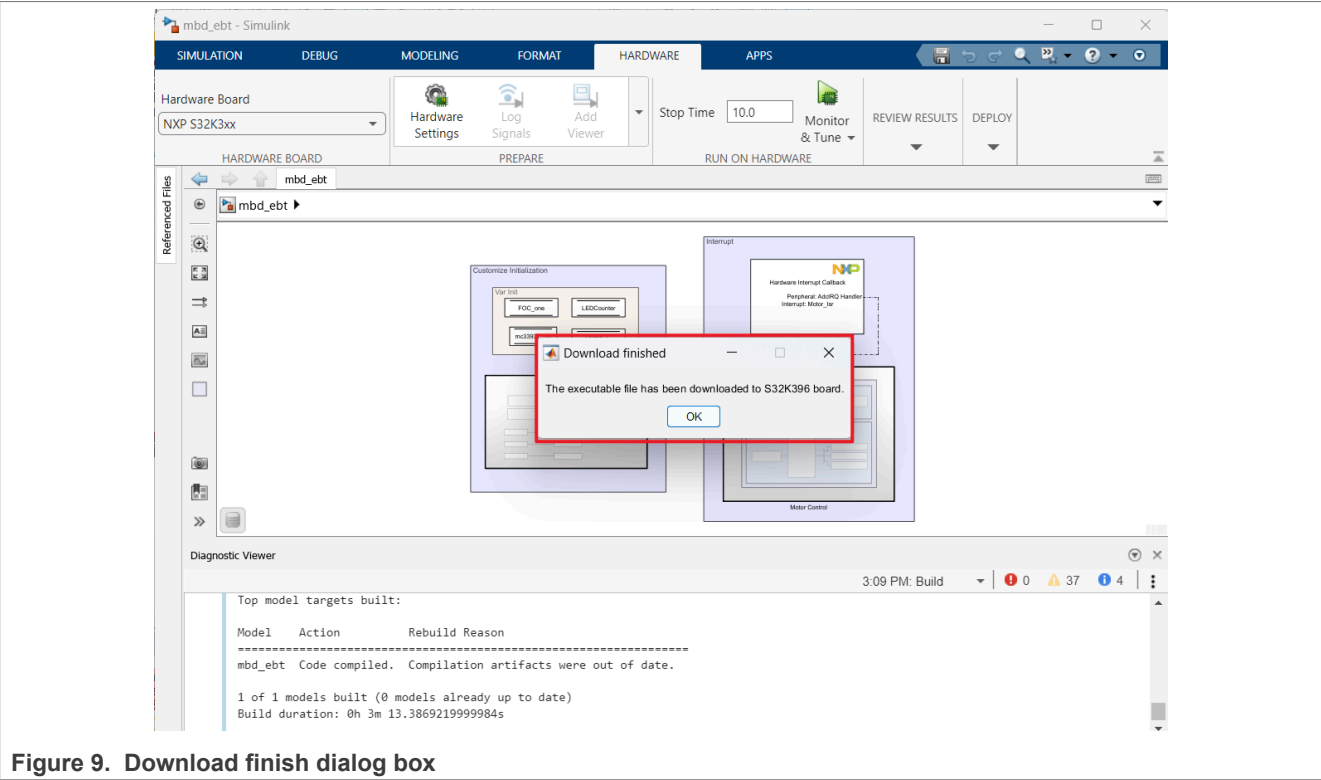
3.3.1 Download via Multilink PE micro debugger

If using the PE debugger to download the project elf file, it can be automatically processed after build. Before building the project, ensure that the configuration is selected as PE debugger in the configuration parameters.

To configure the PE debugger as the download tool, follow the steps shown in [Figure 8](#). Click **Apply** before closing the configuration to save the changes.



When the project finishes build, see [Section 3.2](#), the binary is loaded into the board automatically. And a dialog box pops up. See [Figure 9](#).



3.3.2 Download via Lauterbach

If using the debugger Lauterbach, Simulink cannot download the file. So when the **Download failed** dialog box pops up, abort it, and that means the project has been built and compiled successfully. Then, load the generated file 'mbd_ebt.elf' through the Lauterbach script '*debug/s32k396_flash_c0.cmm*'.

4 Run demo

4.1 Connect hardware

Warning: Do not power-on while connecting any hardware.

As shown in [Figure 10](#), the EVB board is the central controller and the predriver with motor is the actuator. Download the compiled file through the debugger and observe the variables in FreeMASTER using the USB cable. To connect all the hardware devices perform the following steps:

