Document Number: KMS100UG
Rev. 0, 02/2016

Kinetis Motor Suite User's Guide

1 Introduction

Kinetis Motor Suite (KMS) is a bundled hardware and software solution aimed at enabling rapid configuration of motor drive systems and accelerating application development.

KMS includes firmware targeting the Kinetis V (KV) series of microcontrollers (MCUs) and an intuitive PC-based graphical user interface. It supports field oriented velocity and position control of three phase permanent magnet and brushless DC motors.

This document describes the main features of KMS with an emphasis on the KMS PC GUI and its usage during the design cycle.

The companion API Reference Manual and Kinetis V series Reference Manuals provide greater detail on the relevant embedded firmware and hardware, respectively. The KMS Lab Guide provides an in-depth hands-on introduction to utilization of the features described in this document and in the API Reference Manual.

Contents

2 Components

KMS is implemented as several components including hardware and software. The hardware consists of KMS enabled MCUs, which are a subset of the Kinetis V series of MCUs, and development platforms which include the three phase inverter, gate drivers and feedback circuits to drive the motor. The software consists of firmware and a PC-side graphical user interface (GUI) application.

2.1 Hardware

2.1.1 Development platforms

KMS is supported on the following development platforms containing a KMS-enabled MCU:

- FRDM-KV31F: Development Platform for Kinetis KV3x Family MCUs
- TWR-KV31F120M: Kinetis KV3x Family Tower System Module
- HVP-KV31F120M: High-Voltage Development Platform Controller Card

For evaluation purposes, a corresponding motor drive platform is required:

- FRDM-MC-LVPMSM: Low-Voltage, 3-Phase PMSM Motor Control Development Platform
- TWR-MC-LV3PH: Low-Voltage, 3-Phase Motor Control Tower System Module
- HVP-MC3PH: High-Voltage Development Platform

The KMS GUI contains the hardware-dependent configuration parameters for each evaluation platform. These can be changed for hardware custom-designed for the user’s end application.

- Refer to the application note Adapting KMS for Custom Hardware for information on how to transition to end-application hardware.

2.1.2 MCUs

KMS is supported across the KV3x MCU portfolio, with KMS-enabled devices designated by an additional character at the end of a typical KV3x part number. A typical Kinetis V series part number is: MKV31F256VLH12. The KMS variant of this MCU would be: MKV31F256VLH12P, where the P indicates KMS support for permanent magnet/brushless DC motors.

- Refer to www.nxp.com/vseries for the list of parts supported by KMS.
- Refer to Section 11, "Resources utilized"for enumeration of the MCU resources KMS requires.
- Refer to Kinetis V series MCU documentation for exact hardware specifications.
- Refer to the application note Adapting KMS for Custom Hardware for information on how to transition to end-application hardware.

2.2 Software

The KMS PC-side GUI application provides a step-by-step framework to identify, configure and test the motor and application performance. It communicates with a KMS-enabled MCU via a UART interface which is used to send and receive data from the embedded firmware. The GUI includes a Software

Oscilloscope, Watch Window, and Motion Sequence Builder to enable design of real-time motor control systems.

KMS motor control software consists of:

- factory programmed, proprietary firmware
- reference projects that are customized through usage of the KMS GUI and then modified in the user's preferred development environment

The preprogrammed, proprietary portion of KMS firmware, which consists of library function that are required to run KMS, is not intended to be readable by the end-user. It is restricted to a specific section of memory in the MCU and marked as execute-only. The existence (or not) of this code on an MCU determines whether or not it is KMS-enabled.

WARNING

KMS proprietary code is not available for reloading onto the MCU by the end user, so **MASS ERASE OF THE DEVICE MUST BE AVOIDED**.

This has implications for the debugging tools that may be utilized. Refer to the KMS Release Notes for the latest information on debugging tool compatibility.

In addition, because unsecuring flash involves mass erase, make sure to only secure flash in devices with final firmware versions. Devices used for development should not be secured to minimize risk of mass erase.

The reference project is generated according to the user's configuration in the KMS GUI. It is provided as a combination of pre-compiled libraries and open source code which can be manually edited and compiled in supported development environments.

- Refer to the KMS Release Notes for KMS software prerequisites required by the relevant KMS software release.
- Refer to the KMS Lab Guide for instructions on locating KMS software prerequisites and a hands-on introduction to both the PC GUI and firmware.
- Refer to the KMS API Reference Manual for information regarding the algorithms implemented in firmware.

3 Installation

This section describes the process of installing KMS as well as the components that KMS installs.

3.1 Procedure

1. To begin working with KMS, download and run the Kinetis Motor Suite 1.0.0 Installer.exe file. This file can be found at nxp.com/kms.

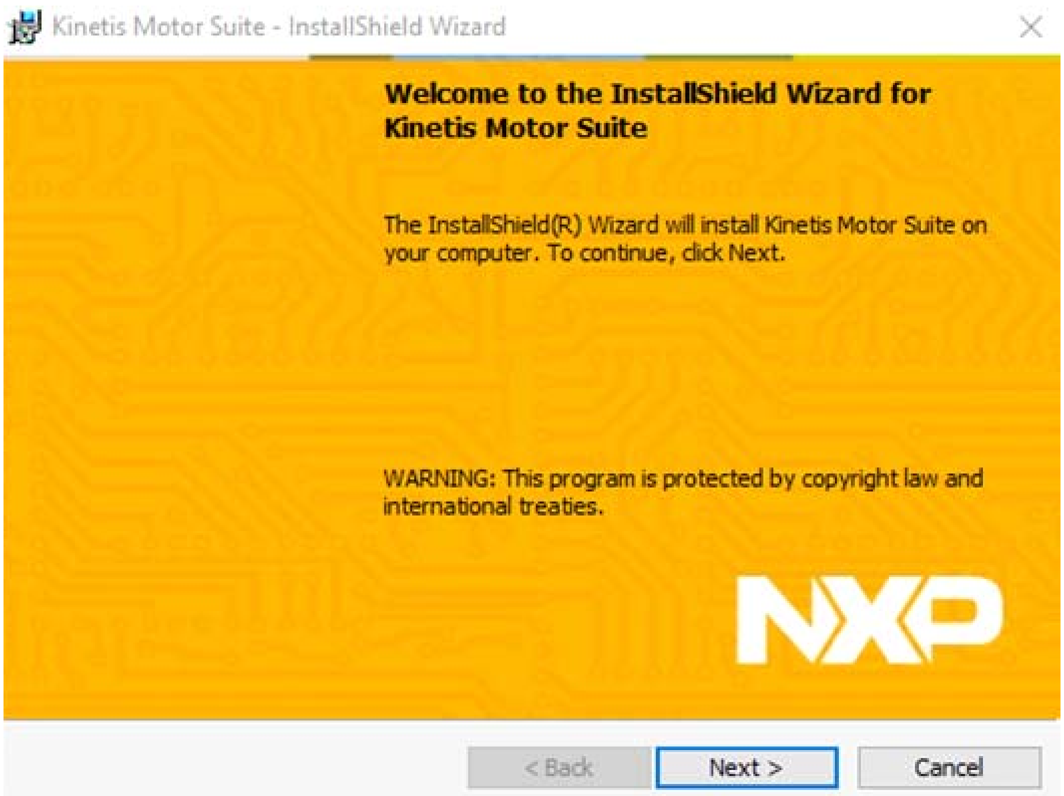


Figure 1. Installation welcome screen

2. Read and accept terms of the KMS end-user license agreement

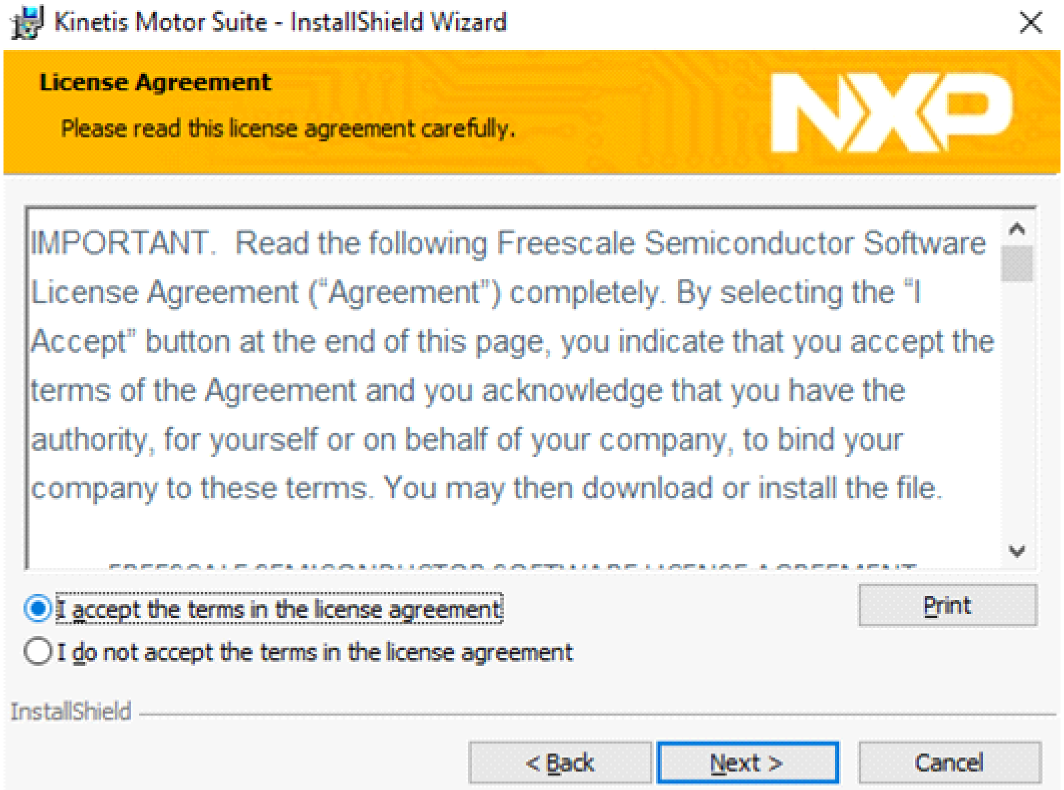


Figure 2. KMS license agreement

3. Select installation folder. The default is C:\NXP\KMS_<version>.

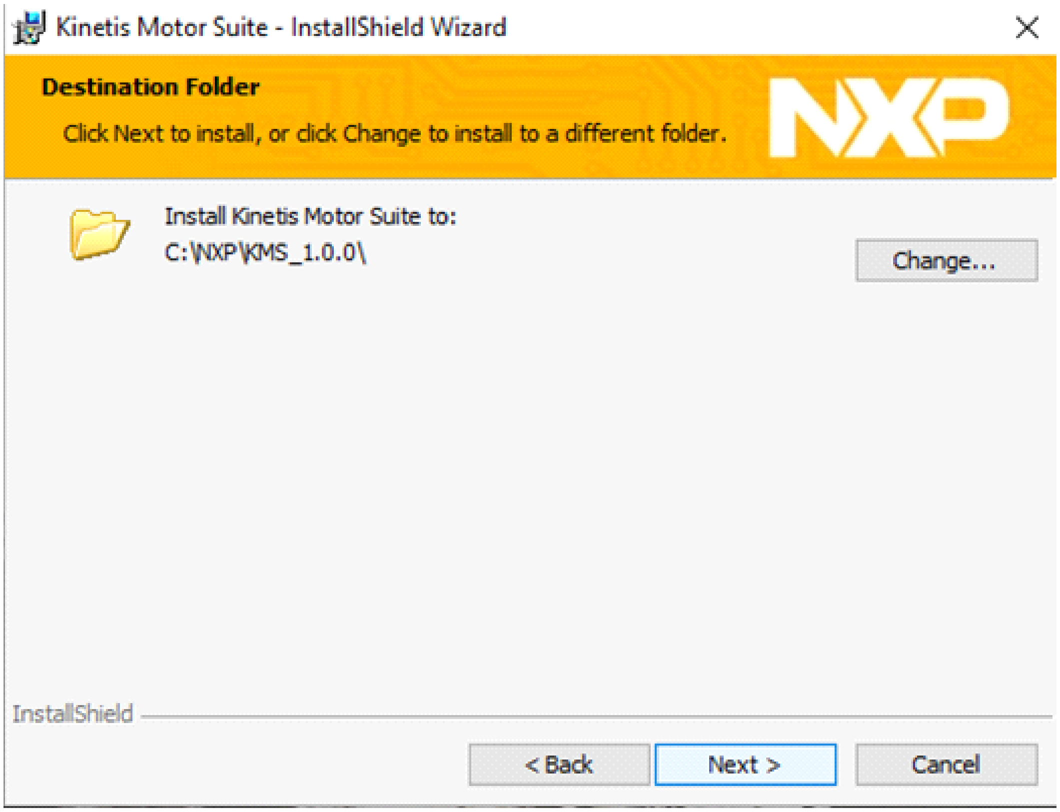


Figure 3. Choose install folder

4. Click to begin installation.

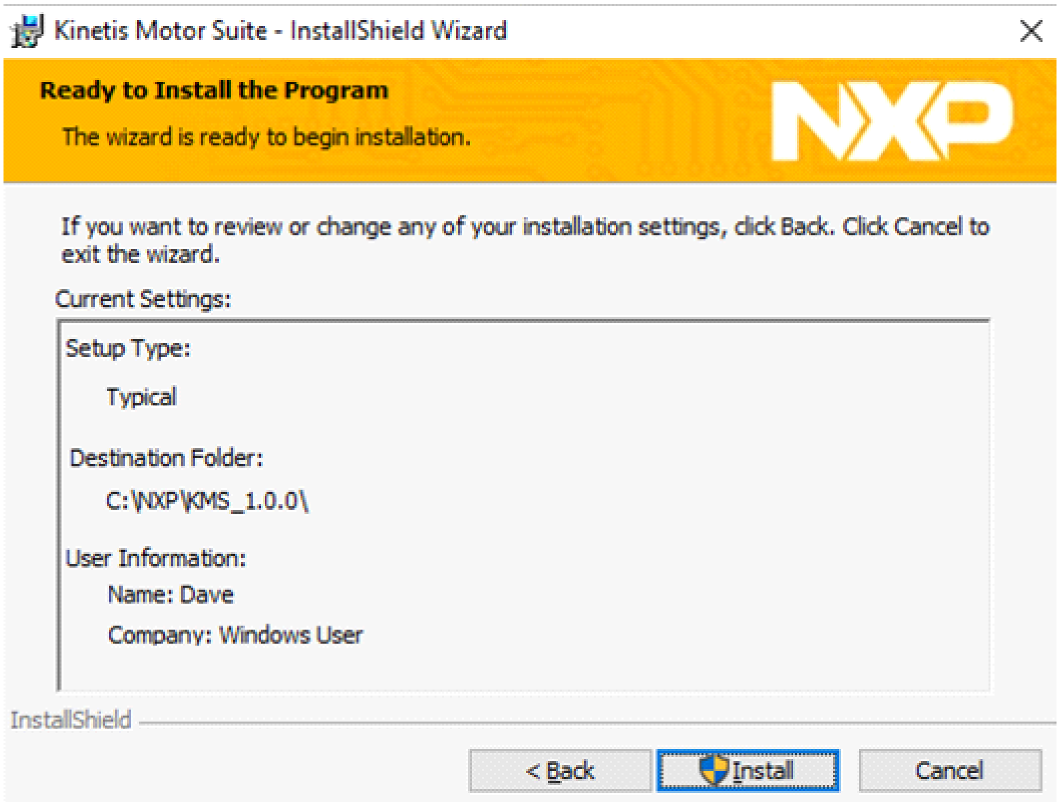


Figure 4. Begin install

5. Finish KMS installation

Figure 5. Click to finish install

6. KMS leverages the 3rd party tool Graphviz for generation of state diagrams for visual verification of motion sequences. The KMS installer manages Graphviz installer. Installation of Graphviz is optional and does not affect core KMS functionality. If you have previously installed KMS and Graphviz, the Graphviz installer asks about repairing or removing the software and thus your screens may not match the below figures.