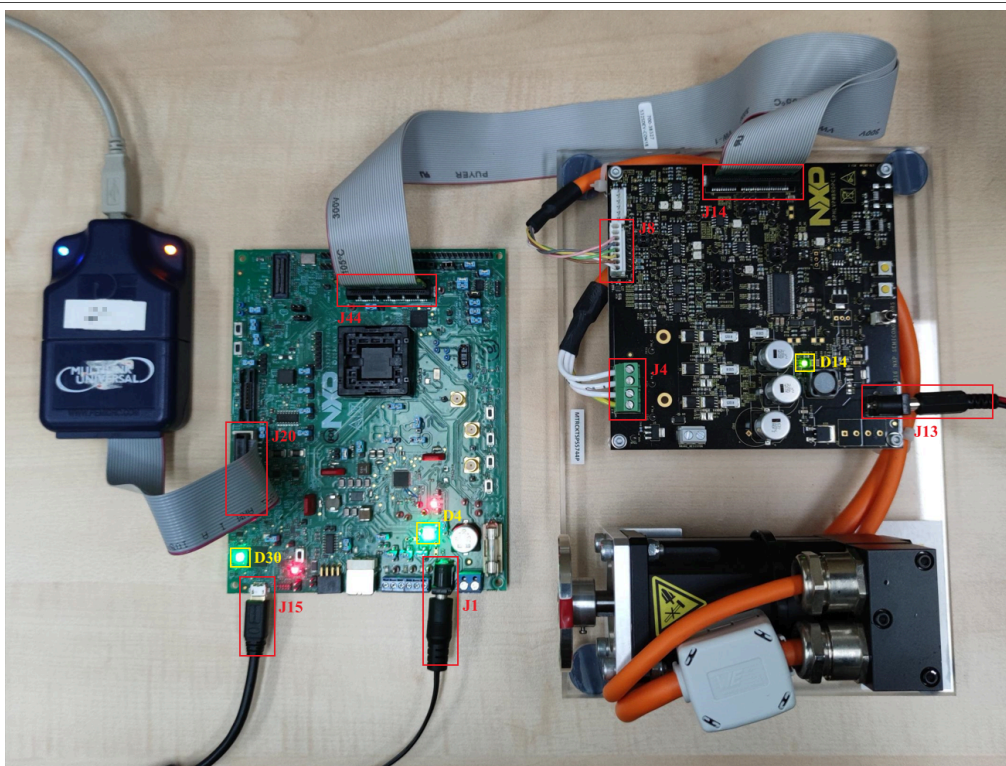


Figure 10. Demo connection

1. Connect the three-phase winding of the PMSM to J4 on the predriver and connect the resolver signal to J8 on the predriver.
2. Connect motor control signals from J14 on the predriver to J44 on the EVB board.
3. Connect the debugger between JTAG J20 and the computer.
4. Connect the USB cable between J15 and the computer, then the EVB LED D30 will be solid green.
5. Plug in the 12V power supply port J14 on the EVB board, then the EVB LED D4 will be solid green.
6. Plug in the 24V driver power on port J13 of the predriver, and then the predriver LED D14 will be solid yellow.

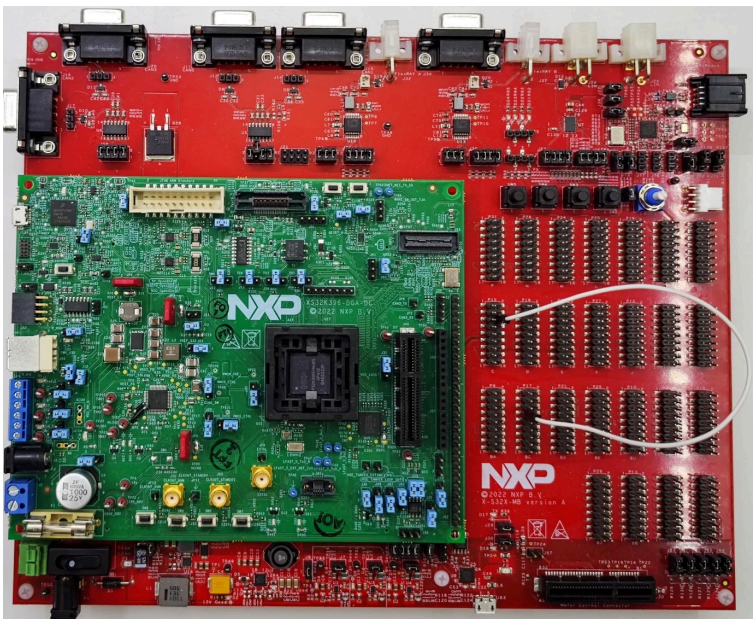


Figure 11. S32K396 EVB

Table 1 shows more details to prepare the EVB board connection for this project. Ensure that all the jumpers on the S32K396 EVB(SCH-54614_A) are in the correct positions.

Note: If the S32K396 EVB version is SCH-55517_B, J71 should connect with 2-3 to use the push button.

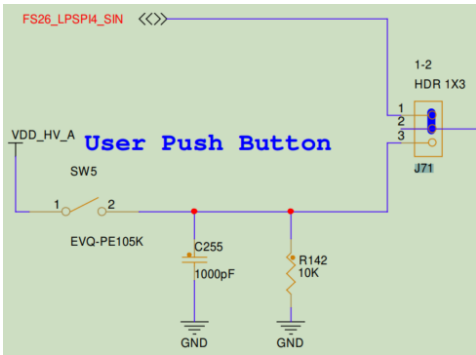


Table 1. Jumpers on the S32K396 EVB

Jumper	Connection
J4	2-3
J6	2-3
J7	2-3
J28	2-3
J45	2-3
J67	Disconnect

4.2 Run Motor with FreeMASTER

Ensure that the 3-phase LV PWR board is powered with 24V.

1. Open FreeMASTER application **K396_PMSM_FOC.pmp**.
2. Refer to Figure 12 to configure the FreeMASTER.
3. Click the **GO** icon on the FreeMASTER, then observe if the variables on the FreeMASTER are being updated. If the variables are being updated, the connection has been established successfully. Otherwise repeat step 1.

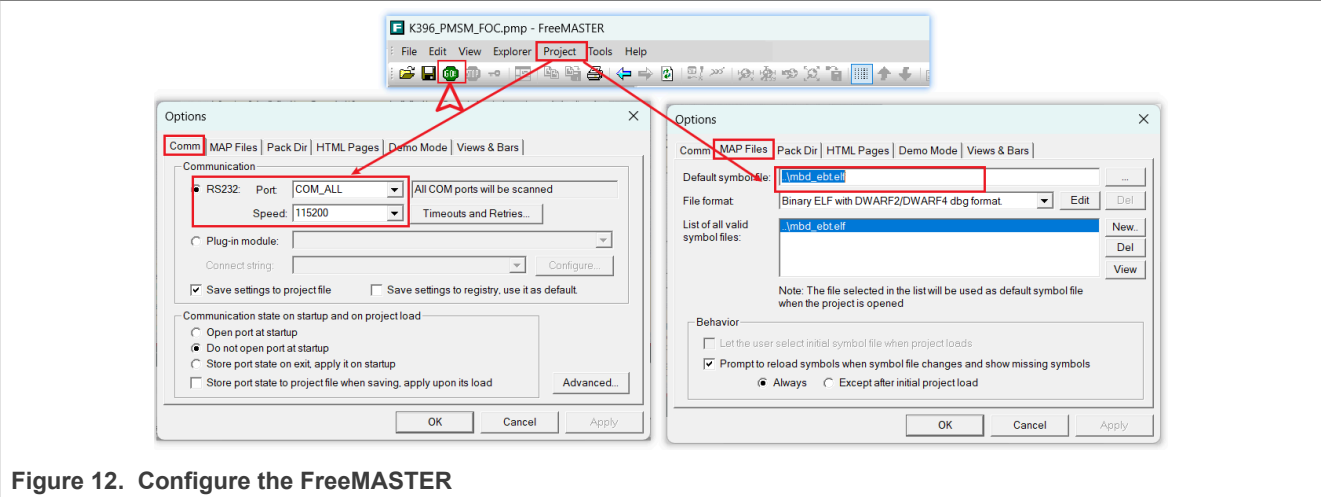


Figure 12. Configure the FreeMASTER

4. Click the **ON/OFF** button on the **App Control** page to run the motor.

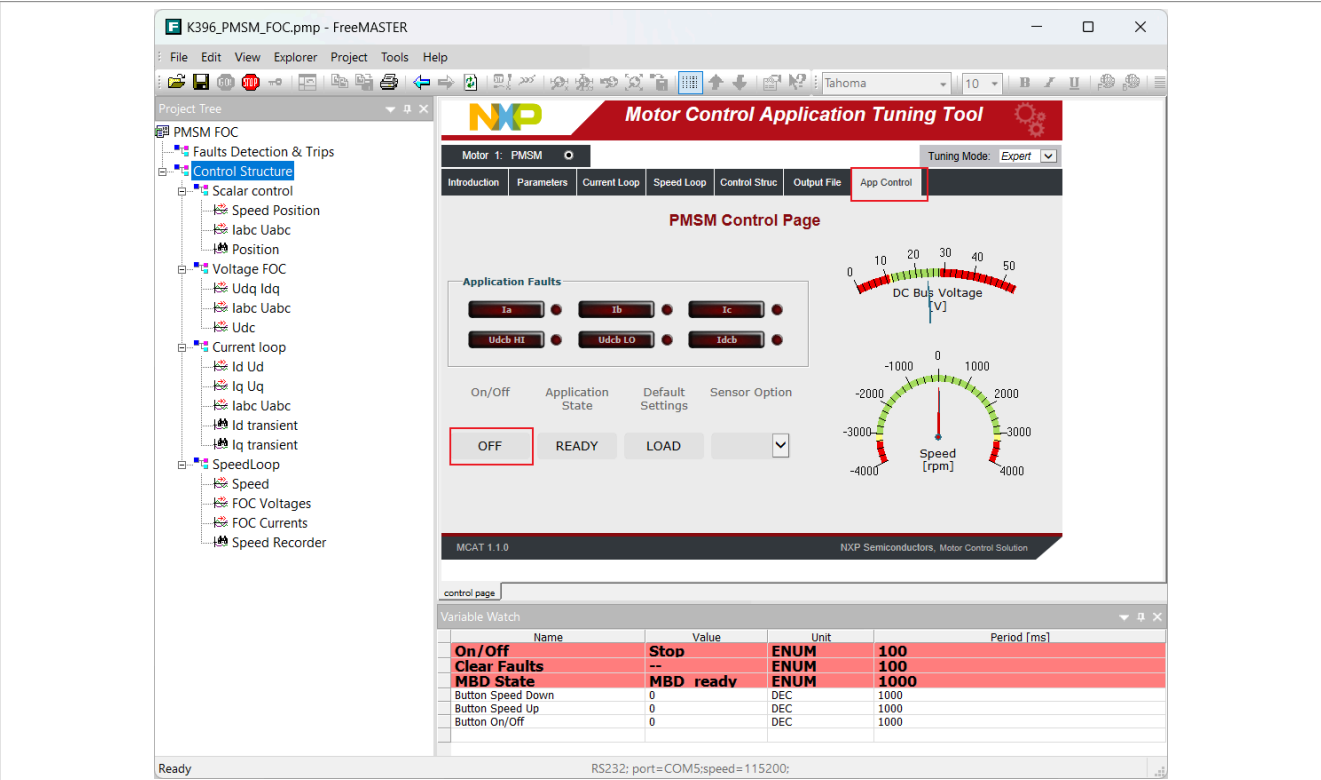


Figure 13. Control motor with the FreeMASTER

5. Click the speedometer pointer on this page, to set a required speed. The motor accelerates to track the required speed.