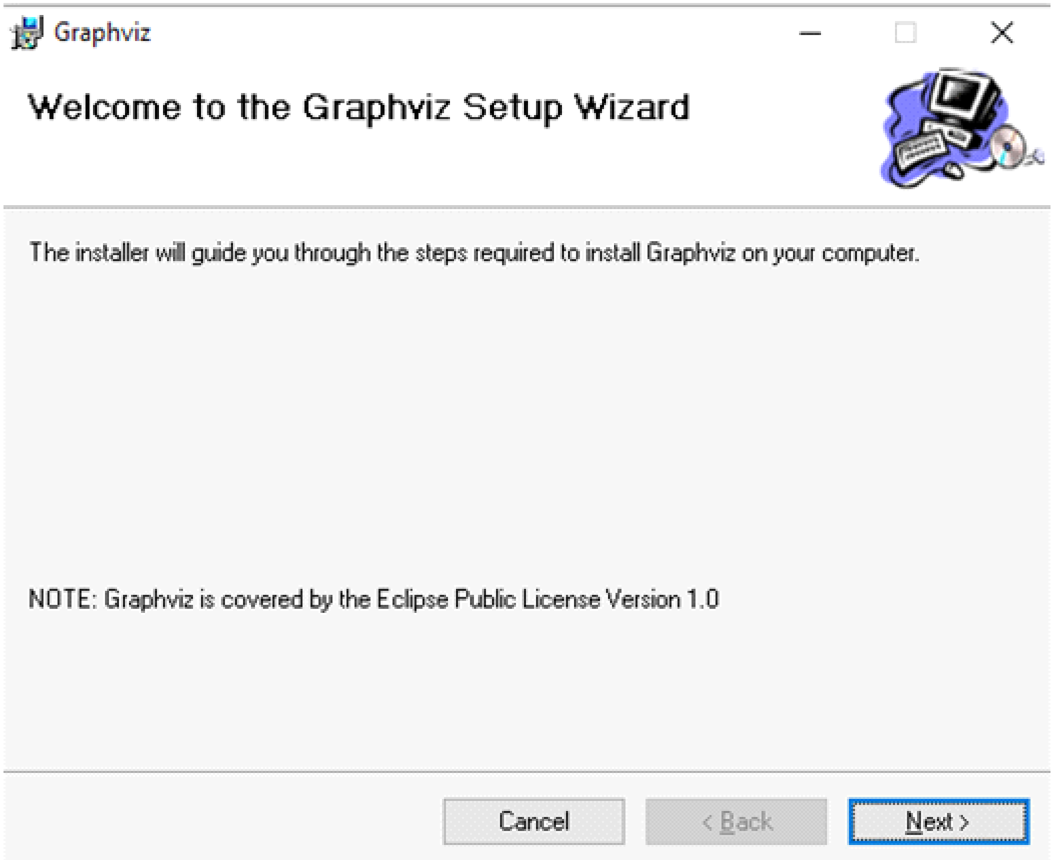


Figure 6. Graphviz installer welcome screen

7. Select Graphviz location

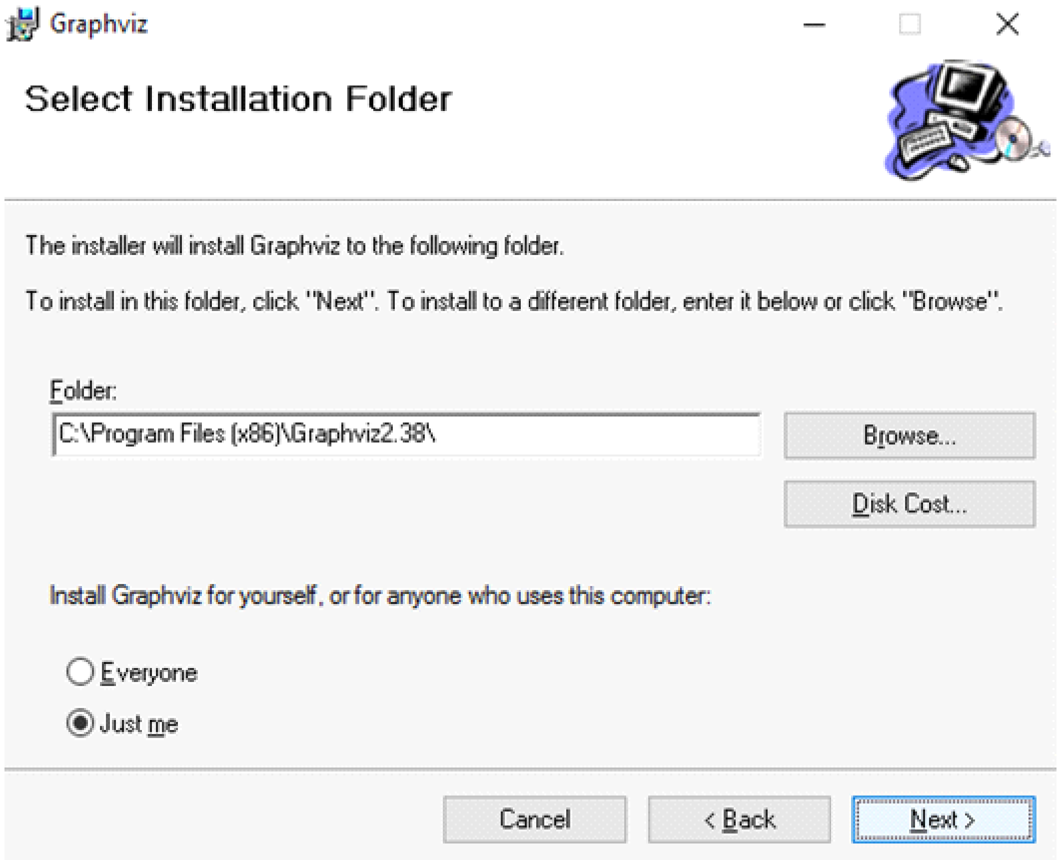


Figure 7. Graphviz location

8. Confirm and start installation.

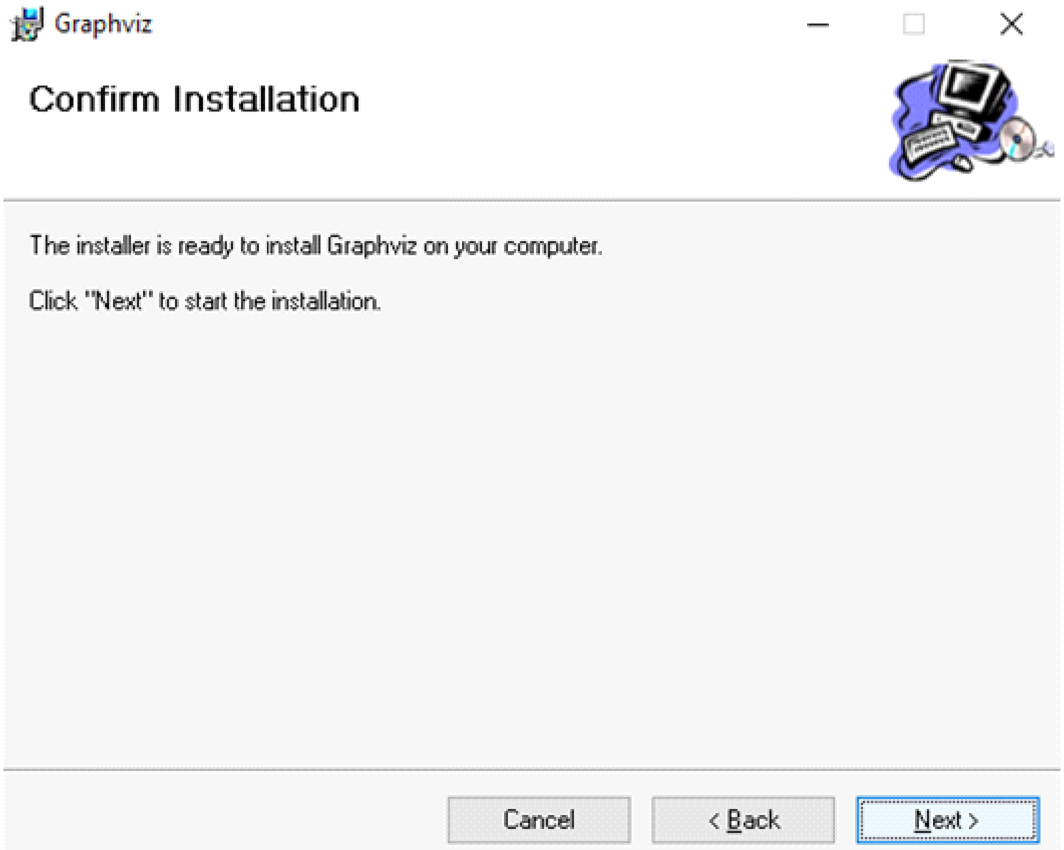


Figure 8. Begin Graphviz installation

9. Conclude Graphviz installation

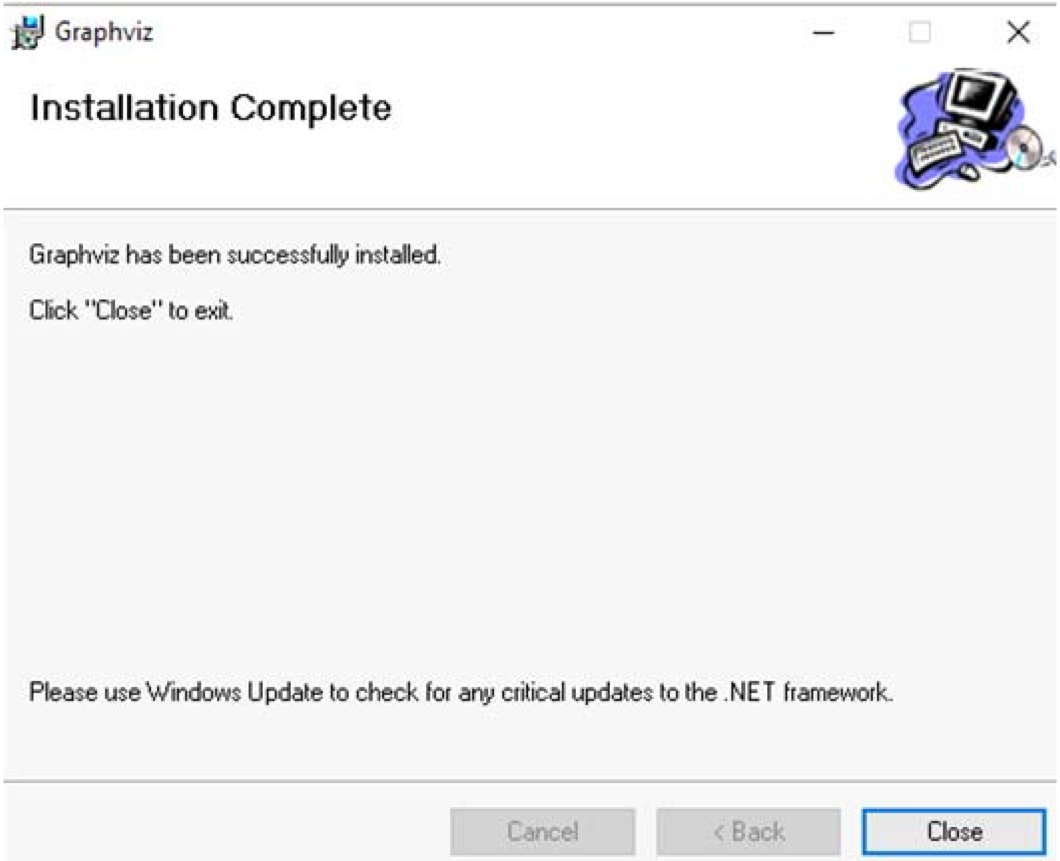


Figure 9. Click to finish Graphviz installer

3.2 Result of installation

At the conclusion of KMS installation, regardless of whether you have allowed Graphviz installation as well, your system has two primary locations for KMS files:

- C:\NXP\KMS_<version> (or your custom location specified during installation)
- C:\Users\<username>\Documents\KMS_<version>

The former contains all KMS system data whereas the latter contains all user information.

The folder hierarchy and contents for KMS system data is shown in Table 1. Users should not need to access these files and are in fact encouraged to avoid them.

Table 1. System data folder structure

Installation parent directory	KMS folder	Folders	Contents
C:\NXP	KMS_<version>	en-us	UI text strings by language
		IronPythonLib	KMS scripting engine underpinnings
		ReferenceProjects	Firmware, UI definition, and script files for each KMS reference project variant by platform, control type, development environment, and version
		(N/A)	Core UI code

By contrast, the user data location should be regularly accessed by the user. The folder hierarchy and contents for KMS user data is shown in Table 2.

Table 2. User data folder structure

Installation parent directory	KMS folder	Folders	Contents/purpose
C:\Users\<username>\Documents	KMS_<version>	Archive	Two days' worth of KMS trace logs for troubleshooting and support purposes
		Config	File for defining certain KMS application options. Not recommended for manual editing.
		Documents	KMS product documentation (including this document) that is available from KMS GUI
		Motion Sequence Builder	Example motion sequences and folders for storing user motion sequences & diagrams
		SavedCmdLineOutputs	Default save location for the output of KMS utilizing the command line to compile and download updated firmware to the MCU
		SavedCSV	Default save location for comma separated data generated by the Software Oscilloscope
		SavedPlots	Default save location for plot images generated by the Software Oscilloscope
		SavedProjects	Default placement location for KMS projects. KMS project structure is described in Section 4, "System configuration".
		Setting	Brief set of user preferences. Not recommended for manual editing.
		Tmp	Temporary folder
		Traces	Location of a trace log for the current KMS session. After the session concludes, the file is transferred to the Archive.

The folder level is pictured in Figure 10.

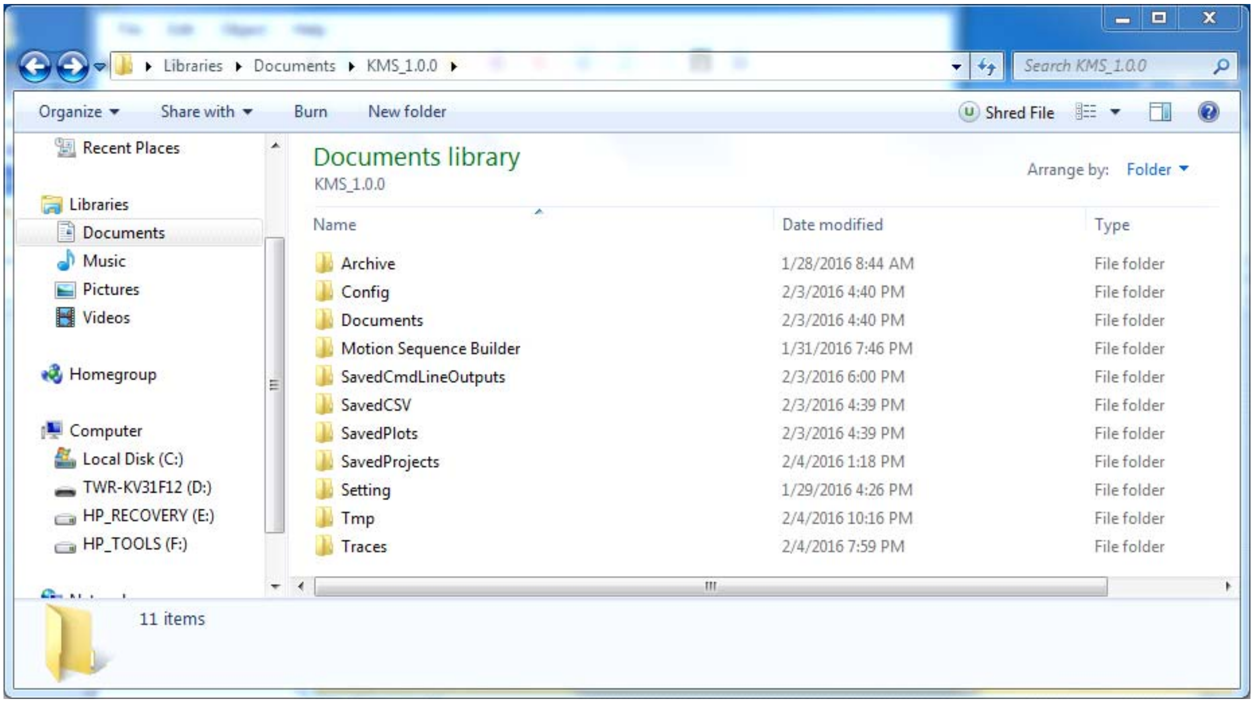


Figure 10. KMS user data folders

4 System configuration

4.1 New vs. Open

1. Installation provides a desktop icon for launching KMS.

Figure 11. KMS desktop icon

2. Double-click to launch. The KMS splash screen appears then gives way to the KMS landing page.

Figure 12. KMS splash screen

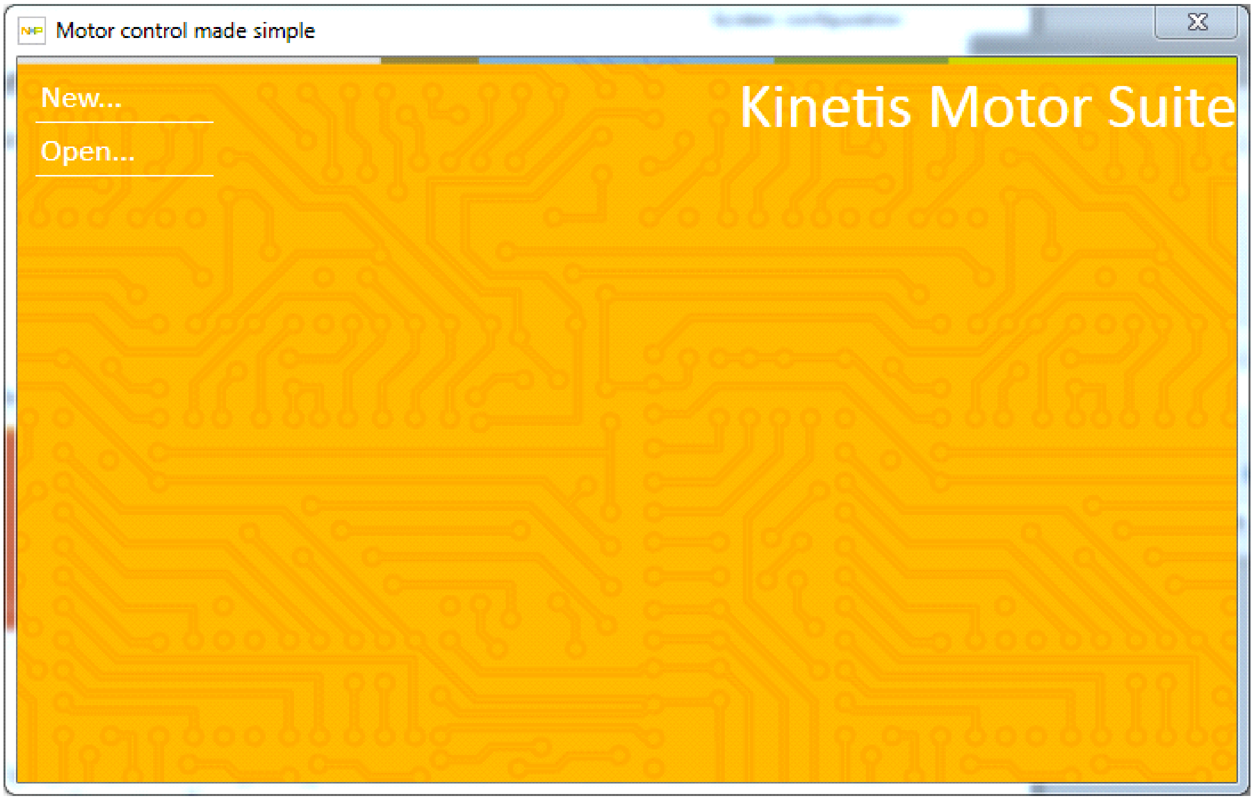


Figure 13. KMS landing page

The user has two options:

- New...
- Open...

New... starts a KMS project from scratch, whereas Open... allows the user to select a previously started KMS project.

4.1.1 New...

Upon starting a new KMS project and before opening the main KMS GUI window, the KMS GUI leads the user through the system configuration that is required to properly set up the KMS GUI and firmware reference project.

At any time, you may click the restart button to begin the configuration process again.

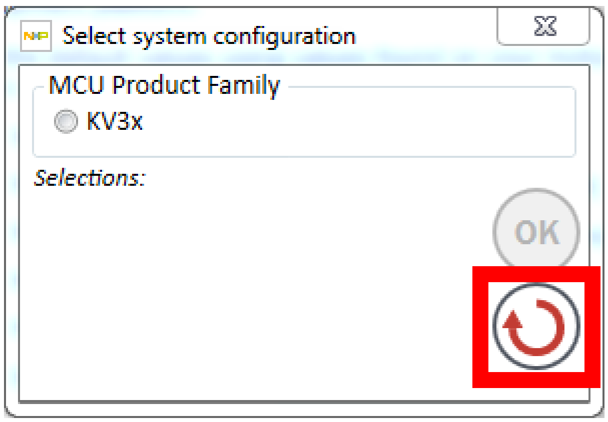


Figure 14. Click to restart

1. Select the MCU Product Family (Figure 15).

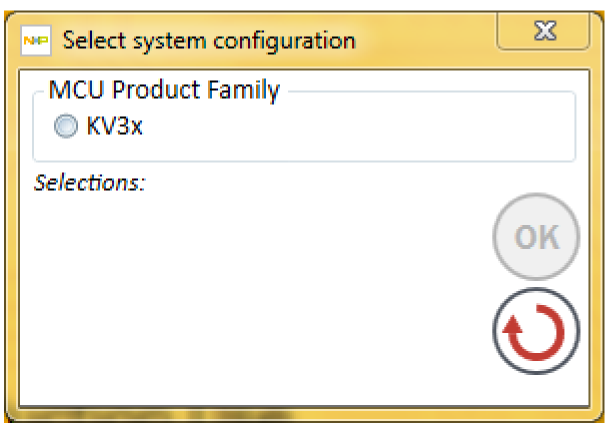


Figure 15. Select MCU product family

— Refer to KMS Release Notes for the supported MCUs.

2. Select the development platform being utilized (Figure 16).

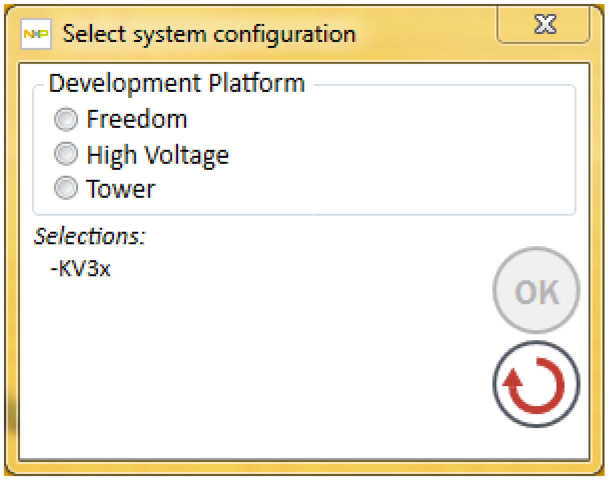


Figure 16. Select development platform

— Refer to Section 2.1.1, "Development platforms" for part numbers of supported development platform hardware.

3. Select the type of control that will be used in the application (Figure 17)

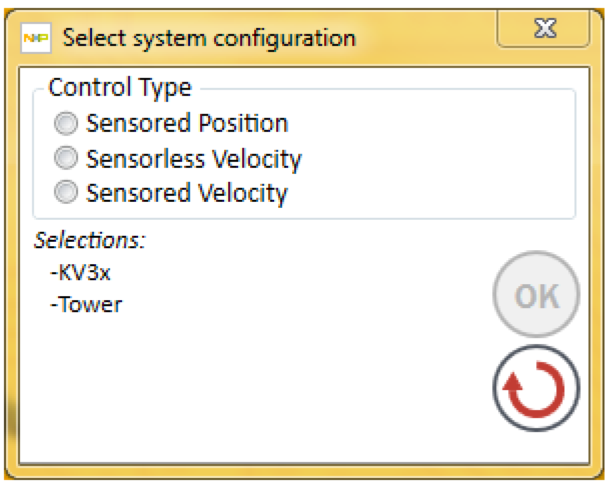


Figure 17. Select control type

Table 3. Control type options

Option	Description	Example application
Sensored Position	Motor spins from point to point and is able to hold steady upon reaching the desired point. Requires a sensor to precisely determine motor angular position.	Security camera
Sensorless Velocity	Motor spins at one or various speed and the commanded speed must be maintained. Extremely low speed operation and ability to change directions are not the most critical performance criteria, so operation without a sensor for angular position is possible through usage of the motor's electrical signals. (The strength of the relevant signals is proportional to motor speed, hence the challenge of low speed operation).	Ceiling fan
Sensored Velocity	Motor spins at one or various speed and the commanded speed must be maintained. Extremely low speed operation and/or the ability to change motor direction under load are critical, so sensor is required to precisely determine motor angular position.	Industrial sewing machine

4. Select the type of motor that will be used in the application (Figure 18)

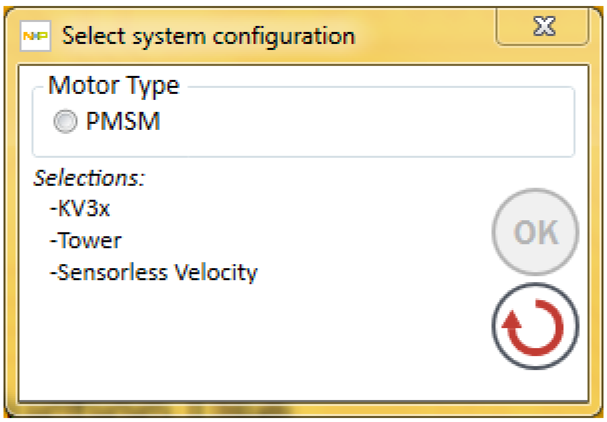


Figure 18. Select motor type

Table 4. Motor type options

Option	Description
PMSM	Acronym for permanent magnet synchronous machine. KMS supports both true PMSM and brushless DC (BLDC) motors; KMS does not differentiate here because both are based on permanent magnets and are driven sinusoidally by KMS (the typical distinction between PMSM and brushless DC motors is sinusoidal vs. simpler, less efficient trapezoidal operation).

5. Select the development environment (IDE) that will be used to create the application code (Figure 19)

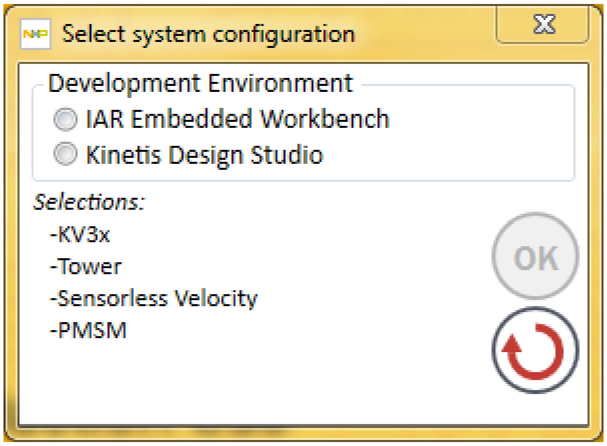


Figure 19. Select development environment

— Refer to KMS Release Notes for version information of supported IDEs.

6. If available, select the version of the KMS Motor Observer reference project that you want to use. If you only have access to one version of the reference project you have specified by virtue of your previous system configuration selections, this version is automatically selected and displayed.

   For instance, in Figure 20, the version number 1.0.0.8 appears automatically if the installed KMS contains only one Motor Observer reference project that fits the criteria:

   – KV3x
   – Tower
   – Sensorless Velocity
   – PMSM
   – Kinetis Design Studio

You may have access to multiple versions as KMS evolves and publishes enhancements to existing reference projects.

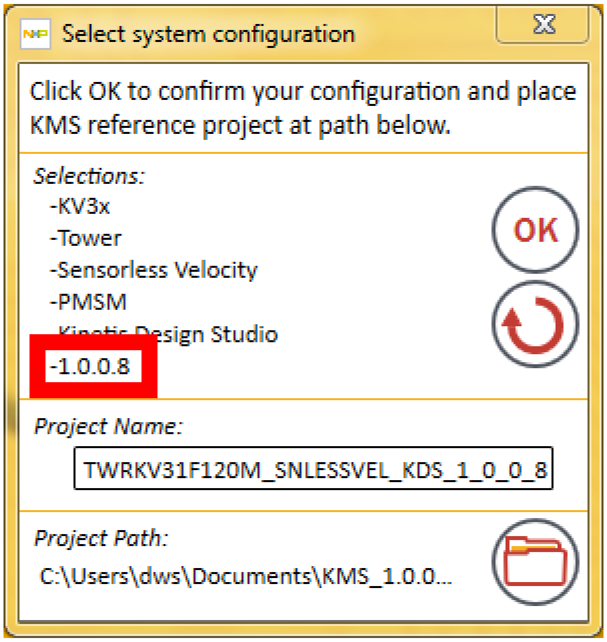


Figure 20. Reference project version number

7. Confirm the system configuration by clicking OK. This step copies from the KMS system data location to the KMS user data location a version of the Motor Observer reference project specified by your system configuration selections. You may change the parent directory in which you place this KMS project (Figure 21). You may also change the name of the KMS project folder name (Figure 22).

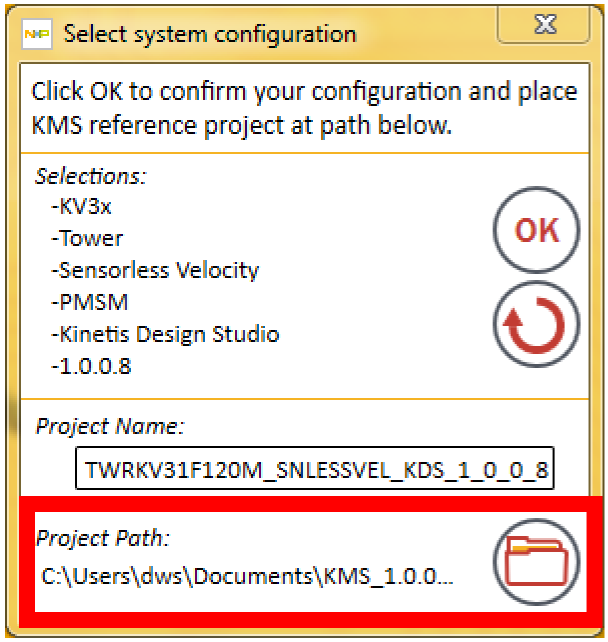


Figure 21. Change reference project parent directory

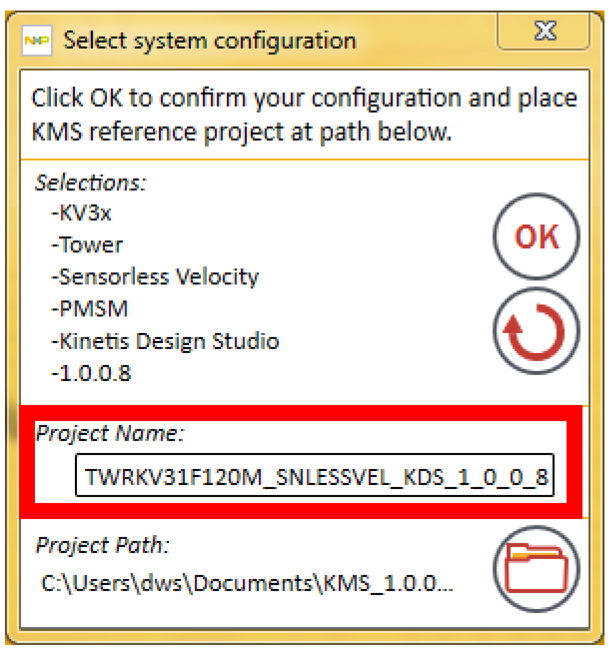


Figure 22. Change reference project folder name.