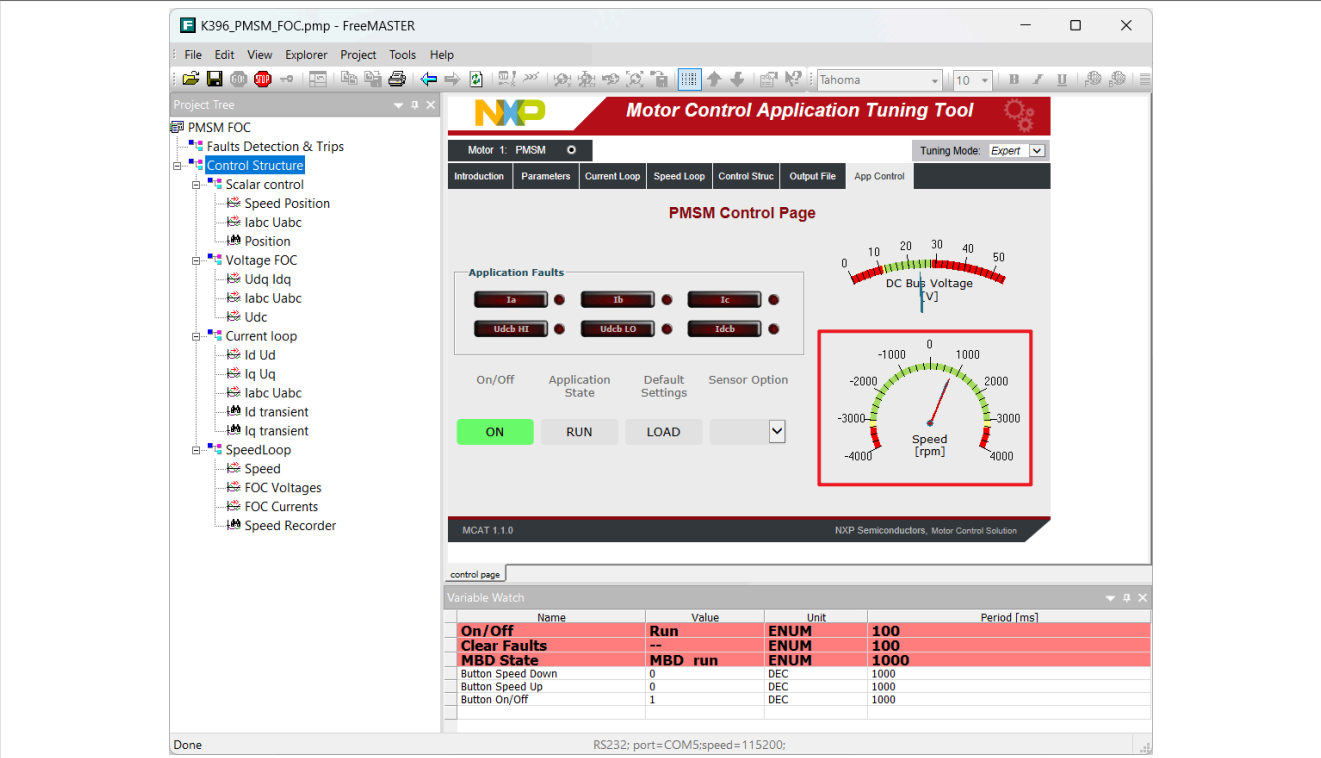


Figure 14. Set speed via FreeMASTER

6. Also, there are other control modes can be select in pane **Control Struc**. Select other modes when control state is **OFF**, and modify the required input parameters and press **Enter** to make it effective.

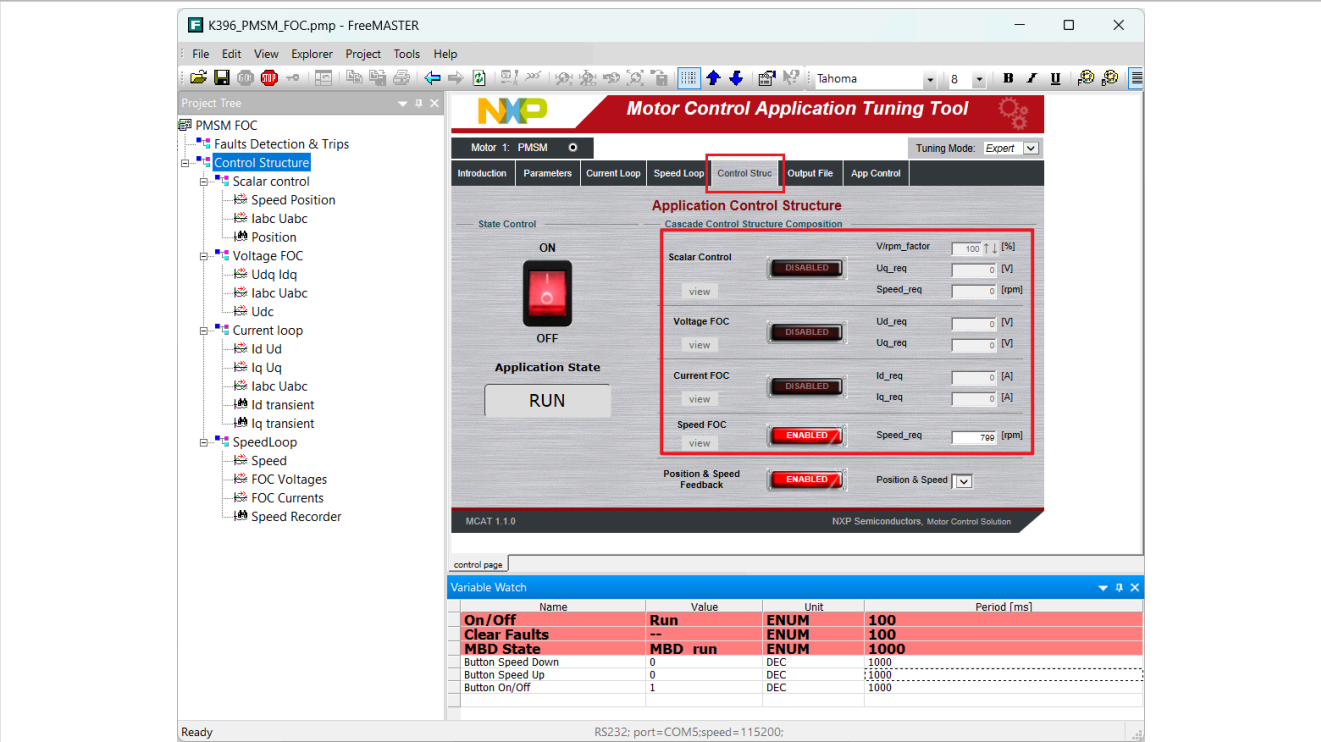


Figure 15. Control Mode and parameters set

5 PIL demo

Connect the hardware following steps in 4.1, and open the PIL model “pil_model_ebt.mdl”. Before running, we need to link the local project source paths of the hardware model first. Then run the PIL simulation demo with one click. Below are the detailed steps.

First, please check the configuration settings.

1. Click “Set Path” in the MATLAB menu, select “Add Folder” to add the folder “data_dictionary” and “modules” under the project root path, and save changes. This step ensures that the built model can successfully find the dependency files and reference modules.

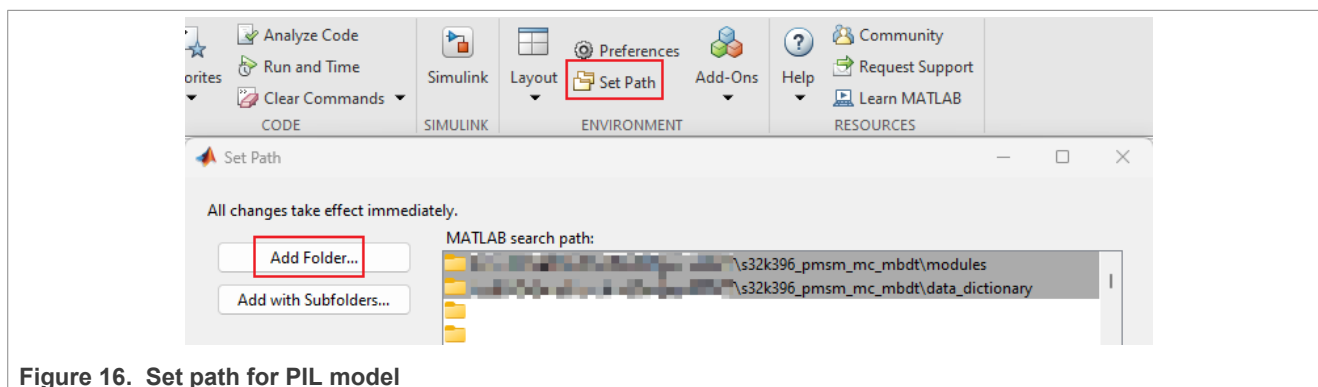


Figure 16. Set path for PIL model

2. Open data dictionary “MCAT_Parameters.sldd” and modify the parameter values as follow: MBD_ALIGN_DURATION set as 30, MBD_SPEED_UP_CNT set as 2, MBD_SPEED_DOWN_CNT set as 2, MBD_APP_OFF_CNT set as 2, MBD_CALIB_CNT set as 2. Save changes and close this data dictionary. Open file “PMSM_appconfig.h” and modify ALIGN_DURATION as 30.

3. The first time you open the PIL model “pil_model_ebt.mdl”, please click “Migrate Existing Project” in the pop-up window.

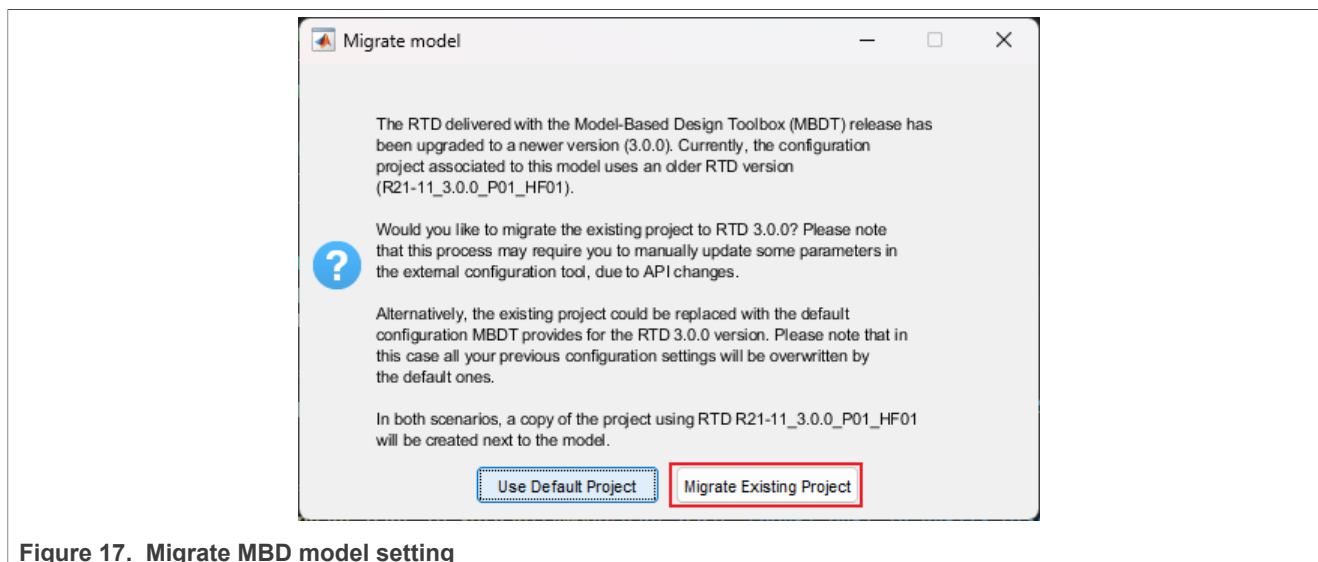


Figure 17. Migrate MBD model setting

4. Wait for the model to open, then right-click at the blank space of the model and open “Model Configuration Parameters”.