8. The result of this project creation is shown in Figure 23.

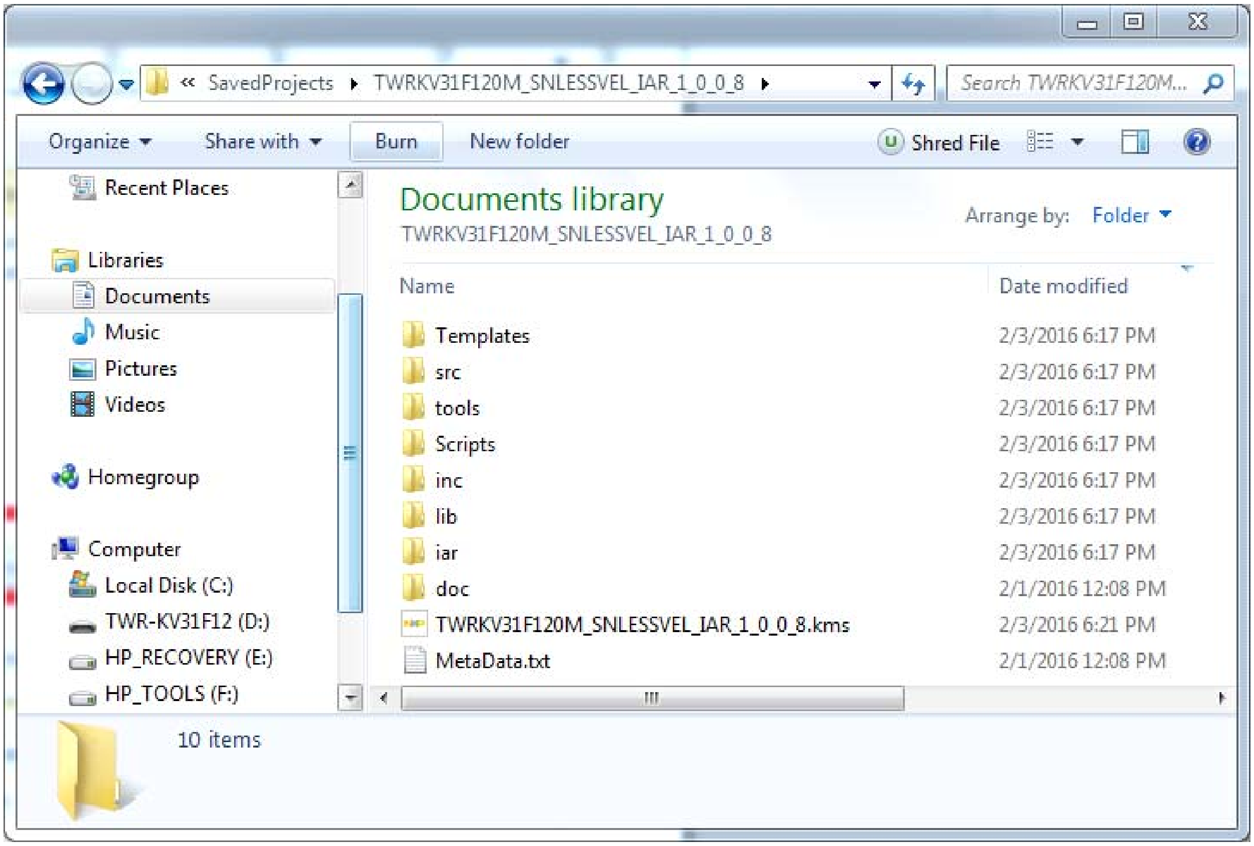


Figure 23. Example of KMS project structure

.A description of the folder hierarchy and contents for a KMS project is shown in Table 5.

Table 5. KMS project folder structure

Project path	Project name	Folders	Contents/purpose
C:\Users\<username>\Documents\Saved Projects	<platform>_<control type>_<IDE>_<version>	Templates	Closed source file that places KMS GUI elements for the chosen configuration
		src	Open source .c files for underlying KMS firmware reference project
		tools	Open source scripts for editing compile and download from KMS behavior
		Scripts	Closed source motor & motion control scripts used to adapt KMS to user motor and system
		inc	Open source .h files for underlying KMS firmware reference project
		lib	Precompiled library of KMS firmware reference project blocks
		iar/kds	KMS firmware reference project files for selected development environment
		doc	Any required documents. Empty by default.
		(N/A)	.kms file: closed source KMS GUI data. Doubleclick to open KMS GUI using this data.
		(N/A)	Metadata.txt: KMS reference project parameters (platform, development environment, control type, etc.).

9. After confirming your system configuring and placing the reference project, KMS begins the process of setting up the KMS GUI and MCU connection.

4.1.2 Open

Upon clicking to open an existing KMS project and before opening the main KMS GUI window, the KMS GUI displays a series of radio buttons showing the five most recent KMS projects, defined by their .kms files (see Table 5 for brief description of .kms file), as well as a radio button to allow the user to manually navigate to a different .kms file.

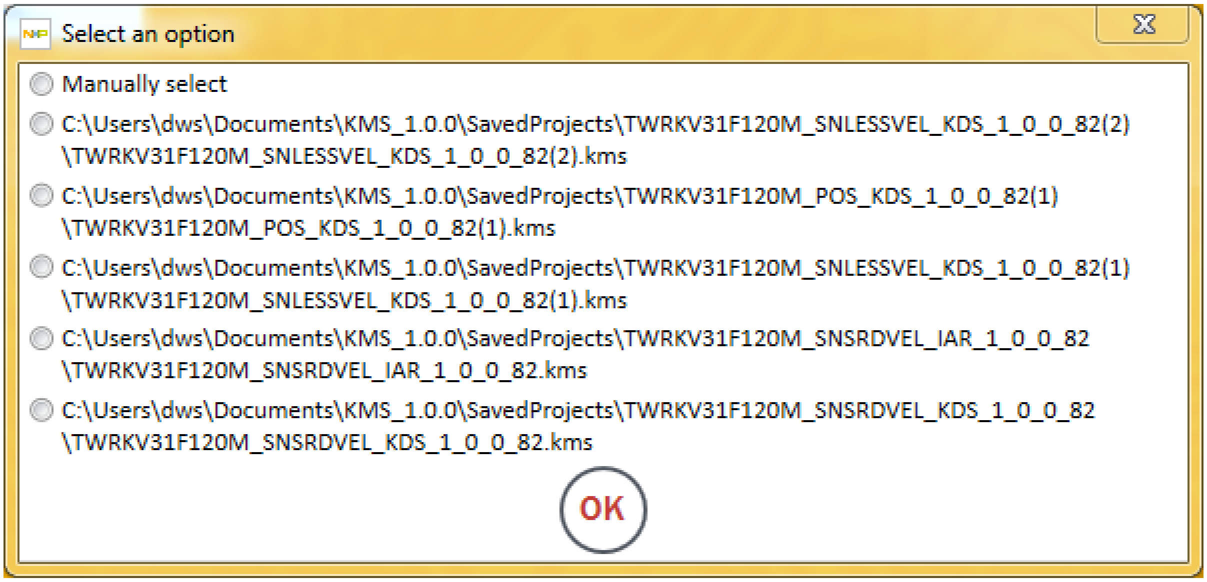


Figure 24. Click Open... to select or navigate to an existing KMS GUI project

Select an option, then click OK. If you choose “Manually select,” a typical Windows file finder dialog box appears.

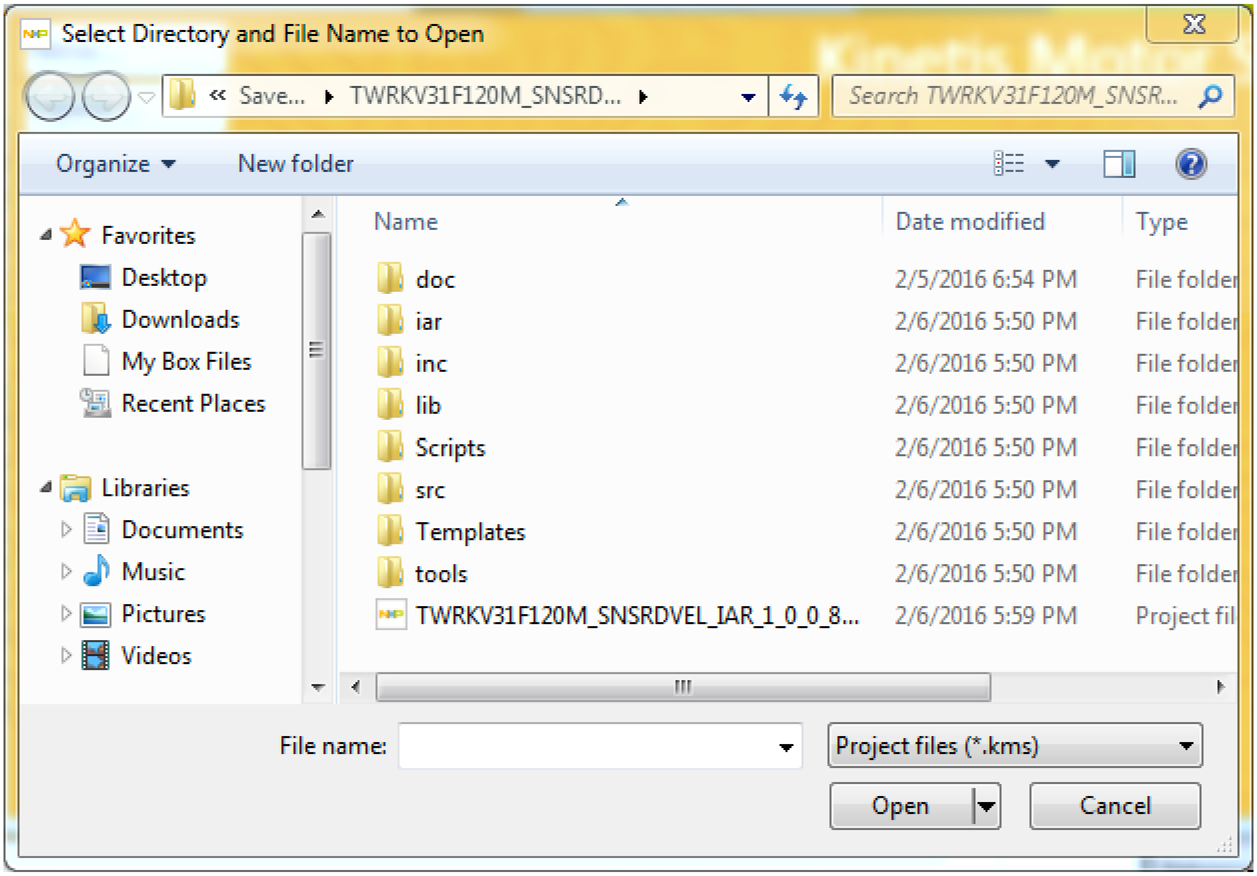


Figure 25. Manually select .kms file

After specifying the desired .kms file, whether via the recent files options or the manual selection, KMS begins the process of setting up the KMS GUI and MCU connection.

4.2 GUI-MCU connection

KMS relies on the ability to pass information from the GUI to the target MCU and vice-versa. For this to occur, KMS needs:

- awareness of certain tools being utilized
- a communication port
- consistency between the binary image on the MCU and the image in the selected KMS project

4.2.1 KSDK

KMS relies on the Kinetis SDK (KSDK) to handle hardware specifics. After specifying your desired project configuration, KMS searches for KSDK. If KMS cannot find KSDK on your machine, because it has not been installed, exists in a non-default location, or is one of several versions, KMS prompts you to define the location.

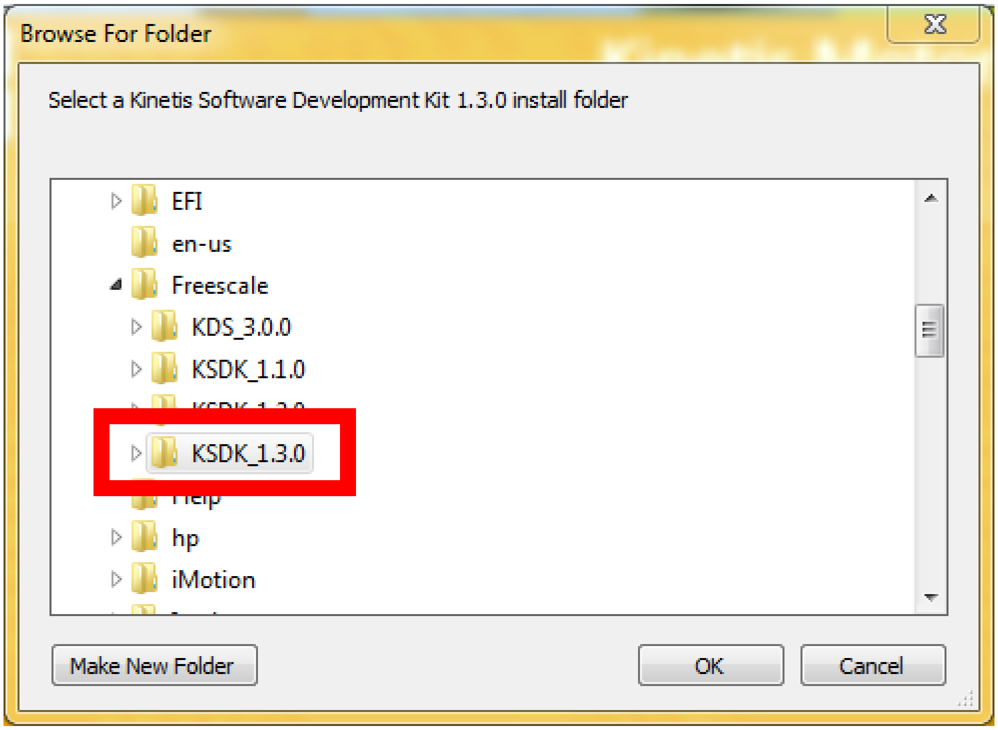


Figure 26. Select KSDK location

When you specify this location, KMS creates an environment variable on your system named KMS_KSDK_PATH. This can be viewed by going to your PC's Control Panel->System->Advanced system settings->Environment Variables. KMS uses this variable to define where it looks for KSDK files, so it is recommended that this variable and the files it refers to are not manually edited.

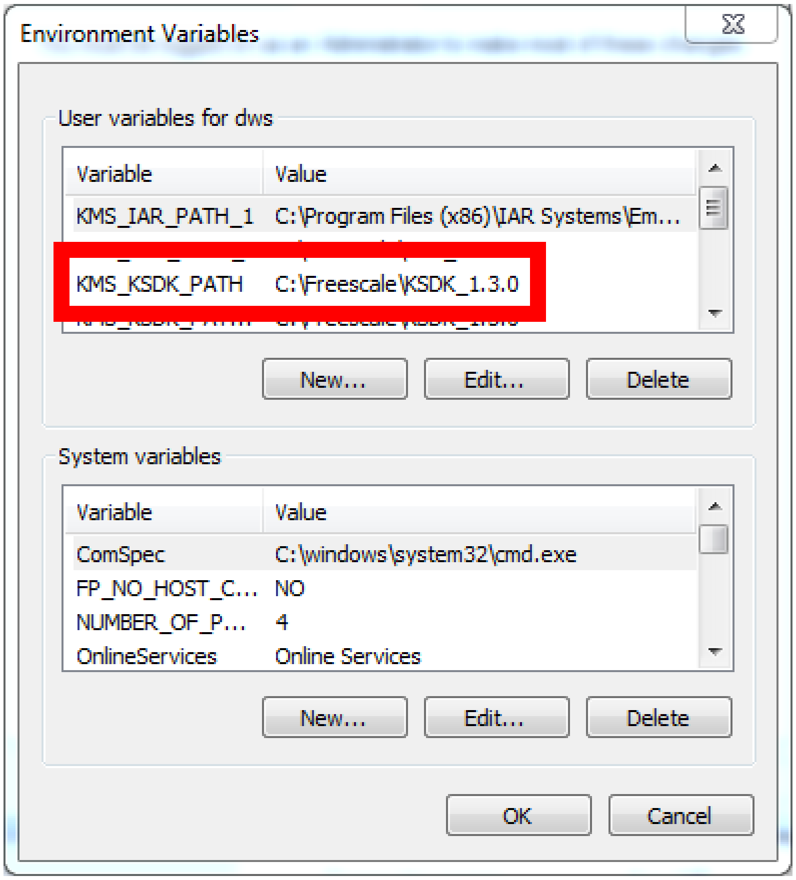


Figure 27. KSDK path variable created by KMS

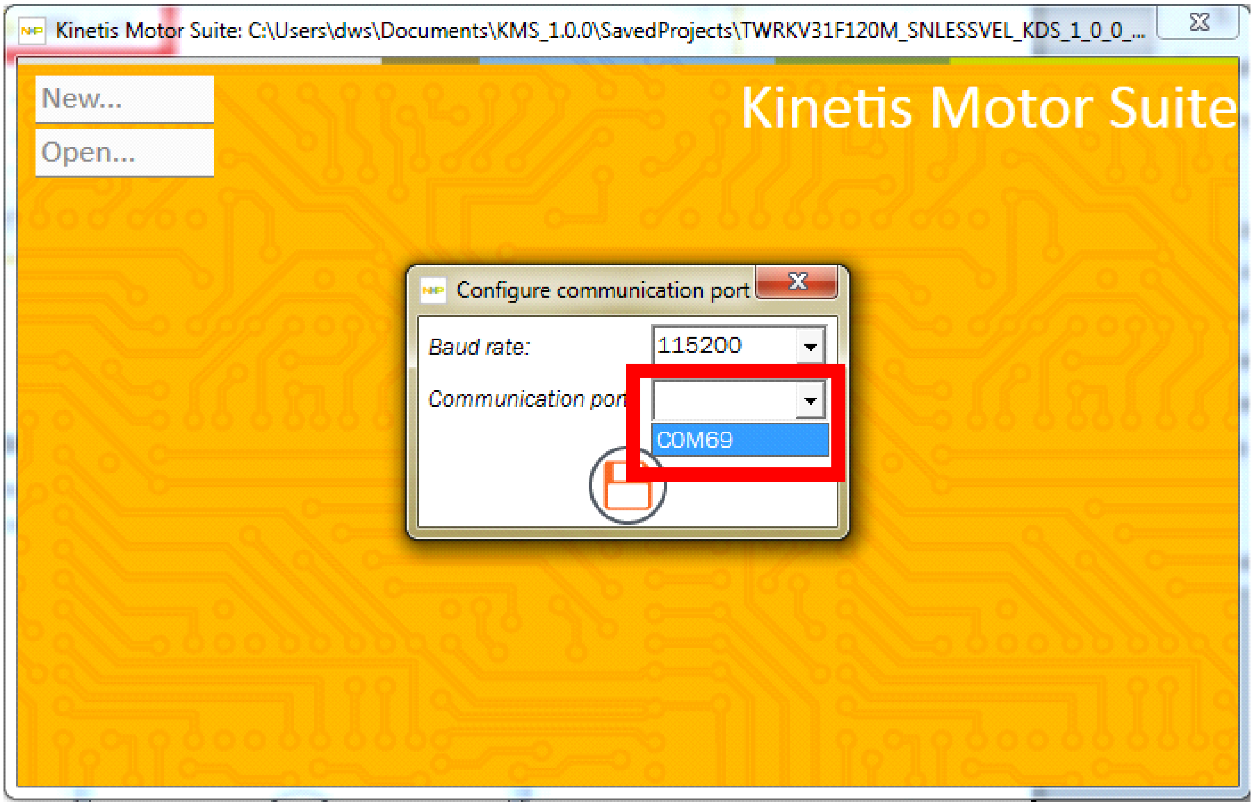
4.2.2 Communication

KMS communicates via USB serial port to the MCU. KMS requires definition of the port to be utilized.

After the KSDK location has been defined, KMS attempts to communicate with the MCU. First, KMS looks for a previously used COM port saved in KMS project settings. This search can succeed only if you have previously used KMS and have chosen to open an existing project.

- If KMS succeeds in finding a value for a previous COM port, it attempts to connect. If successful again, KMS proceeds to the next step: determining actual vs. expected MCU image.

- If no port exists or the previous value does not result in successful communication, KMS displays a dialog asking you to specify communication settings.



KMS prepopulates a default baud rate and dropdown options for COM port. The options in the COM port dropdown menu are defined by the ports identified by your PC, which can be viewed in the system Device Manager.

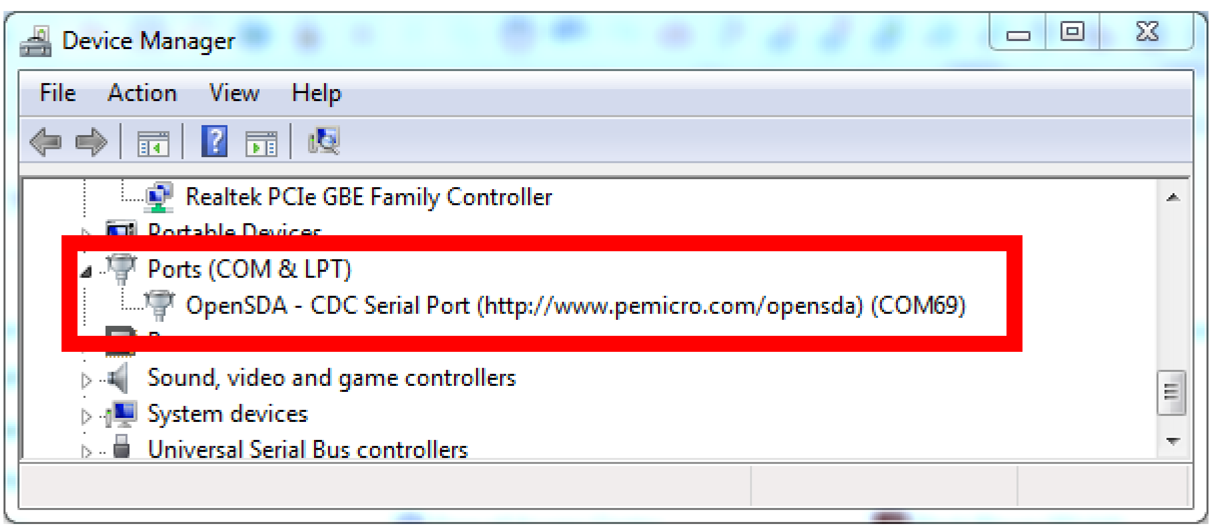


Figure 28. Available communication ports

In the KMS dropdown, select the communication port that corresponds to the PC-identified P&E Micro OpenSDA - CDC Serial Port. This is the default debug and serial adapter connection for KMS-enabled development platform hardware.

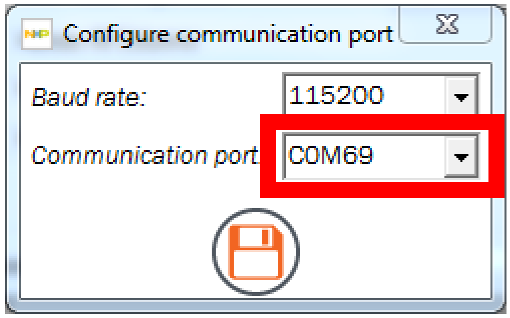


Figure 29. Specify communication port

Click to save this setting and proceed.

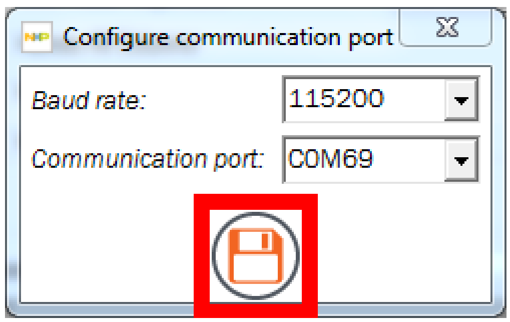


Figure 30. Click to save

- Refer to the KMS Release Notes for information on requirements for communication and debug software
- Refer to the KMS Lab Guide for instructions on locating, installing, and confirming the required communication and debug software

NOTE

If your development platform is not connected to your PC, you may exit this dialog box without specifying a communication port and proceed to the KMS GUI, operating offline. Communication configuration can be defined at any time from the KMS GUI Project menu or by right-clicking on the communication button at bottom right. These items are described later in this document.

4.2.3 Image validation

After selection of the communication port, KMS attempts to communicate with the device. There are three main scenarios:

- the KMS reference project image already on the MCU matches the image specified in the selected KMS project

- the KMS reference project image already on the MCU does not match the image specified in the selected KMS project
- there is no KMS reference project image on the MCU, only factory programmed KMS execute-only code

The image that KMS is expecting is the .elf (KDS) or .out (IAR) file defined in your KMS project directory.

- For KDS, this file is located in <KMS project directory>\kds\Release:

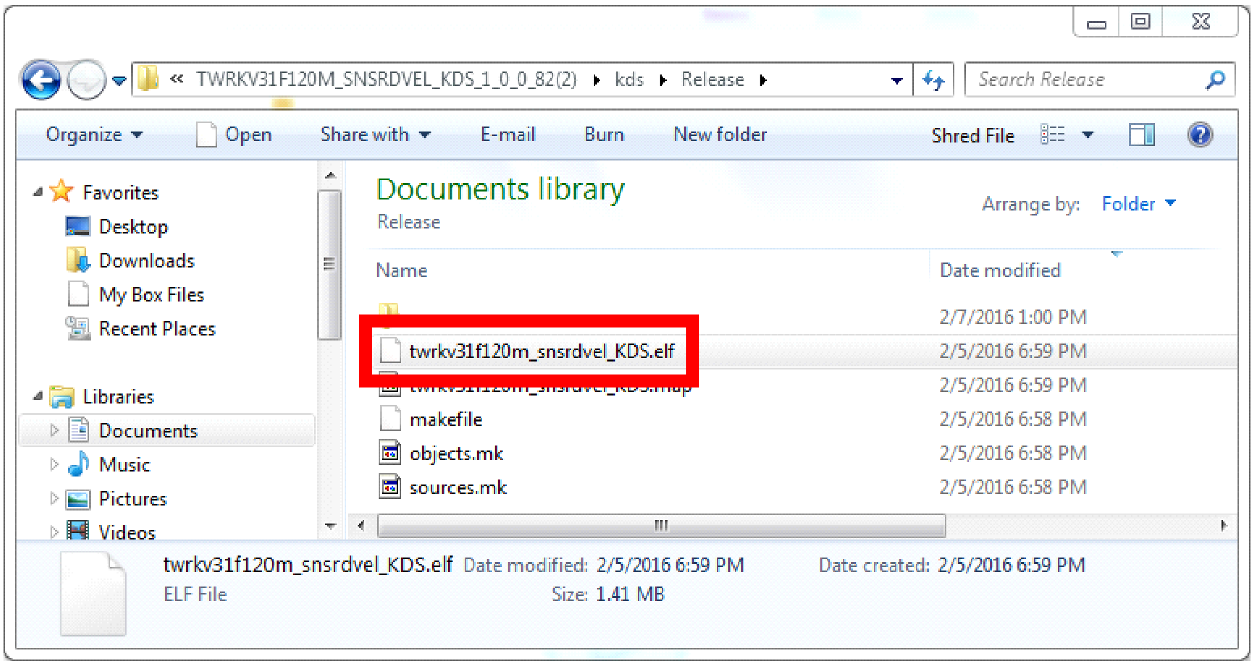


Figure 31. .elf file location

- For IAR, this file is located in <KMS project directory>\iar\Release\exe:

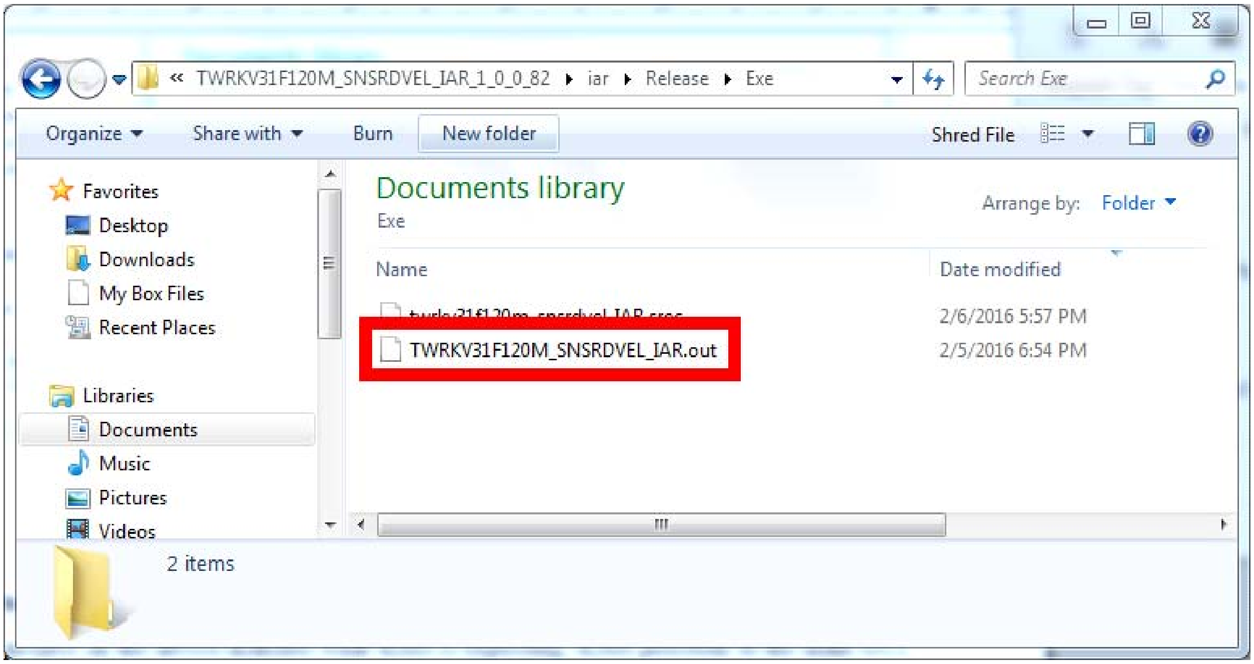


Figure 32. .out file location

4.2.3.1 Image match

If the reference project on the device matches what KMS is expecting, KMS proceeds to the main GUI window and indicates successful communication with the MCU via green arrows at the bottom of the window (see Section 5.1.1.6, "Status icons").

This is typical when you have previously worked with KMS and this MCU, and are opening an existing project you created and used only in KMS (i.e., you have not edited and downloaded code directly from your IDE).

4.2.3.2 Image mismatch

If the reference project on the device does not match what KMS is expecting, KMS prompts to ask if automatically overwriting the software on the MCU is desired.

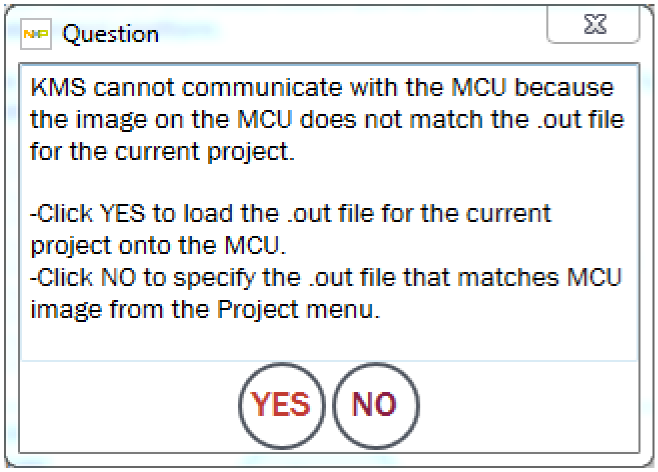


Figure 33. Image mismatch notification

This scenario typically occurs after code is downloaded to the MCU in one KMS session but then a new KMS session is started (instead of re-opening the previous KMS session). It also occurs when switching to a different control type or IDE.

Clicking YES starts the process of KMS downloading the .elf or .out file expected by the GUI project onto the MCU, whereas clicking NO pushes you to the main KMS window without connecting to the MCU.

Clicking YES therefore eliminates the existing code resident on the MCU (except KMS preprogrammed execute-only code). Do not select this option if you are working directly with the code on the MCU and wish to preserve it.

You may instead click NO, then change the .out file expected by the GUI by using the Project menu->Select Paths option to specify a new .out file path that does match the image on the MCU (see Section 5.1.1.8.2, "Project"). Then use the Project menu item Configure Communication Port... to attempt to connect.

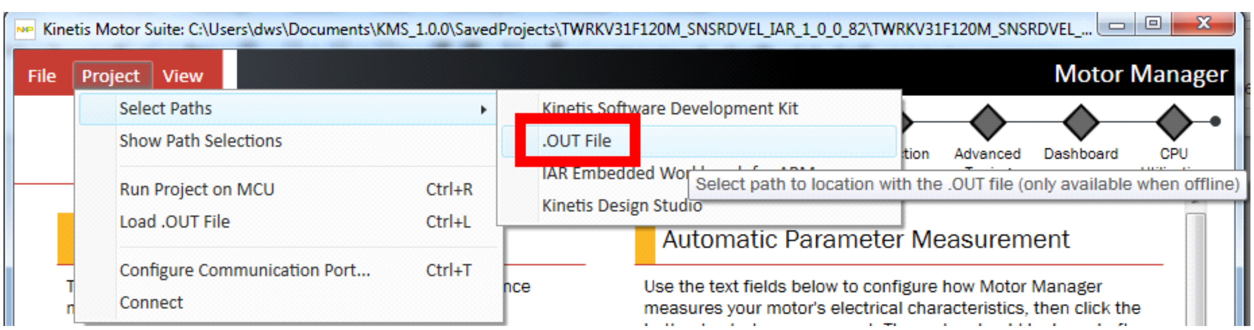


Figure 34. Select .out file path

If it is acceptable to overwrite the existing code on the MCU, KMS prompts for the location of your preferred IDE to leverage its download to target capabilities. You may receive different messages for this prompt depending on whether you have multiple versions of your IDE and if they are located in default installation directories.

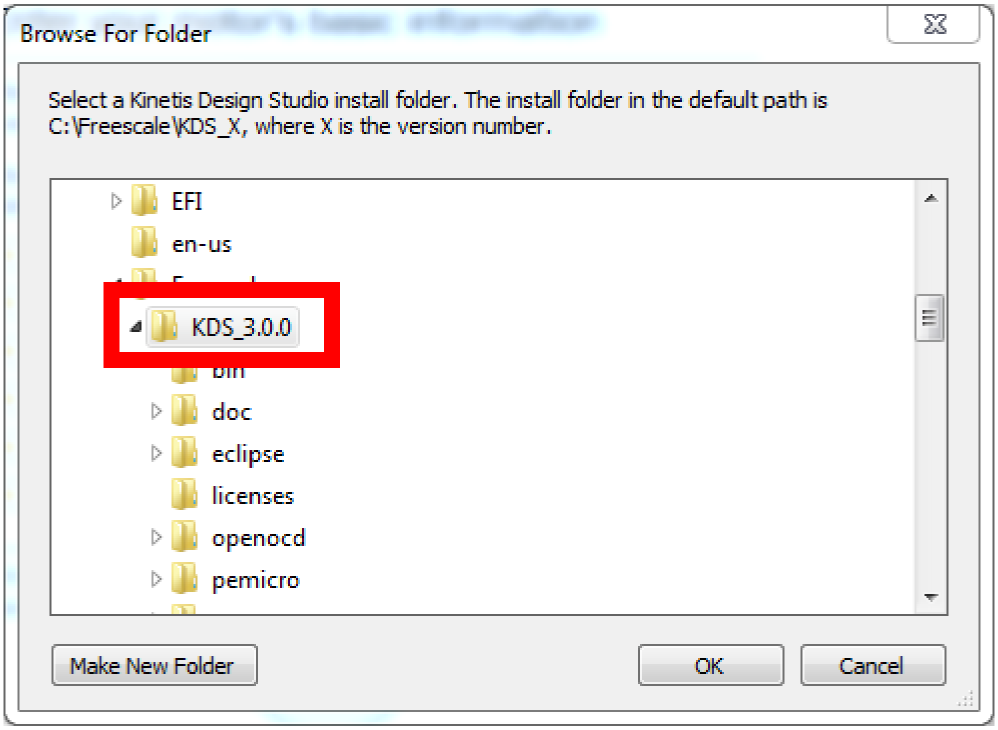


Figure 35. Specify KDS location

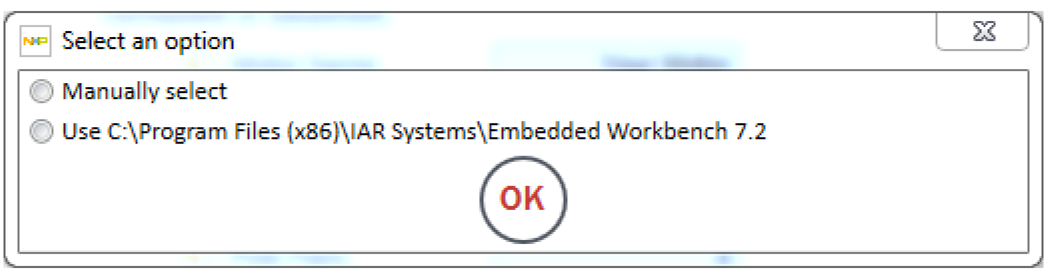


Figure 36. Specify IAR location

As with KSDK, specifying the location of your IDE creates an environment variable on your PC: KMS_KDS_PATH or KMS_IAR_PATH. These variables are reused whenever KMS utilizes compile or download functions.

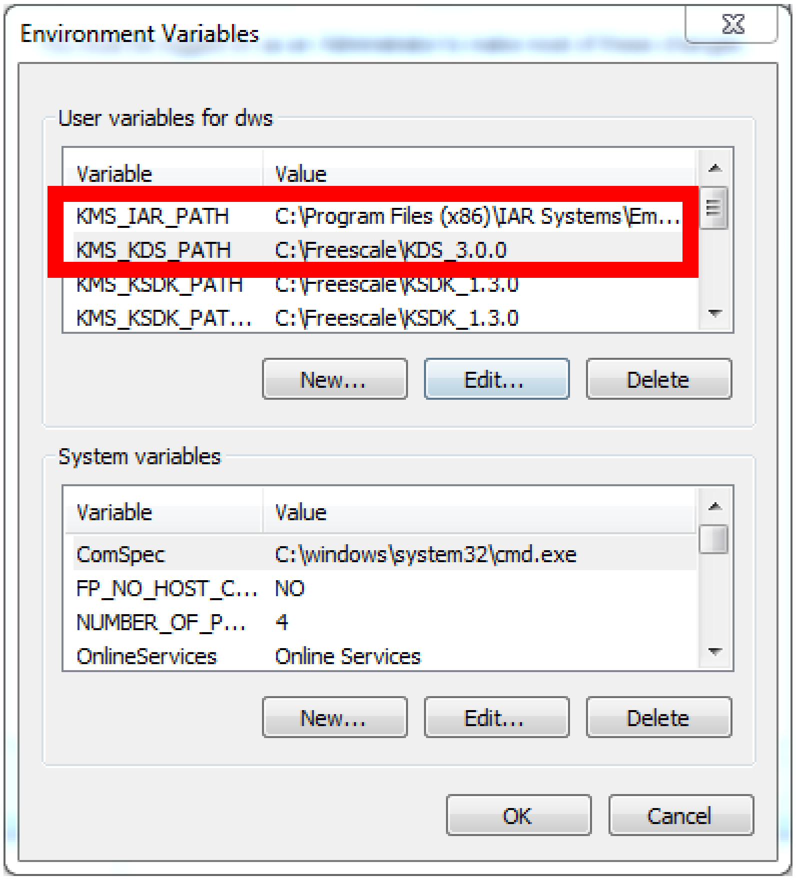


Figure 37. IDE path environment variables

After the location of the IDE has been configured, KMS invokes the specified tool to download the image file defined in the KMS project.

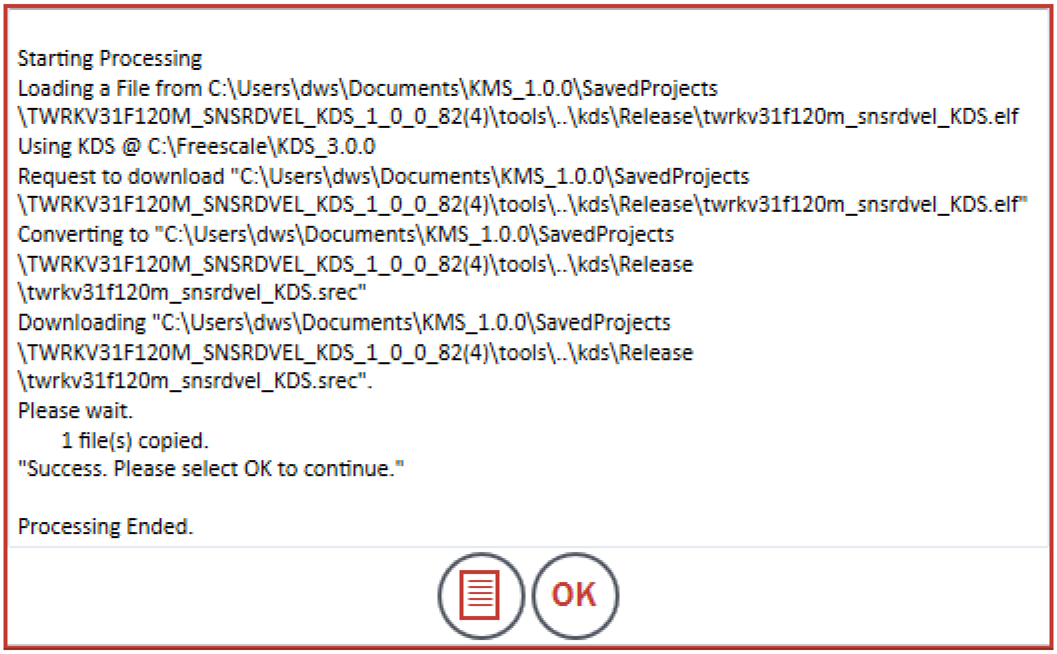


Figure 38. Download via KDS

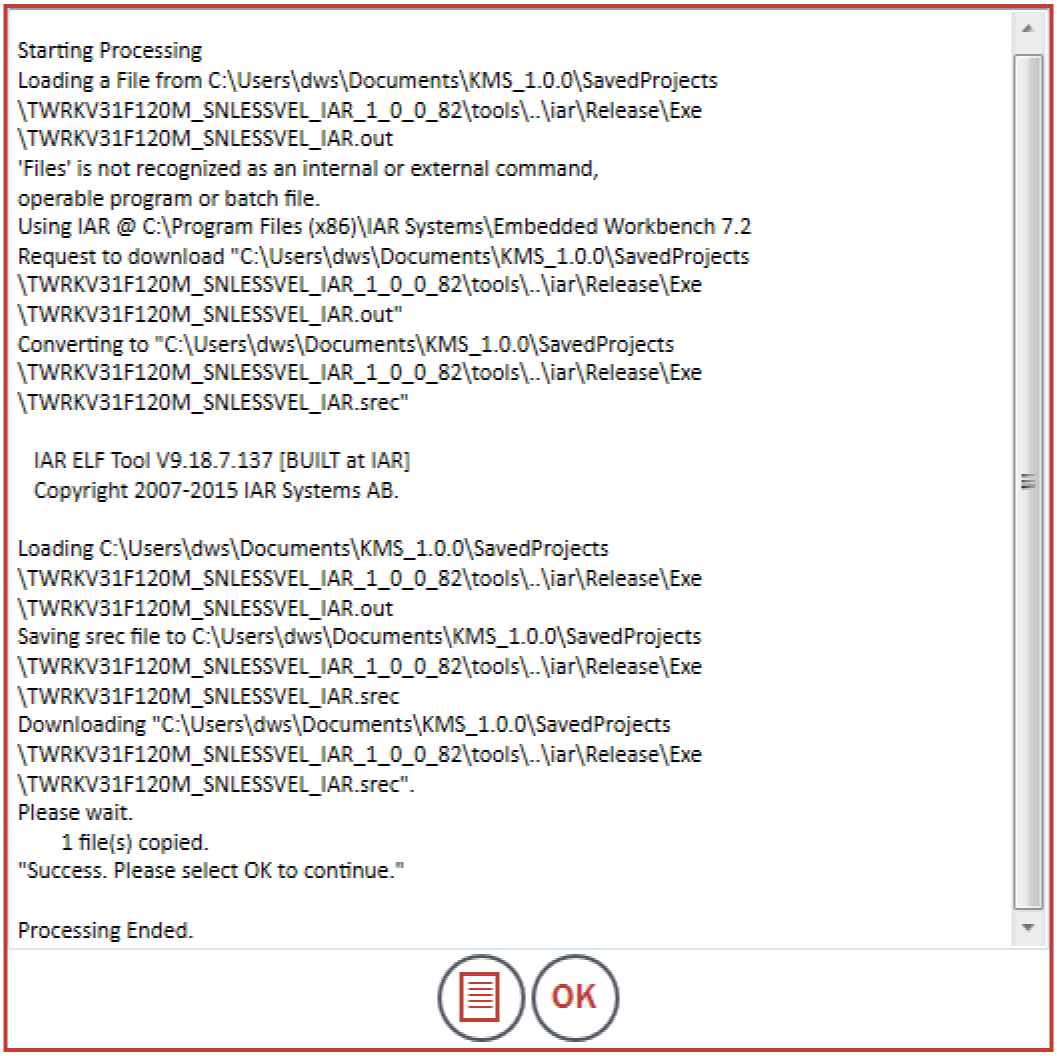


Figure 39. Download via IAR

Click OK to proceed; KMS should connect automatically at this point.

4.2.3.3 No image

If the KMS reference project image (.elf or .out file) does not exist on the MCU, KMS prompts not with the mismatch notification but with an indication that it cannot verify the communication port you specified.