5. Change the EB configuration project at path “Hardware Implementation -> Hardware board settings -> Target hardware resources -> Hardware -> Configuration Template” to the project folder “pil_TresosProject”. If a new

window pops up, select “Overwrite the existing folder”. This step copies the pre-defined EB configuration files to the model configuration folder “pil_model_ebt_TresosProject”.

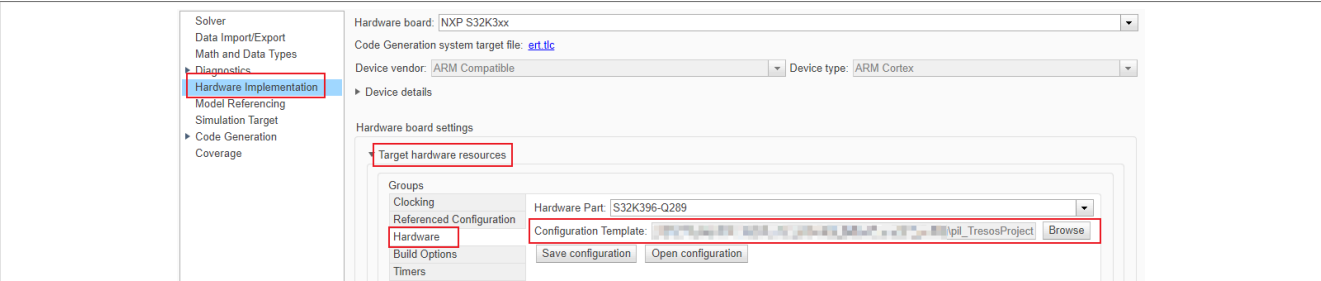


Figure 18. PIL hardware setting

6. Move to the next item “Tools Paths” in the setting. Modify the “EB Tresos Tools location” to the EB Tresos installation path at your computer, change the RTD location to the place where "SW32K3_RTD_4.4_R21-11_3.0.0_P01_HF01" version (with hotfix replaced) is installed. and select “Overwrite the existing folder”.

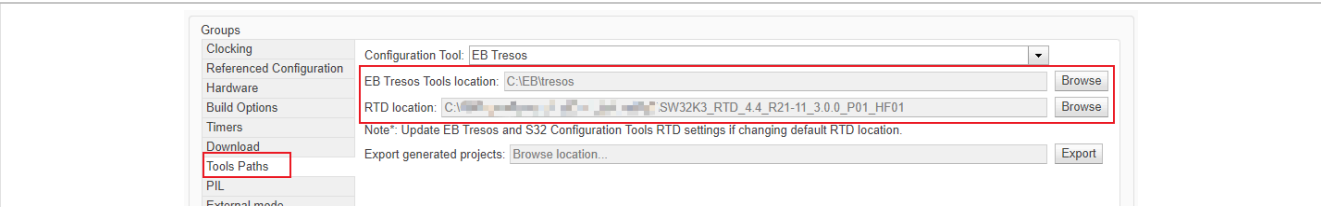


Figure 19. PIL tools paths setting

7. At the item “PIL”, click “Refresh” to generate the EB tresos configuration files. Then set the “Serial port” as identified in the “Device Manager” and set the other combo box the same as in [Figure 20](#).

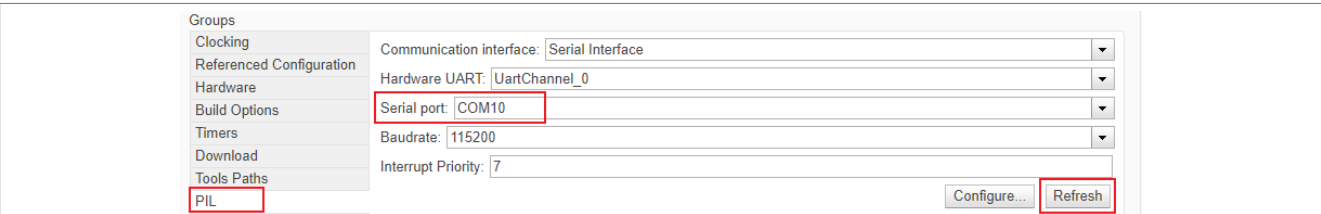


Figure 20. PIL communication setting

8. Apply all the changes and close the model.

Now it's ready to build the model. Open model “pil_top.mdl”.

9. Disconnect the USB cable first.

10. Open “SIL/PIL” in APPS. Check that the mode is set as “SIL/PIL Only” and the system under test is “Model blocks in SIL/PIL m...”.