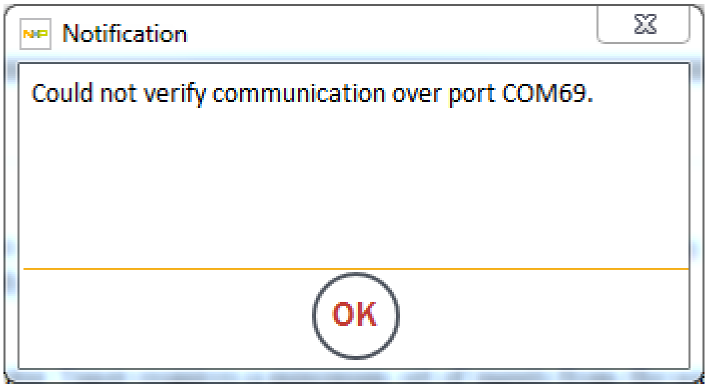


Figure 40. Notification symptomatic of absence of reference project image

This is because communication with the MCU relies on an agent in the KMS reference project image: if the image does not exist on the MCU, neither does the communication agent.

NOTE

This notification is not specific to absence of KMS image on target; it is a general communication issue. If this message appears during normal operation, first assess the physical connection.

To address absence of KMS reference project image, click OK to accept the notification, then select Project->Load .OUT file, and download the image to the MCU. You may be required to specify the location of your IDE as in Section 4.2.3.2, "Image mismatch".

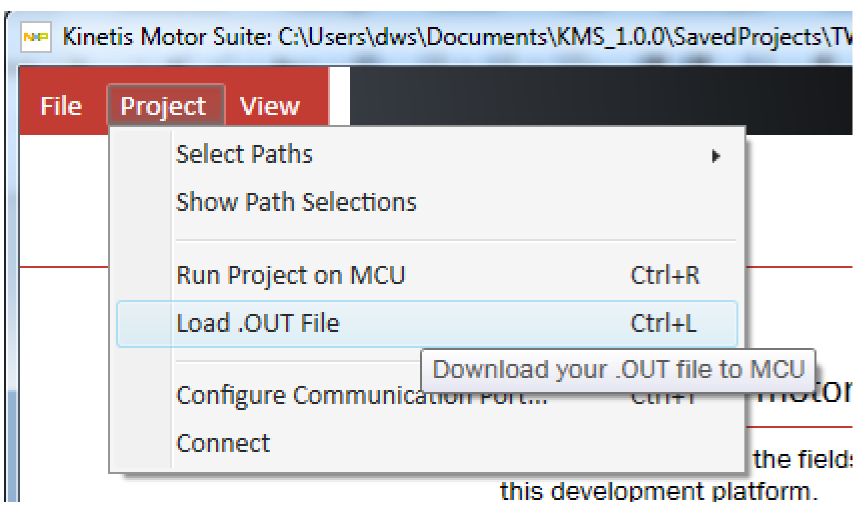


Figure 41. Load .out file to MCU

Retry communication via the Configure Communication Port option in the Project menu or by right-clicking on the communication icon at bottom right.

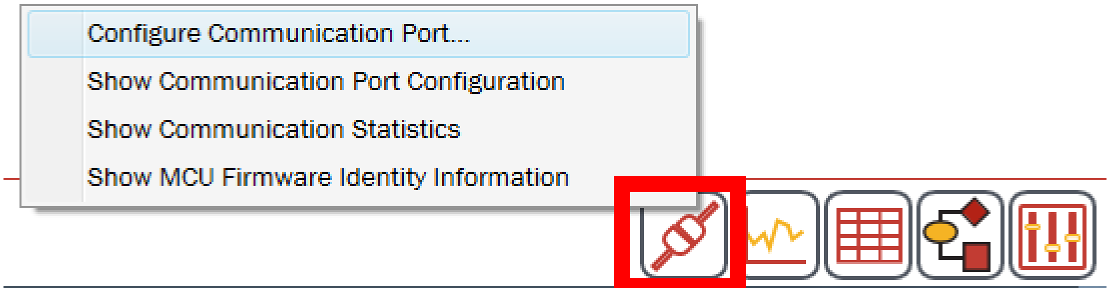


Figure 42. Right click to configure communication

5 KMS GUI basics

Once the KMS GUI and the KMS-enabled MCU are successfully communicating, the design process can begin. The KMS GUI includes tools that can be used at different stages of a design. The purpose and implementation of each key tool is briefly described as follows:

- Motor Tuner: spin your motor
  - — The default startup state for KMS, Motor Tuner requires a minimum set of inputs from the user to identify motor & inertia, then use this information to spin the motor.
- Motor Manager: optimize motor control settings
  - — A superset of Motor Tuner offering greater configuration options, Motor Manager is appropriate for fine-tuning an application and commissioning non-standard motors and hardware.
- Motion Sequence Builder: quickly create application trajectory
  - — A graphical user interface for defining complex motion sequences using fundamental components such as states of motion, transitions between states, variables, and conditions.
- Real Time Debugging: view, change, and plot system variables
  - — A combination of Software Oscilloscope and Watch Window, KMS Real Time debugging enables easy display, tweaking, and recording of system information to aid in system level debugging.

Each of these tools is described in detail in the following sections.

5.1 General operation

However, before describing how to use KMS to design a motor application, it is necessary to understand how to use the KMS GUI itself - that is, the GUI’s basic layout and general operation.

A high level explanation is provided in this section. Refer to Section 13, "GUI elements glossary" for descriptions of specific GUI elements.

Note that the KMS GUI renders differently based on your desired control type (sensorless velocity, sensored velocity, or sensored position). Where functionality is distinct, the different views are presented and explained. Where functionality is equivalent, the sensorless velocity screen is used for explanation.

5.1.1 Structure

Motor Tuner and Motor Manager share basic structural elements and define the manner in which the user interacts with KMS. Motion Sequence Builder and the Real-Time Debugging tools operate as subwindows of Motor Tuner and Motor Manager.

Figure 43 highlights major GUI elements that are described in this section. These are specific to Motor Tuner and Motor Manager.

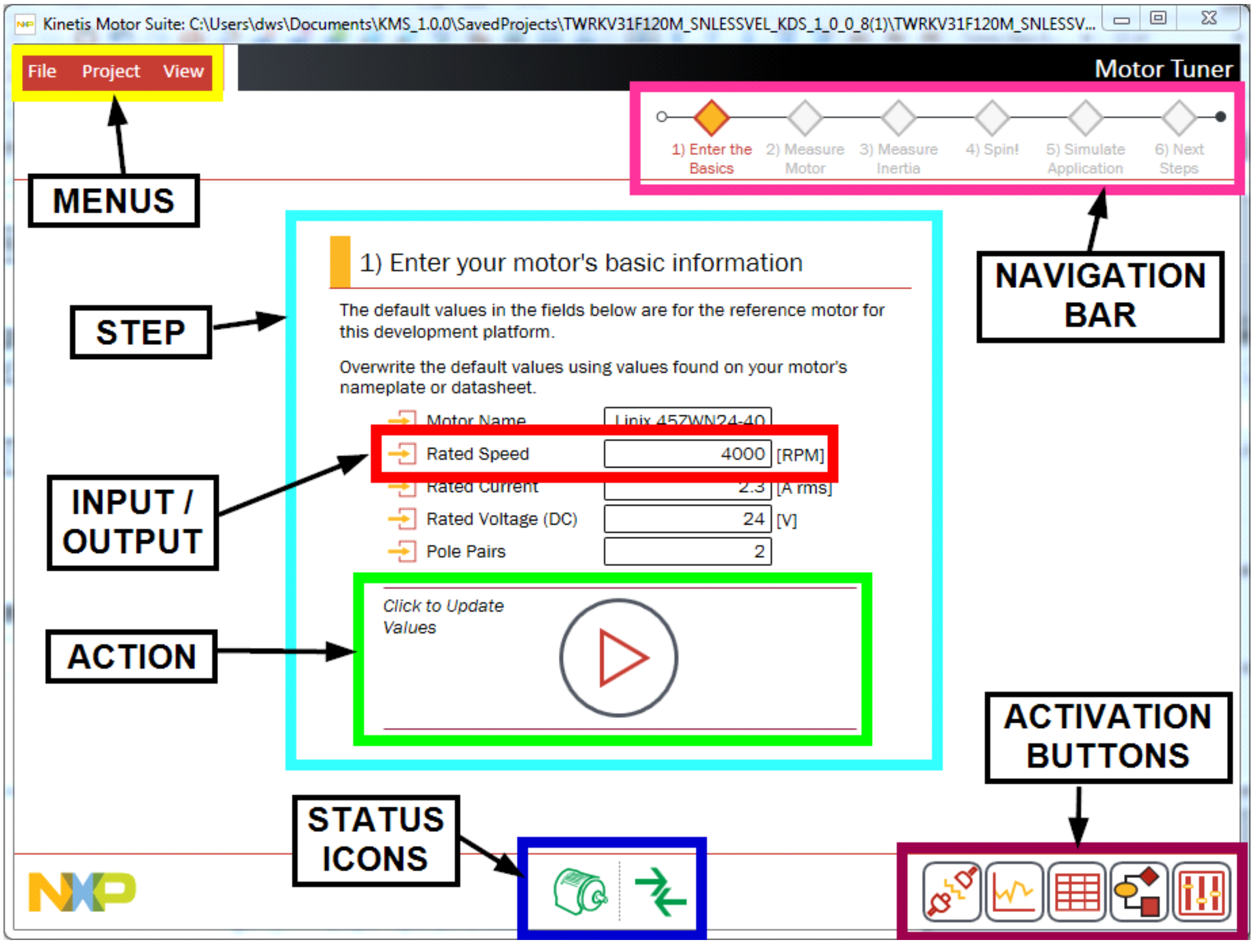


Figure 43. Major GUI elements

5.1.1.1 Input/output

The most common element in the KMS GUI is the input/output parameter.

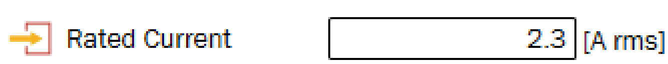


Figure 44. Example input parameter

Figure 45. Example output parameter

Input/output parameters are recognizable for having:

- graphical indicator of input vs. output
- name

- text, dropdown, or check box (editable if an input)
- units designation
- tool tip (on hover)

Input/output parameters are primarily configuration values that may be conveyed to or read from the MCU. If the parameter is an input, it informs the process of setting up the motor drive on the MCU. If the parameter is an output, it displays the scaled value of a variable in the MCU.

Input parameters are evaluated by KMS as they are edited by the user. This result of this evaluation is indicated by different colors and further detail may be given by tool tip:

- Red: invalid entry according to KMS max/min values or forbidden characters
- Green: valid entry according to KMS max/min values and forbidden character list
- Orange: valid entry changed from previous value but not yet taken into account in system configuration (typically occurs when a value must be sent to the MCU to take effect)

Input/output parameters that no longer exist in the KMS reference project are indicated as disabled. This may occur if you remove certain KMS firmware code modules in the IDE but still want to use the KMS GUI. Note however that the ability to continue using the KMS GUI requires the existence of certain core functionality (not all removal will result in simply disabling certain parameters).

Figure 46. Disabled input/output parameter

5.1.1.2 Action

Actions are processes that must be explicitly initiated by the user. Actions are denoted by the existence of the Run/Stop button and, as shown in Figure 47, have a structure consisting of:

- description
- Run/Stop button
- status indicator (optional)

Upon initiation of the command, the Run/Stop button toggles from Run to Stop. The nature of the command is described by the accompanying name and further detail may be gained from the catalogue of actions in Section 13, "GUI elements glossary". The size of the Action element is different in Motor Tuner and Motor Manager.

Figure 47. Example action (in Motor Manager)

5.1.1.3 Step

Related input/output parameters and actions, combined with explanatory text, make up Steps. Steps are indicated by a common header style (Figure 48) and serve primarily to group tightly related configuration options.

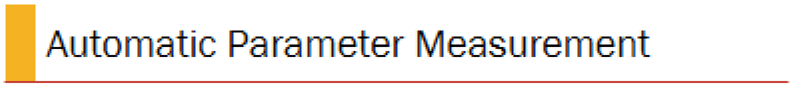


Figure 48. Step style

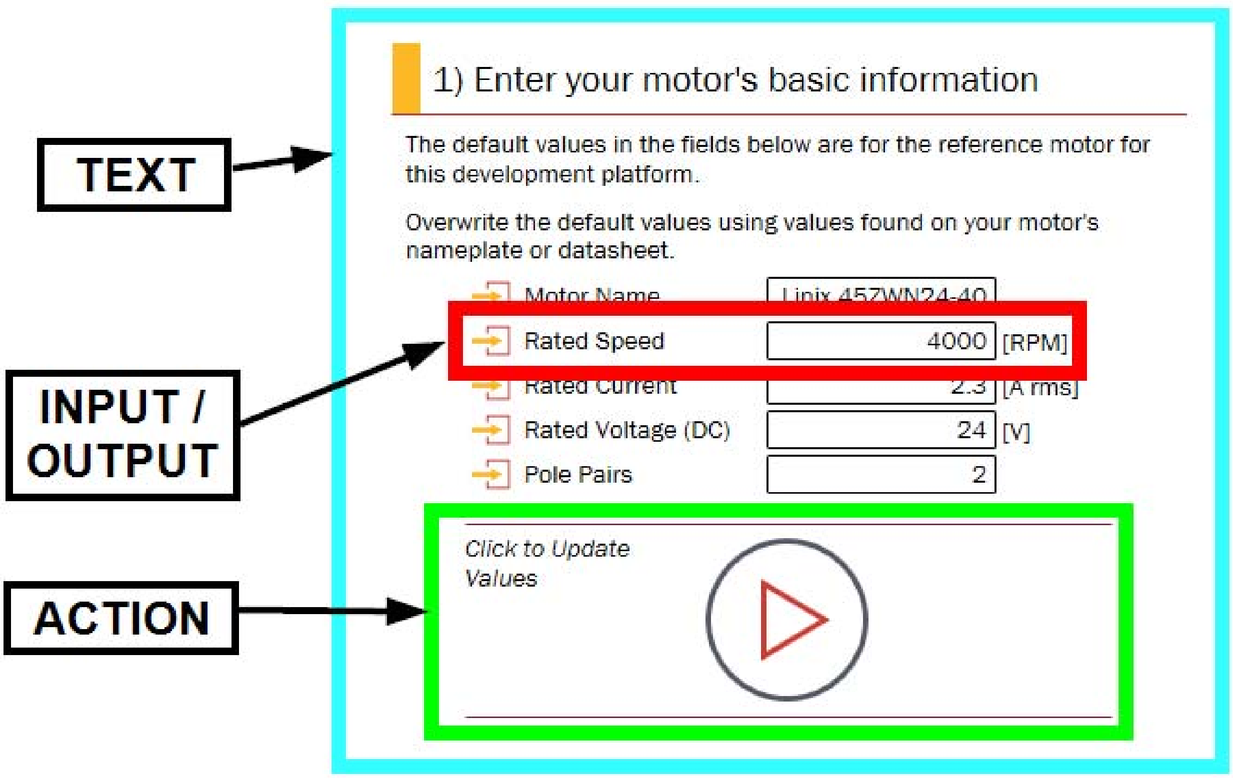


Figure 49. Structure of an example step

5.1.1.4 Pages

Related steps are then combined on a page. In Motor Tuner, where the emphasis is on simplicity, a single Step exists on each page. In Motor Manager, where the emphasis is on flexibility and optimization, KMS may provide numerous steps per page.

Figure 50 shows an example page - the Speed Control page in Motor Manager. Steps are outlined in red.

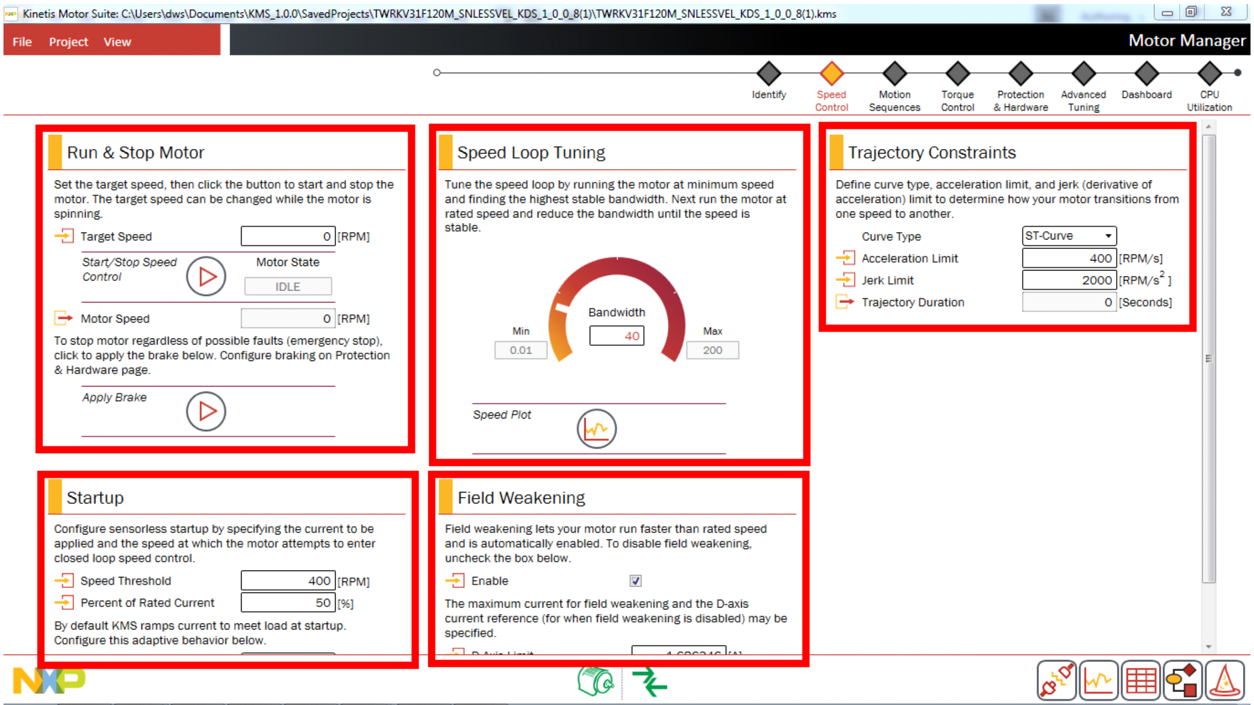


Figure 50. Example page

5.1.1.5 Navigation bar

The navigation bar, located at top right and always visible in the KMS main window, guides the user through the application design cycle by providing clickable links to pages.

In Motor Tuner, KMS automatically pushes the user to the next page upon successful completion of the activity on the previous page. Page links are disabled until Motor Tuner recognizes successful completion of the previous page's activity.

Conversely, in Motor Manager, all pages are accessible and navigation is the responsibility of the user.

The content of the navigation bar depends on the current mode of the system. The number of pages displayed changes based on whether the system is in Motor Tuner (Figure 51) or Motor Manager (Figure 52). The number and nature of pages may also change according to control type (Figure 53).

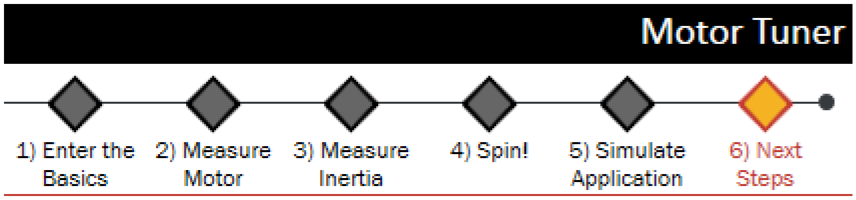


Figure 51. Navigation bar for Motor Tuner (sensorless velocity)

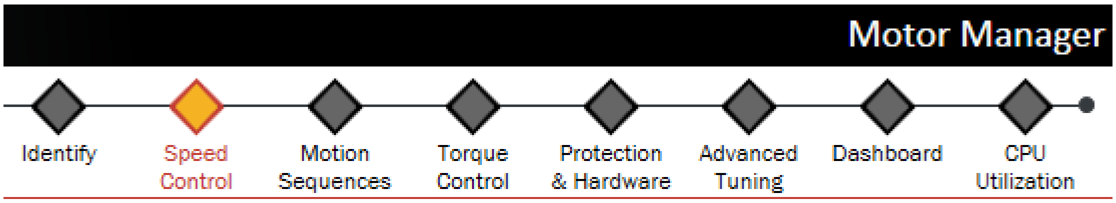


Figure 52. Navigation bar for Motor Manager (sensorless velocity)

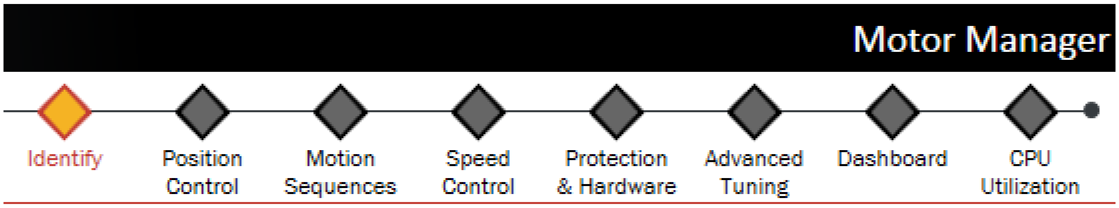


Figure 53. Navigation bar for Motor Manager (sensored position control)

The current page is indicated by the formatting shown in Figure 54.

Figure 54. Active page style

If a page is disabled and not clickable, this is indicated by the formatting shown in Figure 55. This most often occurs when the user has not completed the steps in Motor Tuner.

Figure 55. Disabled page style

5.1.1.6 Status icons

Appearing at the bottom center of the KMS main window at all times are two icons reflecting the status of the system. The motor icon (Figure 56) reflects the state of the motor with respect to known faults from the firmware. The arrows icon reflects the state of communication between PC and MCU (Figure 57).

Figure 56. Motor status indicator - status unknown

Figure 57. Communication status indicator (good)

Colors describe the states. For the Motor Status Indicator, the available states are:

- Green = good operation
- Yellow = unknown (e.g., due to offline operation)
- Red = firmware fault declared

For the Communication Status Indicator, the available states are:

- Green = online/connected
- Red = offline/disconnected

5.1.1.7 Activation buttons

Also in the bottom bar are five activation buttons for communicating with the MCU and accessing the main GUI tools. The five buttons, from left to right, are:

- Connect/Disconnect
  — toggle button that commences or terminates communication between PC and MCU.

Figure 58. Click to connect

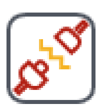


Figure 59. Click to disconnect

— Right clicking on this button displays a menu of further communication options, including configuration.

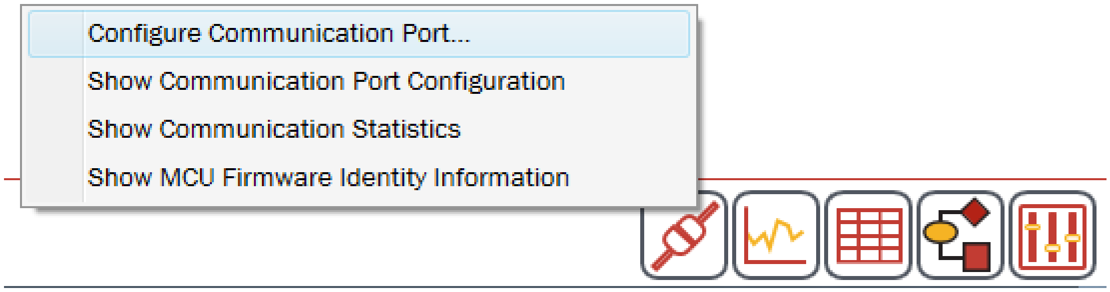


Figure 60. Additional options available by right-clicking on connect/disconnect button

- View Plot Window
  — activates and brings to the front the Software Oscilloscope, described in Section 8.1, "Software Oscilloscope".

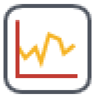


Figure 61. Click to show Software Oscilloscope

- View Watch Window
  — activates and brings to the front the Watch Window, described in Section 8.2, "Watch Window".

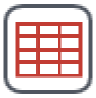


Figure 62. Click to show Watch Window

- Open Motion Sequence Builder
  — activates and brings to the front the Motion Sequence Builder, described in Section 9, "Motion Sequence Builder".

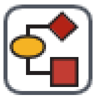


Figure 63. Click to show Motion Sequence Builder

- Switch Mode

— toggle button that switches between Motor Tuner and Motor Manager.

Figure 64. Click to go to Motor Tuner

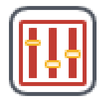


Figure 65. Click to go to Motor Manager

5.1.1.8 Menus

KMS has three dropdown menus at the top left of the screen. These menus serve to categorize and provide easy access to frequently used application options.

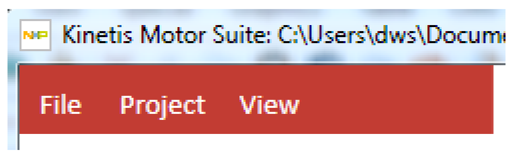


Figure 66. Dropdown menus

Table 6. KMS dropdown menu items

Menu	Menu items
File	Typical application operation: new, open, save, etc.
Project	Options related to the MCU and KMS firmware
View	Access to the main KMS GUI tools

5.1.1.8.1 File

The File menu's options are described in Table 7.

Table 7. File menu

Option	Function
New...	Enables you to start a new KMS project. Allows you to save current project then displays the Select system configuration window so you may change your configuration if desired.
Open...	Allows you to open an existing KMS project. You may save current project, then KMS displays the option to open a recent project or manually select a project file. The project file is denoted by the file extension .kms and is located in the KMS project folder created during the process described by Section 4, "System configuration".
Save	Saves the current KMS GUI project file (.kms file).

Table 7. File menu

Option	Function
Save As...	Allows you to save the current KMS GUI project file (.kms file) with a new name or path.
About	Provides version information for the utilized KMS GUI and indicates third-party software and licenses.
Exit	Closes the KMS GUI after permitting you to save the current project file (.kms file).

5.1.1.8.2 Project

The Project menu provides options meant to configure the interaction between the KMS GUI and embedded firmware reference project. The menu's options are described in Table 8 and Table 9.

Table 8. Project menu

Option	Function
Select Paths	KMS requires knowledge of several items in order to function properly. This option opens a submenu that allows for specification of these items. This submenu is described in Table 9.
Show Path Selections	KMS requires knowledge of several items in order to function properly. This option displays the current selections (if any). See Figure 67.
Run Project on MCU	Compiles and downloads to the MCU the KMS reference project in its current state.
Load .OUT File	Downloads the latest built version of the KMS reference project located at the selected path (default path is under the chosen IDE folder in the KMS reference project directory). Does not compile.
Configure Communication Port	Allows specification of the COM port and baud rate used to connect the KMS GUI to the MCU. Cannot be configured while already communicating.
Connect (Disconnect)	Attempts to connect or disconnect communication with the MCU depending on current state of communication.

Table 9. Select paths submenu

Option	Function	Example path
Kinetis Software Development Kit (KSDK)	KMS relies on the KSDK to configure Kinetis MCUs.	C:\Freescale\KSDK_1.3.0
.OUT File	KMS parses the compiled output of the KMS reference project that is loaded to the MCU in order to communicate with the MCU.	C:\Users\<username>\Documents\KMS_1.0.0\SavedProjects\TWRKV31F120M_SNLESSVEL_IAR_1_0_0_8\iar\Release\Exe\TWRKV31F120M_SNLESSVEL_IAR.out

Table 9. Select paths submenu

Option	Function	Example path
IAR Embedded Workbench for ARM	If chosen during system configuration, KMS configures its reference project to use the IAR IDE tools.	C:\Program Files (x86)\IAR Systems\Embedded Workbench 7.2
Kinetis Design Studio	If chosen during system configuration, KMS configures its reference project to use the KDS IDE tools.	C:\Freescale\KDS_3.0.0

NOTE

The example paths in Table 9 are provided to show the directory level that KMS expects. Your location paths may be different.

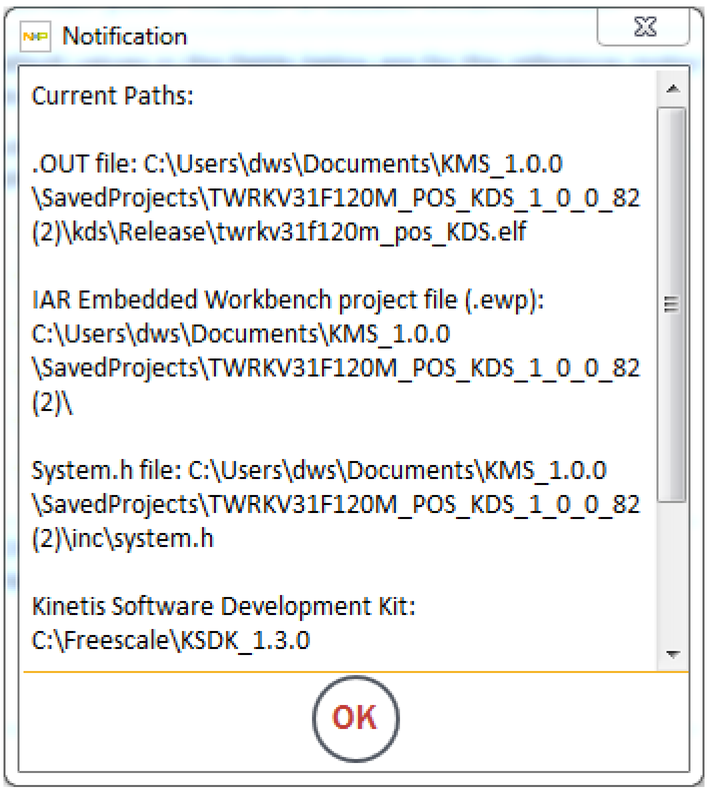


Figure 67. Example output of Show Path Selections menu item

5.1.1.8.3 View

The View menu's options are described in Table 10.

Table 10. View menu

Option	Function
Launch	Opens submenu that allows you to launch key features of the KMS GUI. Described in Table 11.
Go to Motor Manager/ Motor Tuner	Enables toggling between Motor Tuner (wizard) and Motor Manager (optimization) views.
Documents...	Opens window that allows selection of KMS product documentation, including: • API Reference Manual • Lab Guide • User's Guide • Release Notes • Adapting KMS for Custom Hardware App Note

Table 11. View->Launch submenu

Option	Function
Software Oscilloscope	Launches the Software Oscilloscope for plotting and viewing MCU variable changes over time.
Watch Window	Launches the Watch Window for viewing and writing to MCU variables.
Motion Sequence Builder	Launches the Motion Sequence Builder window for creating motion state machines from graphical interface.
Create support time capsule	• Launches a form for entering system and issue details. This information is zipped with the .kms project file, system.h motor characteristics header file, and current session trace log for easy delivery to a support contact. A notification showing where this time capsule has been saved appears after usage. • The same form appears on application errors requiring shutdown. • See Section 12, "Frequently asked questions" for additional information.

6 Motor Tuner

Motor Tuner guides the user through the fundamental steps for getting a motor up and running. These steps are shown in Table 12.

Table 12. Motor Tuner

Page	Applicable control types	Description
Enter the Basics	All	Enter your motor's basic information, which can be found on the motor nameplate or in the motor data sheet
Measure Motor	All	Energizes and rotates your motor to measure the electrical characteristics.
Measure Inertia	All	Accelerates and decelerates the system to identify the mechanical characteristics. This step may be performed with a bare motor shaft or with the motor connected to the unloaded application.
Tune Controller	Sensored Position only	Aligns the motor then requires you to adjust a single value until your motor is holding position well.
Spin!	All	Spin your motor to the rated speed (velocity) or for a single revolution (position). The KMS Software Oscilloscope plots desired vs. actual motion.
Simulate Application	All	Command your motor to follow a trajectory that simulates a simple washing machine (velocity) or security camera (position).The KMS Software Oscilloscope plots the trajectory.
Next Steps	All	At this point, you may create your own application trajectory using Motion Sequence Builder, or optimize your motor control settings with Motor Manager.

Each Motor Tuner page is explained in this section according to the following structure:

- Screen: what the page looks like
- Description: the functional purpose of the page
- Parameters: key on-screen items specific to the page that the user may want to utilize during the design cycle.

Parameters are typically in/out parameters as outlined in Section 5.1.1.1, "Input/output". On a per-page basis, they are aggregated into a table and described in terms of function and relationship to the underlying embedded motor control software, as shown in Table 13.

Table 13. Explanation of parameter table

Parameter	Description	MCU variable
Name of parameter as it appears on screen	Explanation of purpose	• N/A

Note that not every item that appears on a page is accounted for in the parameter table. Items that are not enumerated in the parameter table are typically self-explanatory or do not offer anything to the user in terms of configurability.

Note also that not every named parameter is explicitly tied to a firmware variable. This is because the KMS GUI is not simply an interface for reading/writing to the MCU; rather it embeds a level of intelligence on top of this. Parameters that are not tied directly to an MCU variable are typically used in broader system calculations to configure motor drive operation.

6.1 Enter the basics

6.1.1 Screen

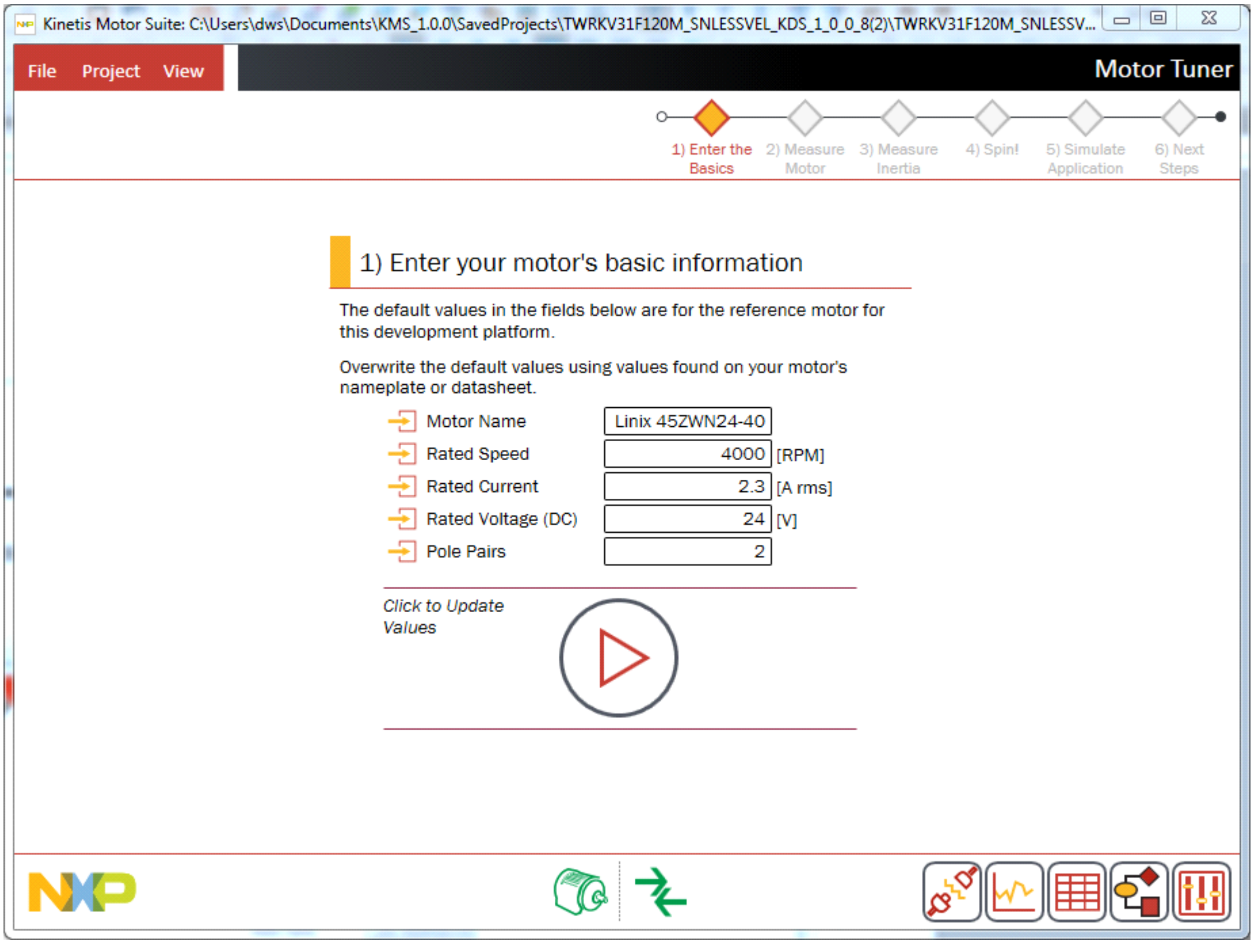


Figure 68. Enter the basics (sensorless velocity)

6.1.2 Description

On this page, the user is prompted to enter the motor's basic information, which can be found on the motor nameplate or in the motor data sheet. The parameters that are expected are described in Table 14.

6.1.3 Parameters

Table 14. Basic motor parameters

Parameter	Description	Firmware variable
Motor Name	Enter a name for the motor for convenience. This is not required.	• N/A
Rated Speed	The maximum speed at which the rated torque can be delivered. This value is used to scale motion-related values (e.g., speed command or acceleration limit) in the firmware	• N/A
Rated Current	The RMS (not peak) current for which the motor is rated. This value is used to establish the maximum output of the speed controller.	• N/A
Rated Voltage (DC)	Rated DC bus voltage of the motor. If specified on datasheet as VAC, multiply by square root 2 to arrive at VDC value.	• N/A
Pole Pairs	The number of pairs of magnetic poles in one mechanical revolution.	• N/A
Encoder Lines*	Number of lines (pulses) on encoder wheel	• N/A

*Sensored control (velocity or position) only

6.2 Measure motor

6.2.1 Screen

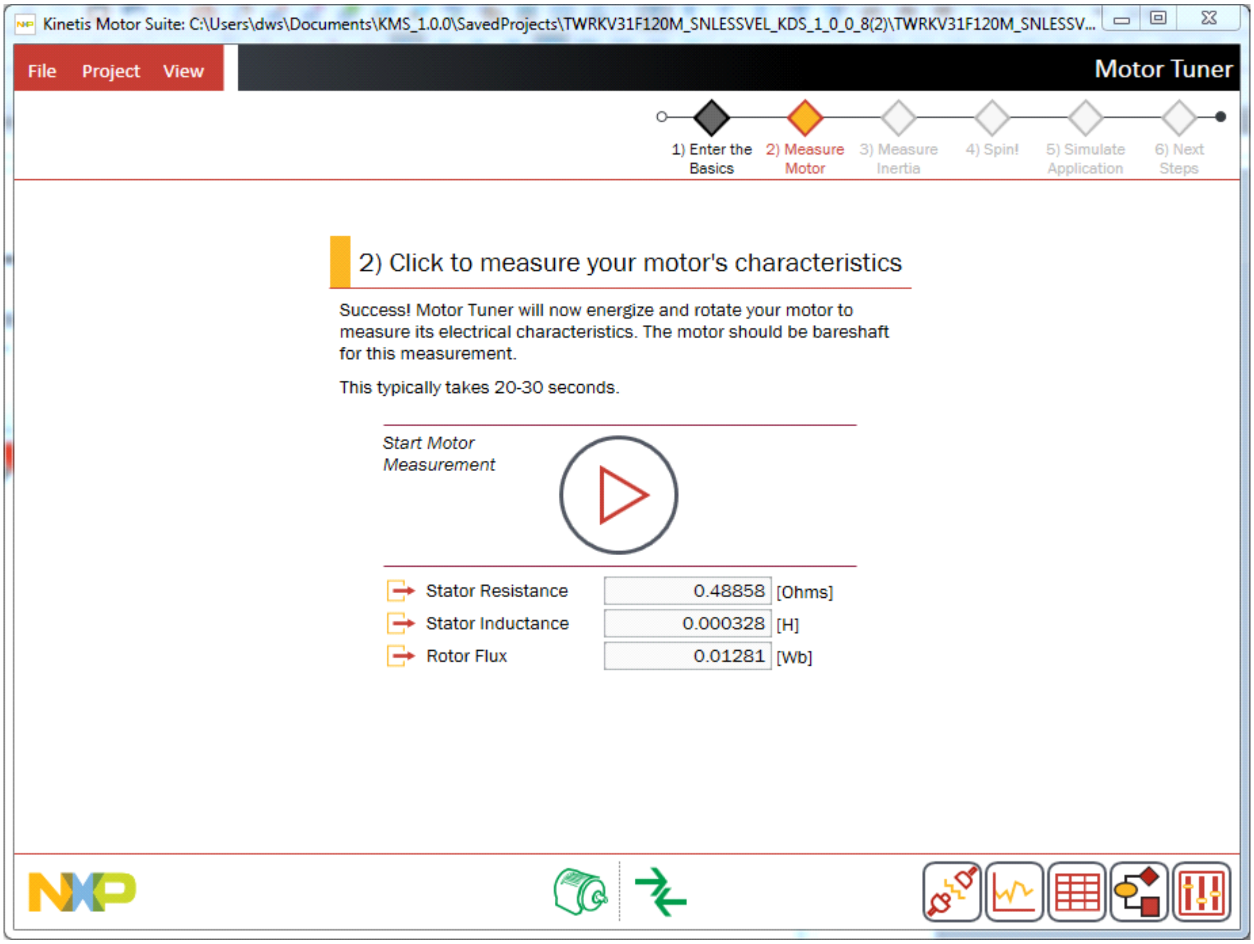


Figure 69. Measure motor

6.2.2 Description

In the second step, Motor Tuner energizes and rotates the motor to measure its electrical characteristics.

- Refer to the KMS API Reference Manual for a more detailed description of the measurement routines.

NOTE

When running with a sensor (sensored velocity or position), this step incidentally validates your encoder setup: if the encoder has been incorrectly configured or specified (e.g., mismatch of motor phases, wrong number for lines of encoder, not plugged in, etc.), Motor Tuner cannot complete the Rotor Flux measurement and returns a notification to retry.

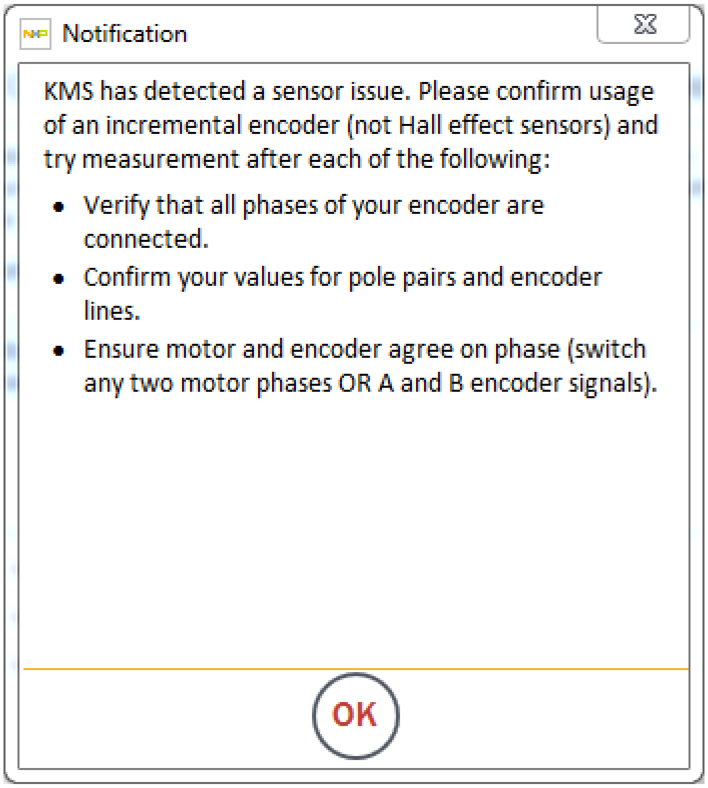


Figure 70. Notification for encoder issue

6.2.3 Parameters

Table 15. Automatically identified motor parameters

Parameter	Description	Firmware variable
Stator Resistance	The per phase winding resistance of your motor	• scm.output.statorRes
Stator Inductance	The per phase winding d-axis inductance of your motor at rated current. The q-axis inductance is assumed equivalent (non-salient motor).	• scm.output.statorInd
Rotor Flux	The rotor flux (or permanent magnet flux linkage) of your motor	• scm.output.pmFlux

6.3 Measure inertia

6.3.1 Screen

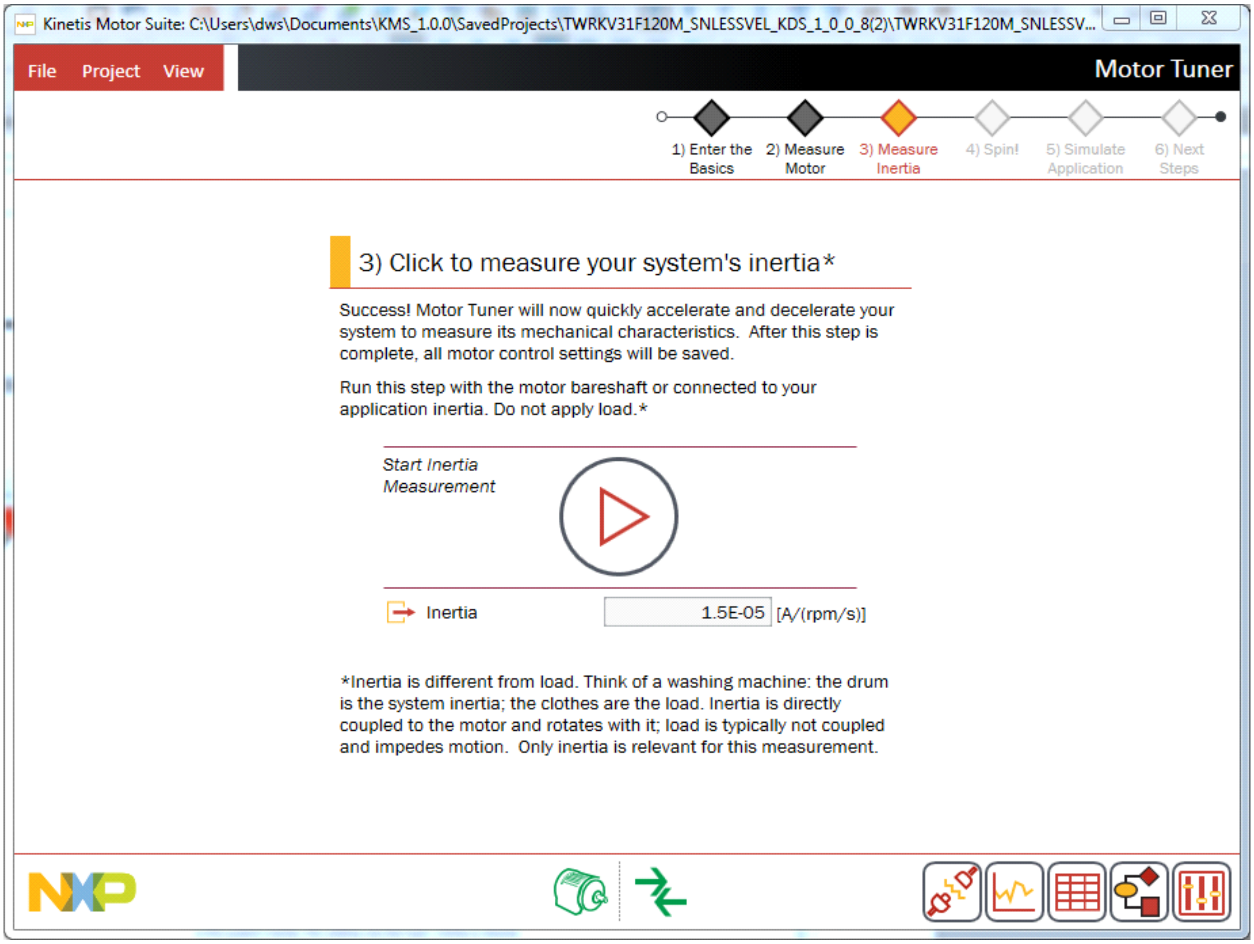


Figure 71. Measure inertia

6.3.2 Description

Inertia is important for controlling the motion of the application but is often neglected by traditional motor control approaches. KMS uses inertia as a direct input to create an appropriate mechanical model of the system for advanced control.