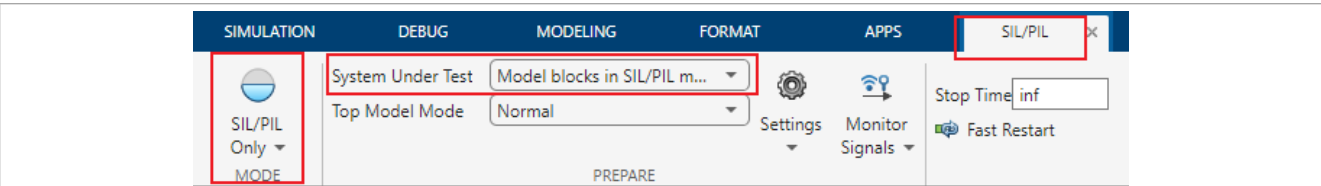


Figure 21. PIL application setting

11. Click “Run SIL/PIL” and wait for Simulink to build the model.

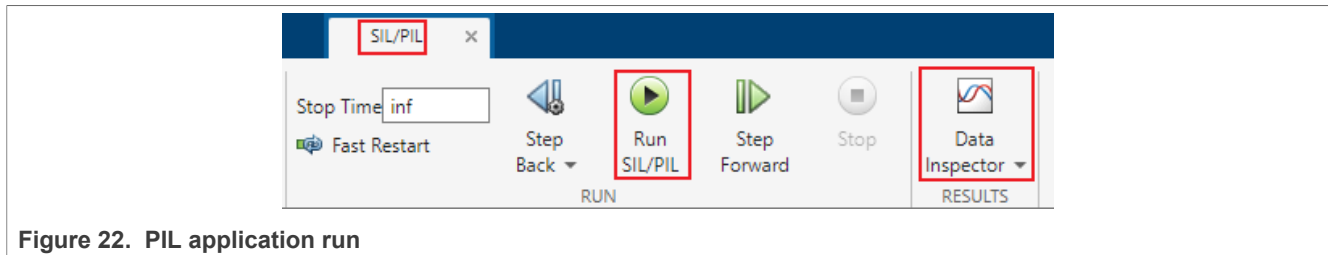


Figure 22. PIL application run

12. Download the debug file with PE micro debugger.

13. Connect the USB cable again after download starts and before download finishes.

When successfully downloaded, the Simulink time bar starts to count. Open “Data Inspector” to monitor the logged signals in the model. Then use buttons to control the motor running status.

1. Select the data to be observed in the left area and draw it in the right graphics area.

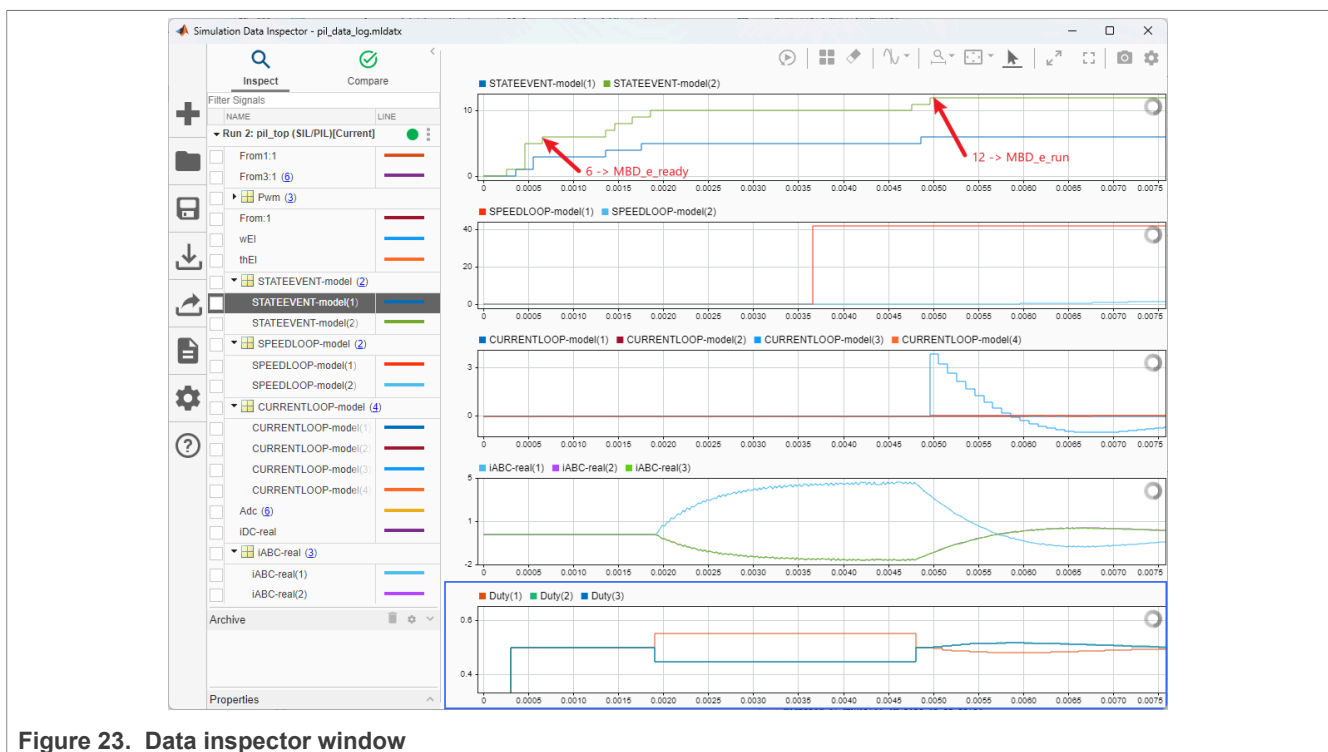
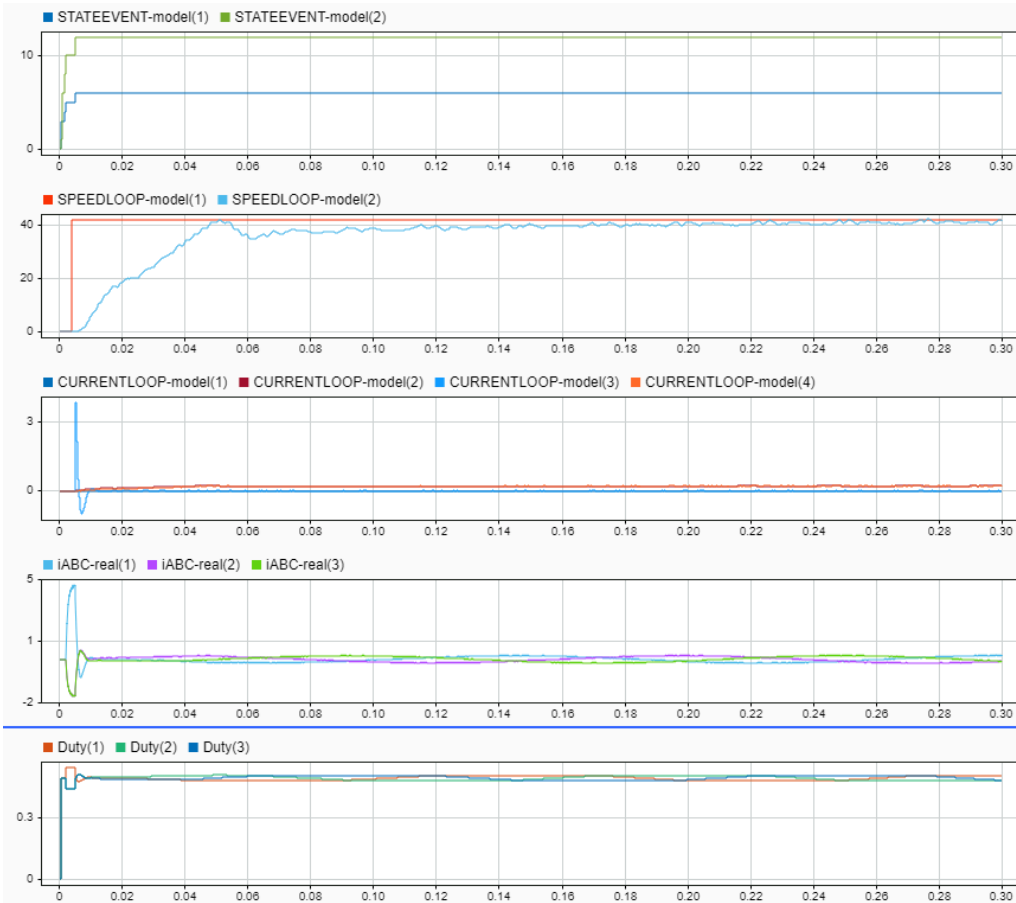