Motor Tuner accelerates and decelerates the system to identify the mechanical characteristics. This step may be performed with a bare motor shaft, or with the motor connected to the unloaded application.

If you are simply interested in running your motor (without anything connected to the motor shaft), Motor Tuner measures the inertia represented by the motor shaft itself. However, if you are running your application or testing with inertia, connect the motor to the relevant inertia before performing this measurement. Anything that spins directly with the motor during operation should be accounted for in this measurement.

In some instances, Motor Tuner may not be able to automatically measure the system's inertia with the default settings. In this case, Motor Tuner automatically adjusts the settings and prompts you to rerun the inertia test as shown in Figure 72.

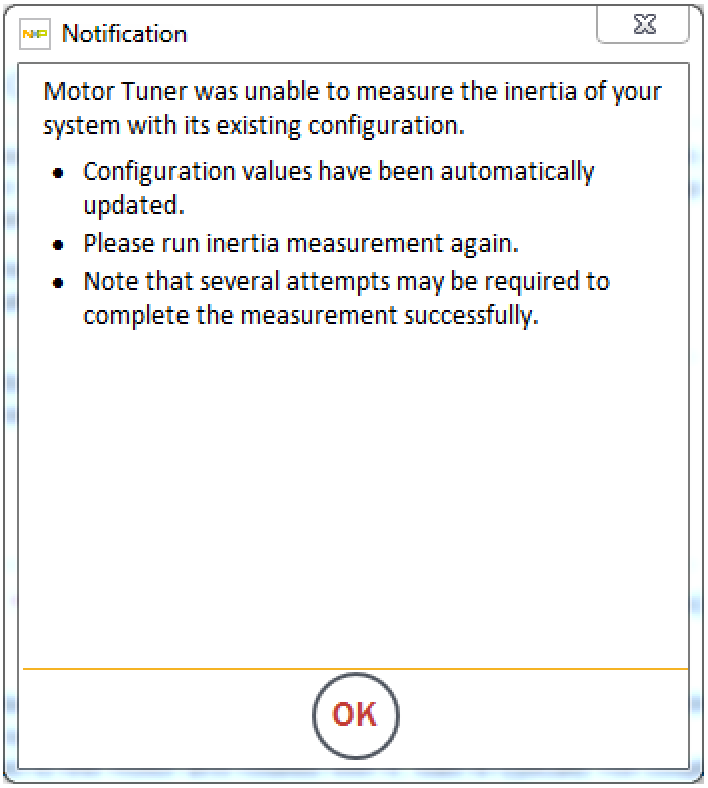


Figure 72. Motor Tuner automatically adjusts the inertia measurement parameters

After clicking OK, you must click the Start Inertia Measurement button again. This process may need to be repeated multiple times depending on your system.

NOTE

When running in a velocity control mode (sensored or sensorless), KMS automatically saves system information upon successful identification of inertia. This primarily means that the system.h file that contains motor & application specific values in the KMS firmware reference project is updated at the conclusion of this step.

- Refer to the KMS API Reference Manual for a more detailed description of the inertia measurement routine.

6.3.3 Parameters

Table 16. Inertia parameters

Parameter	Description	Firmware variable

Inertia	This inertia is not in SI units. Rather, it represents the ability of your system to accelerate. The larger this value, the more torque is required for your system to accelerate.	• speed.config.inertia
Inertia*		• position.config.inertia

*Sensored position

- Refer to the KMS API Reference Manual for a description of the inertia measurement routine.

6.4 Tune Controller (sensored position only)

6.4.1 Screen

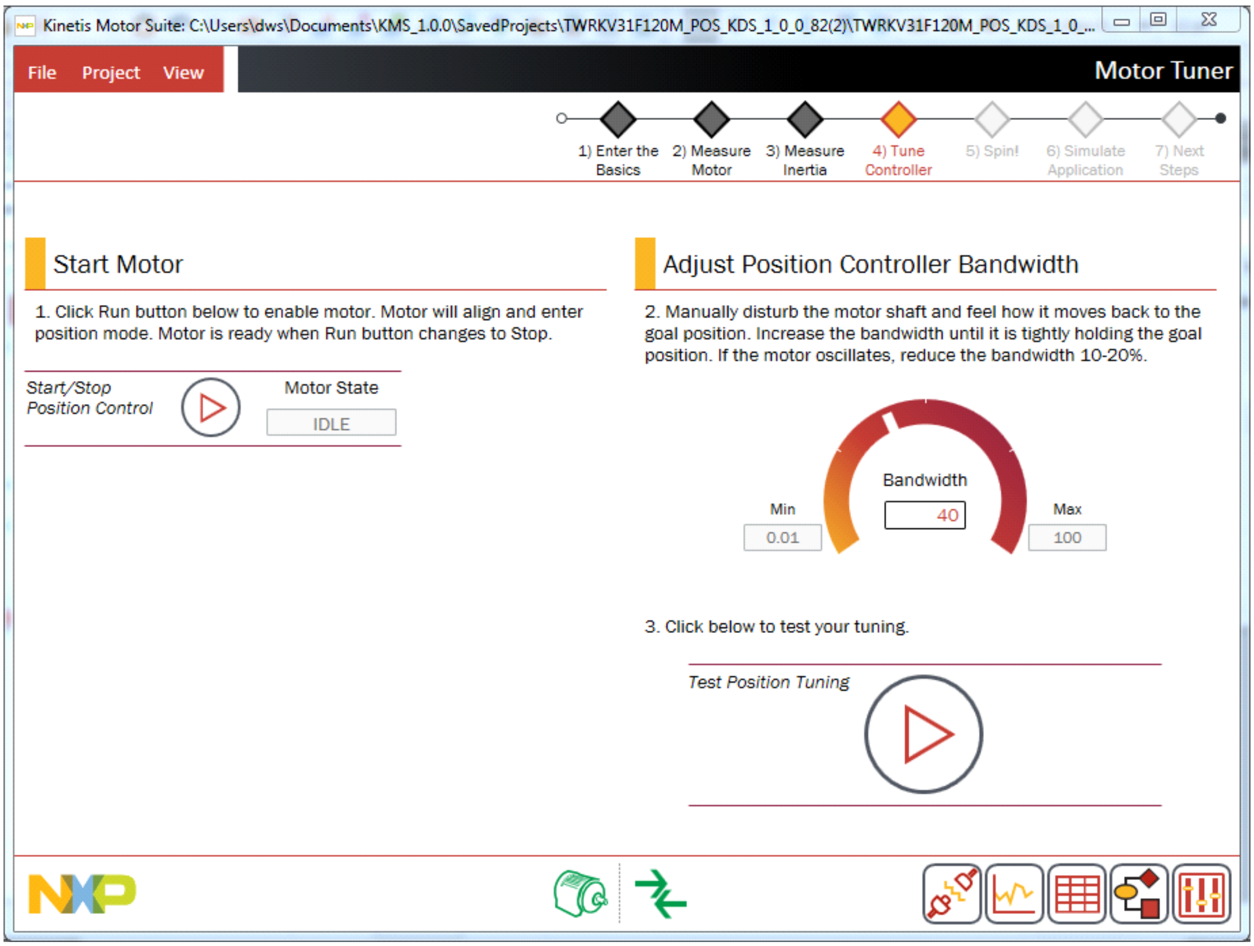


Figure 73. Tune Controller

6.4.2 Description

6.4.2.1 Background

KMS offers differentiated performance and ease of use in motion control. KMS features a proprietary control algorithm, which actively estimates system disturbances and compensates for them in real time.

Disturbances may include:

- Uncertainties (e.g. - resonant mode)
- Nonlinear friction
- Changing loads
- Environmental changes

Using this proprietary algorithm, KMS presents better disturbance rejection and trajectory tracking performance than an industry-standard proportional-integral (PI) position controller. It can also tolerate a wide range of inertia change, enabling improved accuracy and minimized mechanical system duress.

KMS also features a single tuning parameter, bandwidth (Figure 74), which determines the stiffness of the system and dictates how aggressively the system rejects disturbances. Both position and velocity control are achieved with this single parameter.

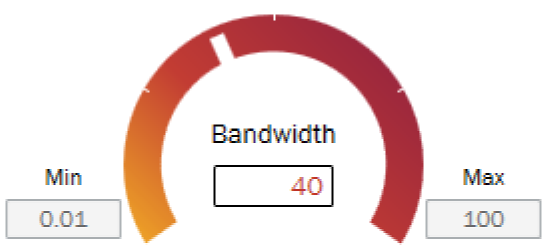


Figure 74. Bandwidth knob

By virtue of single coefficient tuning, KMS allows the user to quickly test and tune position control from soft to stiff response. The bandwidth typically works across the entire dynamic range of an application, reducing complexity and system tuning. Once tuned, the controller works over a wide range of dynamics.

This is in stark contrast to a standard PI position controller, which often has difficulty achieving balance with a corresponding velocity controller and which typically requires different configuration values upon a change in dynamics.

6.4.2.2 Tuning

KMS provides a default value for position regulator bandwidth but this should be optimized for your motor and application using the simple tuning process described here and on the Tune Controller page.

NOTE

The Tune Controller page appears only in Motor Tuner for sensored position control. This is because tuning of the controller is not typically required to proceed through Motor Tuner for sensorless or sensored velocity control. However, for challenging motors or when the motor is in application, tuning similar to that described here for sensored position may be required. See Section 7.2.2, "Speed loop tuning (velocity control only)" for a discussion of velocity controller tuning.

In the "Start Motor" section, click to start position control. The motor aligns and the Motor State changes from "IDLE" to "RUN POSITION".

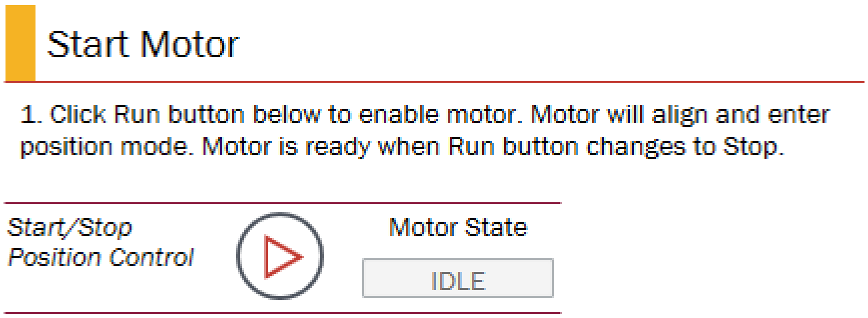


Figure 75. Click to place motor into position control mode

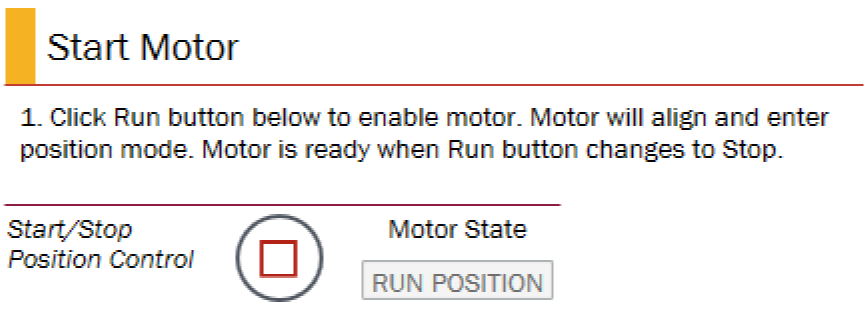


Figure 76. Motor in position control mode

Assess shaft stiffness by gently grabbing the shaft and attempting to turn. If stiffer position control is desired, increase the Bandwidth in increments of 10-20 radians per second (rad/s), assessing shaft stiffness at each setting.

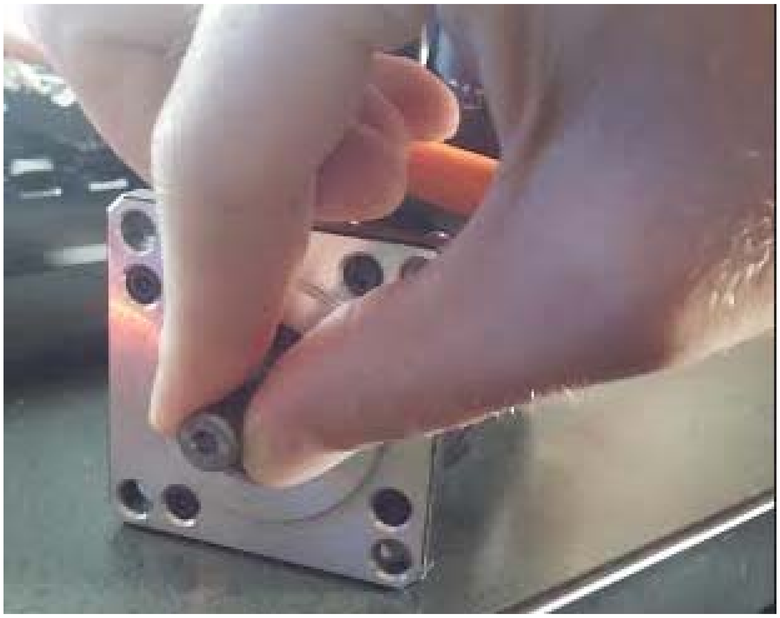


Figure 77. Manual test of stiffness

Once desired shaft stiffness is attained, click to initiate Test Position Tuning to move to the next page of Motor Tuner.

Figure 78. Click to advance and test tuning

NOTE

If the motor is in application or the shaft is otherwise inaccessible for a tactile assessment, the tuning process can occur by assessing step test performance in the KMS Software Oscilloscope. Proceed to the next page in Motor Tuner to see a simple step test and iteratively adjust Bandwidth as described above until the motor is able to precisely track the desired trajectory.

6.4.3 Parameters

Table 17. Tune controller parameters

Parameter	Description	Firmware variable
Start/Stop Position Control	Places motor into position control operating mode. Toggles to allow motor to be placed into idle (stopped) operation.	• user.state = 10 for position control • user.state = 0 for idle
Bandwidth	The motion regulator is tuned by adjusting a single parameter, bandwidth, measured in radians per second (rad/s). Bandwidth determines how aggressively the controller compensates for disturbance. Increasing this value will increase the stiffness of the position controller. If the motor begins to oscillate or vibrate, decrease this value 10-15%.	• position.config.lq20Bw_radps

6.5 Spin!

6.5.1 Screen

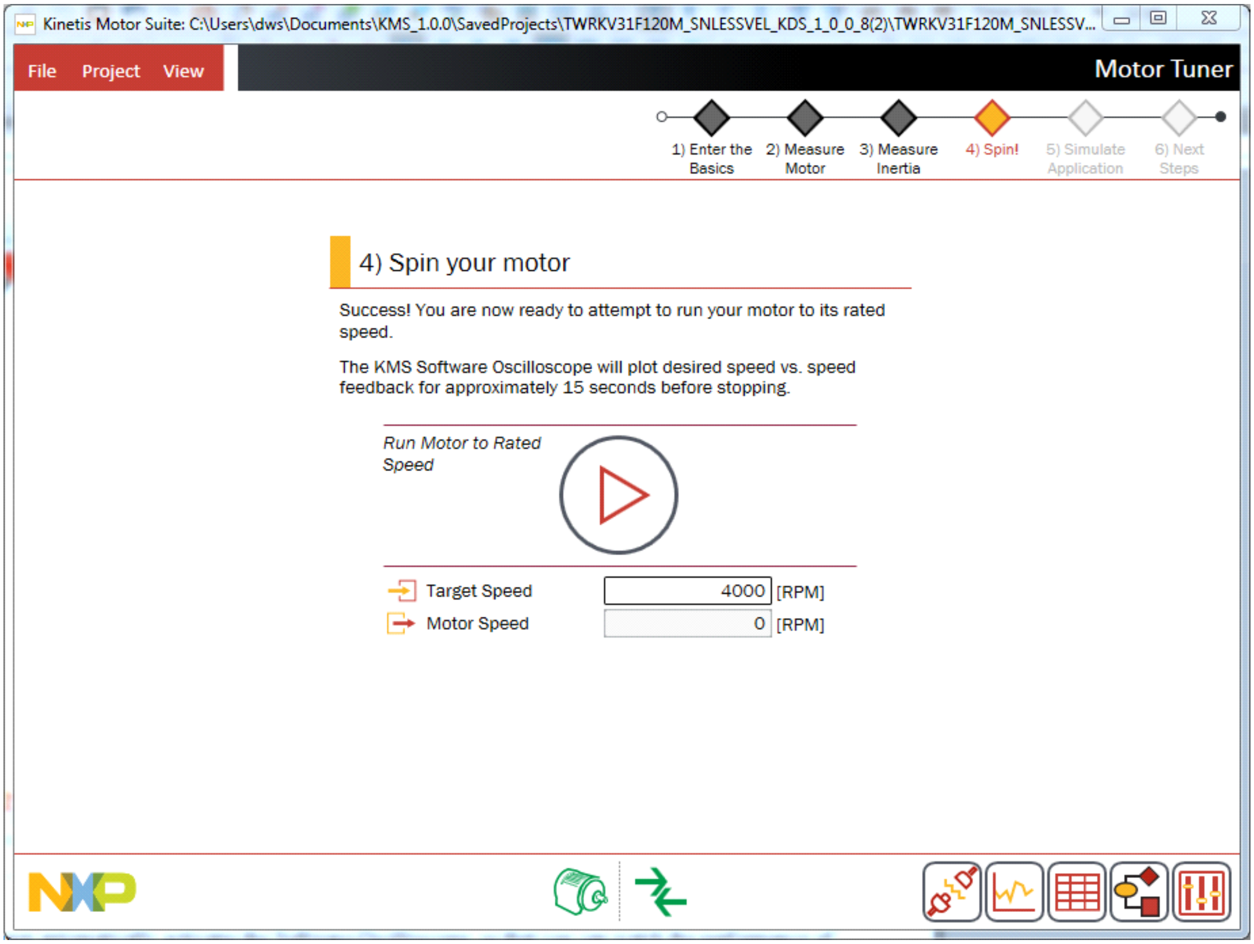


Figure 79. Spin! (velocity control)

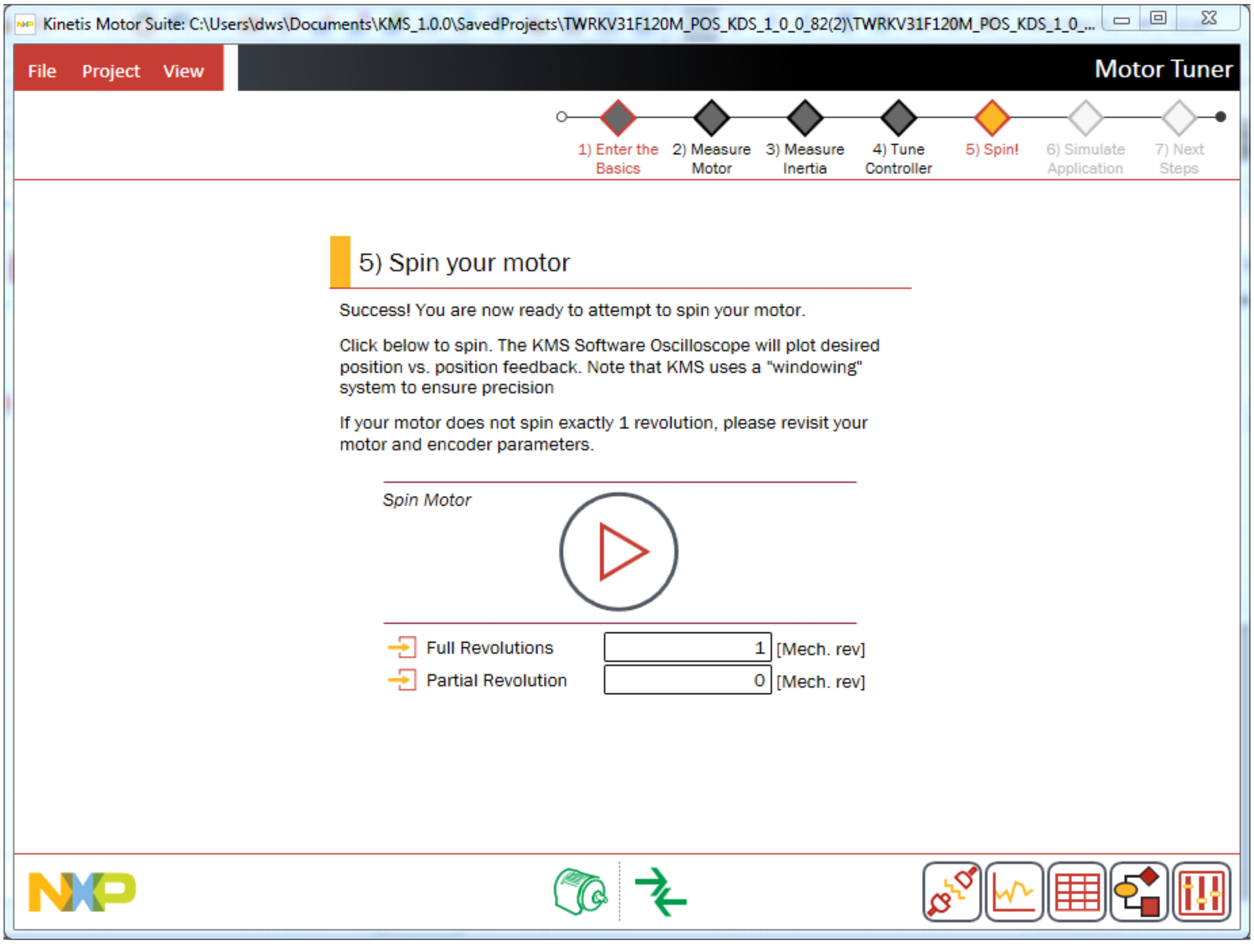


Figure 80. Spin! (position control)

6.5.2 Description

In this step, Motor Tuner runs your motor to the rated speed (velocity control) or for a single revolution (position). If it is not desirable to run your motor in this manner (e.g., if your application is attached and the rated speed of the motor is too high for your application), you may overwrite the default command. This screen is shown in Figure 79 (velocity control, both sensored & sensorless) and in Figure 80 (position control).

Motor Tuner automatically activates the Software Oscilloscope in this step, so that you can watch the performance of your motor as it accelerates to the rated speed. Commanded trajectory is plotted in red and estimated actual trajectory is plotted in green.