Figure 23. Data inspector window

1. When the logged signal STATEEVENT-model(2) reaches 6, which means the state machine event “MBD_e_ready” occurs, press and hold the SW5 button for a while until STATEEVENT-model(2) becomes 7.
2. Then wait a moment for the state machine to complete the initialization process, and the signal STATEEVENT-model(2) will eventually become 12.
3. Press and hold the SW8 until the logged signal SPEEDLOOP-model(1) steps from 0 to 41.9. It give an input speed demand of 41.9 rad/s.

4. Wait for the feedback speed to track the demand speed. Click “Stop” in Figure22 to end this simulation process or directly set a “Stop Time” in Figure22.



Speed tracking process

6 Revision history

Table 2. Revision history

Document ID	Release date	Description
UG10136 v.1.0	23 May 2024	Initial release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at <https://www.nxp.com/profile/terms>, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Suitability for use in automotive applications — This NXP product has been qualified for use in automotive applications. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as "Critical Applications"), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys' fees) that NXP may incur related to customer's incorporation of any product in a Critical Application.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

MATLAB — is a registered trademark of The MathWorks, Inc.

Tables

Tab. 1.	Jumpers on the S32K396 EVB	9	Tab. 2.	Revision history	15
---------	----------------------------------	---	---------	------------------------	----

Figures

Fig. 1.	Add folder to MATLAB path	3	Fig. 13.	Control motor with the FreeMASTER	10
Fig. 2.	Open project	4	Fig. 14.	Set speed via FreeMASTER	11
Fig. 3.	Hardware settings	4	Fig. 15.	Control Mode and parameters set	11
Fig. 4.	Change the configuration template	5	Fig. 16.	Set path for PIL model	12
Fig. 5.	Change tool path and RTD location	5	Fig. 17.	Migrate MBD model setting	12
Fig. 6.	Build project	6	Fig. 18.	PIL hardware setting	13
Fig. 7.	Build project	6	Fig. 19.	PIL tools paths setting	13
Fig. 8.	Configure the download tool	7	Fig. 20.	PIL communication setting	13
Fig. 9.	Download finish dialog box	7	Fig. 21.	PIL application setting	13
Fig. 10.	Demo connection	8	Fig. 22.	PIL application run	14
Fig. 11.	S32K396 EVB	9	Fig. 23.	Data inspector window	14
Fig. 12.	Configure the FreeMASTER	10			

Contents

1 Introduction 2

2 Hardware, Software, and Tools 2

3 Build demo 2

3.1 Open model and setup configuration 2

3.2 Build the project 6

3.3 Download elf file into board 6

3.3.1 Download via Multilink PE micro debugger 6

3.3.2 Download via Lauterbach 8

4 Run demo 8

4.1 Connect hardware 8

4.2 Run Motor with FreeMASTER 9

5 PIL demo 12

6 Revision history 15

Legal information 16

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.