For sensorless velocity control, KMS transitions from open to closed loop speed control when it reaches 10% of the motor’s Rated Speed. Prior to this point, the motor is operating in closed loop current control, but speed is not being considered because the signal upon which the speed estimate is assumed too small. The real speed feedback signal in this region is therefore ignored and not displayed. As a result, there appears to be a sharp distinction at the transition point (Figure 81), whereas it is actually just the "turning on" of a real speed feedback signal.

Figure 81. Transition from open to closed loop control

By contrast, when running sensored velocity control, the motor is always in closed loop control due to the external sensor. This results in smoother operation at low speed, which is a chief benefit of sensored control. Nevertheless, the encoder has to catch up when starting from zero, so shows a small divergence from commanded speed near zero.

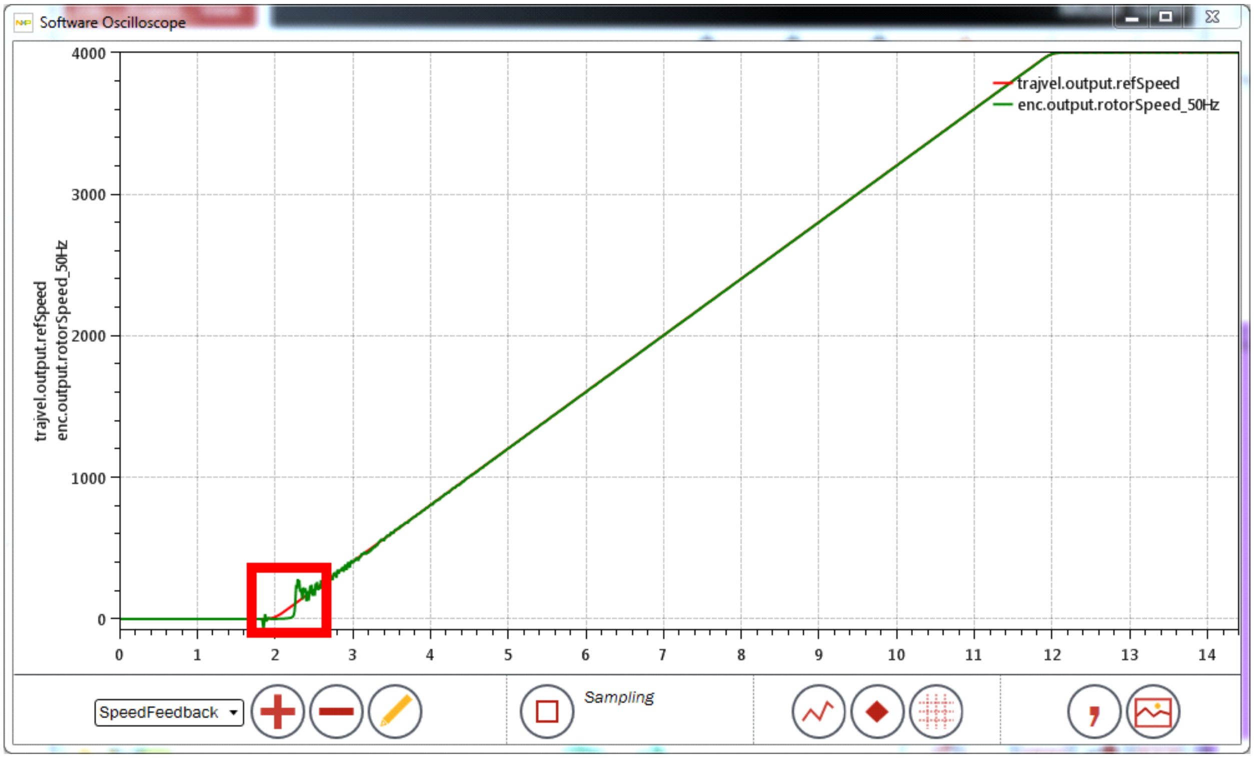


Figure 82. Sensored velocity operation during Spin!

For either velocity control type, the motor spins to the Target Speed and remains at that speed for five seconds. After that, Motor Tuner stops the motor, closes the Software Oscilloscope, and proceeds to the next step.

When running in position control mode, the motor spins a single mechanical revolution while the Software Oscilloscope plots. Approximately two seconds after the motion has stopped, Motor Tuner closes the Software Oscilloscope and proceeds to the next step.

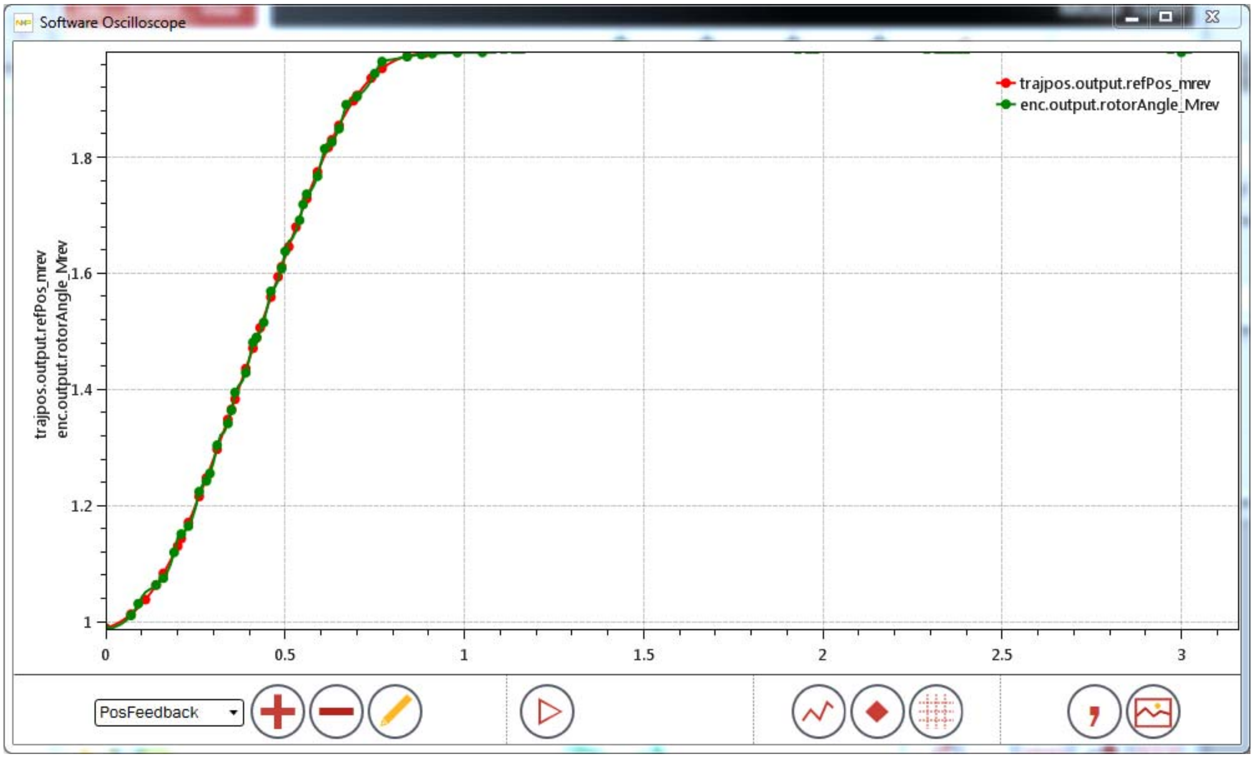


Figure 83. Plot of motor behavior for sensored position control Spin! page

6.5.3 Parameters

Table 18. Spin! elements

Parameter	Description	Firmware variable
Target Speed	Desired speed to which the motor should accelerate to prove functionality. If this speed is too large for the relevant application, it can be reduced.	• user.command.targetSpeed
Motor Speed*	Feedback speed of motor based on sensorless angle estimator.	• est.output.rotorSpeed_50Hz
Motor Speed**	Feedback speed of motor based on quadrature encoder feedback.	• enc.output.rotorSpeed_50Hz
Full Revolutions***	Number of complete revolutions the motor should turn to prove functionality. This number can be adjusted to show a different position movement.	• user.command.posStepInt_mrev
Partial Revolution***	Number of partial revolutions the motor should turn to prove functionality. This number can be adjusted to show a different position movement.	• user.command.posStepFrac_mrev

*Sensorless velocity

**Sensored velocity
***Sensored position

6.6 Simulate application

6.6.1 Screen

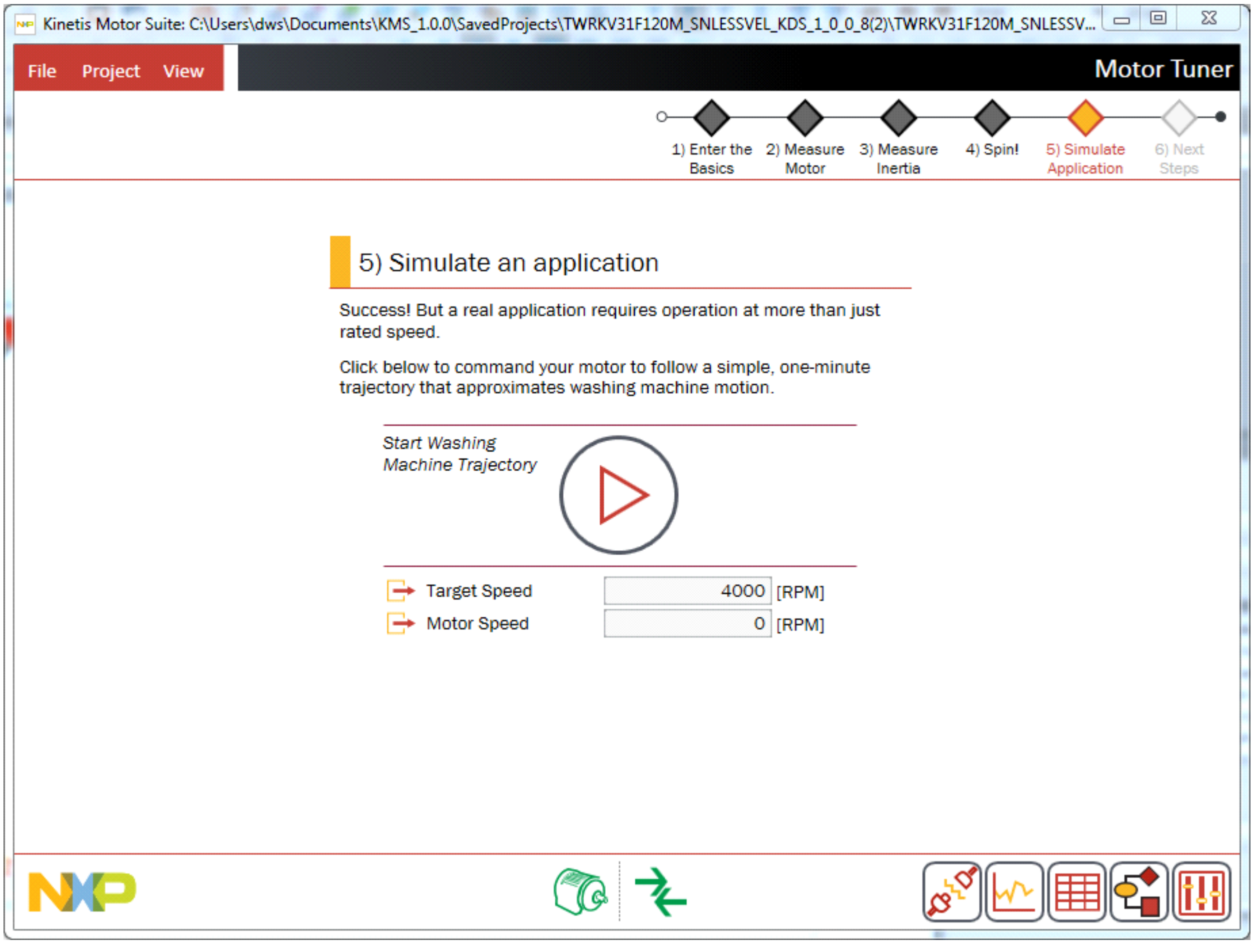


Figure 84. Click to simulate washing machine trajectory

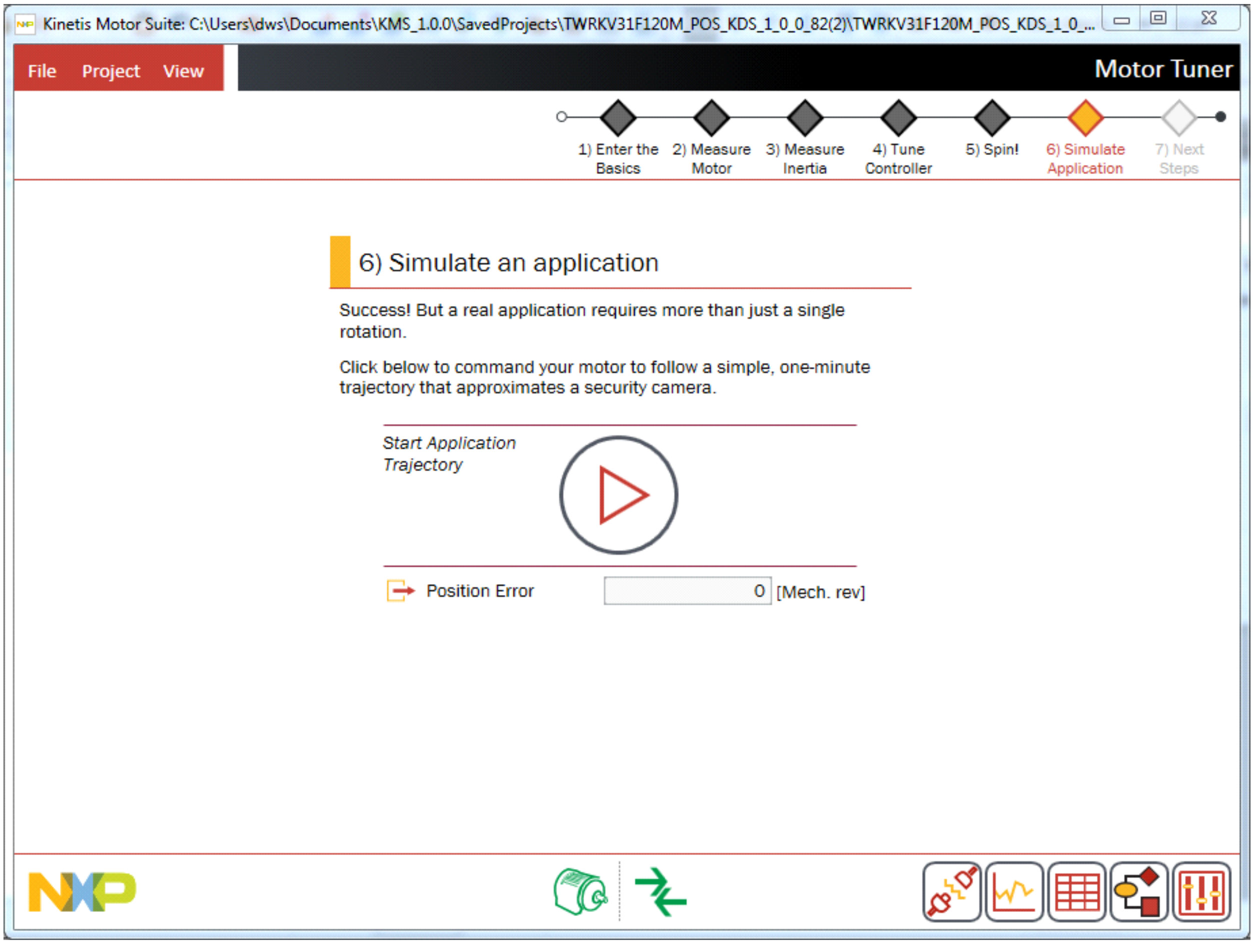


Figure 85. Click to simulate security camera trajectory

6.6.2 Description

Motion Sequence Builder is one of the tools available in KMS. It allows you to easily build complex motion sequences and automatically generate application code. Detailed information about Motion Sequence Builder can be found in Section 9, "Motion Sequence Builder".

KMS comes with a few motion sequence examples.

The primary velocity control example simulates the operation of a simple washing machine. The motor:

- “agitates” by ramping to a certain speed, then reversing direction to reach the same speed in the opposite direction (repeating this behavior several times)
- “spin cycles” by ramping up to a speed twice the agitation speed
- comes to a halt and concludes the motion sequence.

The primary position control example simulates the operation of a security camera. The motor:

- slowly pans back and forth

- quickly pans to a given location when motion is detected
- slowly completes the surveillance
- returns to the initial position

Motor Tuner automatically launches the Software Oscilloscope so that you can observe your motor's performance as it executes the simulation (Figure 86 for velocity control; Figure 87 for position control).

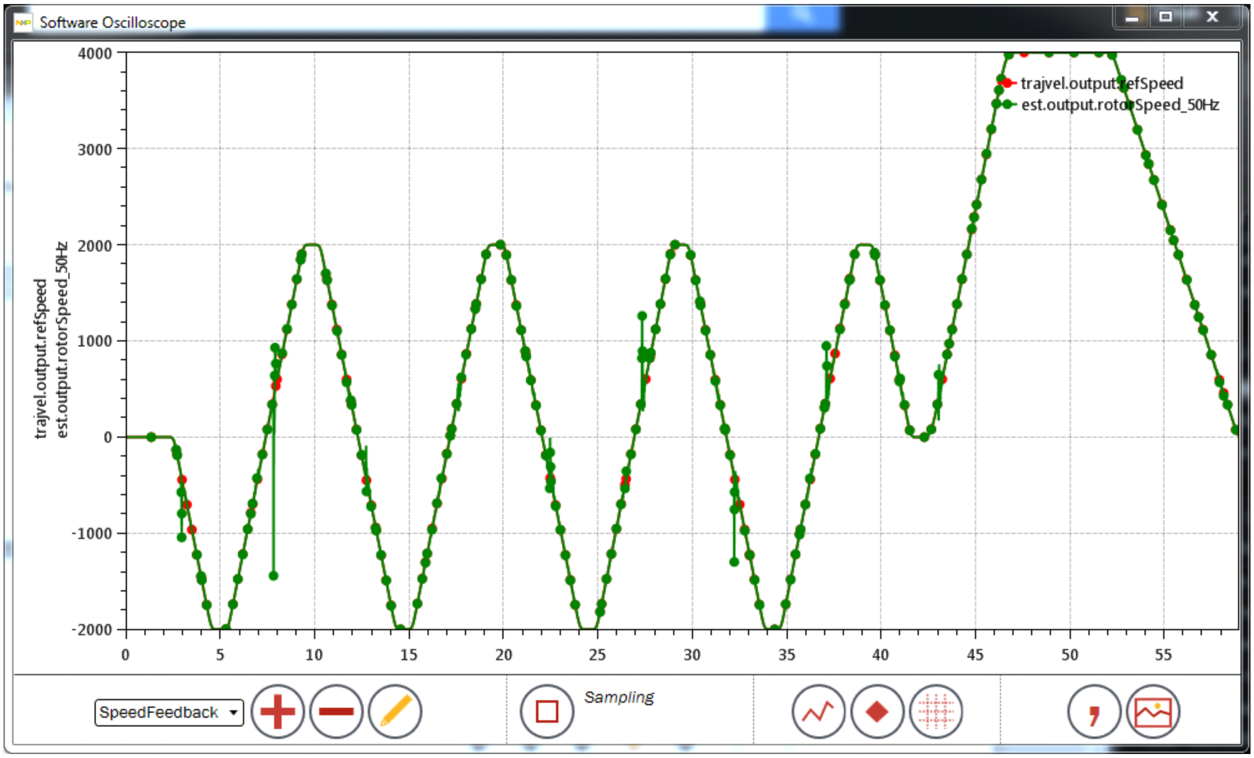


Figure 86. Software Oscilloscope displays washing machine simulation (velocity control)

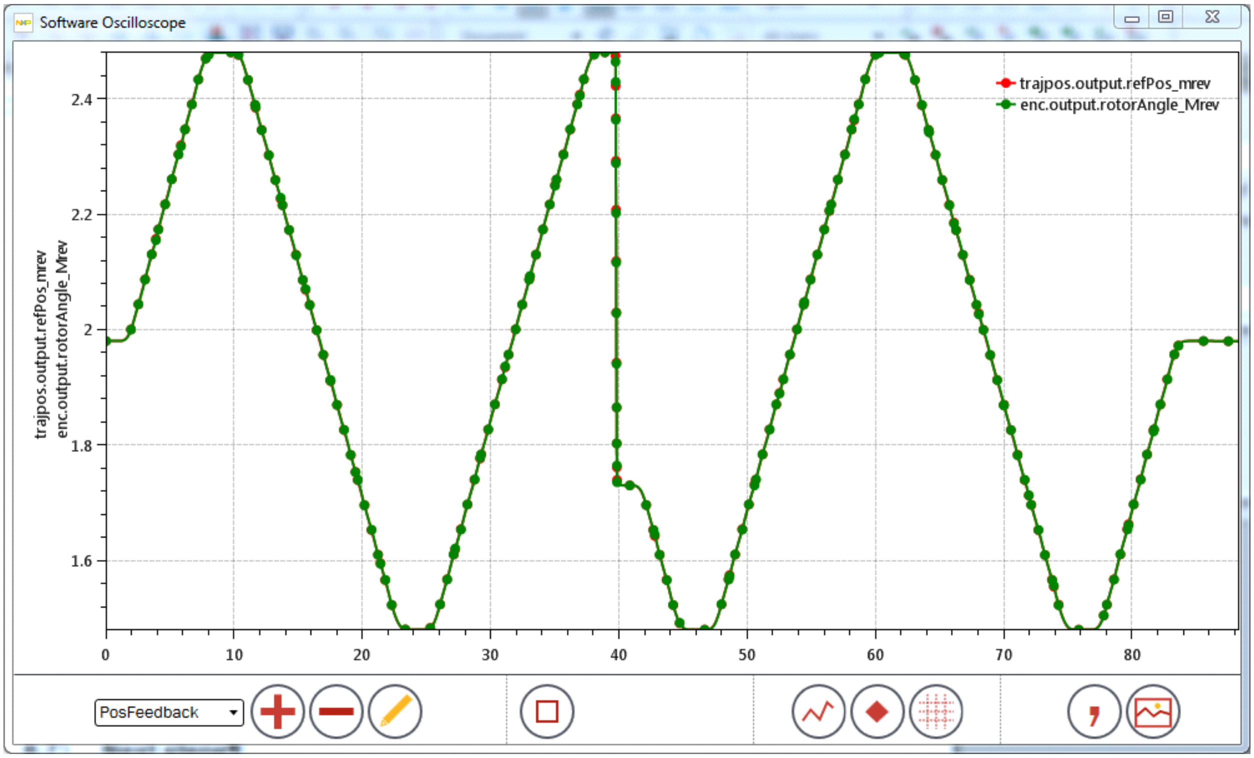


Figure 87. Software Oscilloscope displays security camera simulation (position control)

Upon completion of the motion sequence, Motor Tuner closes the Software Oscilloscope and advances the user to the next step.

6.6.3 Parameters

Table 19. Simulate application parameters

Parameter	Description	Firmware variable
Target Speed	Current goal speed for the motor. This value is updated as the motion sequence is executed.	• trajvel.config.targetSpeed
Motor Speed	Feedback speed of motor based on sensorless angle estimator.	• est.output.rotorSpeed_50Hz
Motor Speed*	Feedback speed of motor based on quadrature encoder feedback.	• enc.output.rotorSpeed_50Hz
Position Error**	Difference between the goal position and the current position. This value represents how well the position controller is tracking the changing position reference.	• position.output.posErr_mrev

*Sensored velocity
**Sensored position

6.7 Next steps

6.7.1 Screen

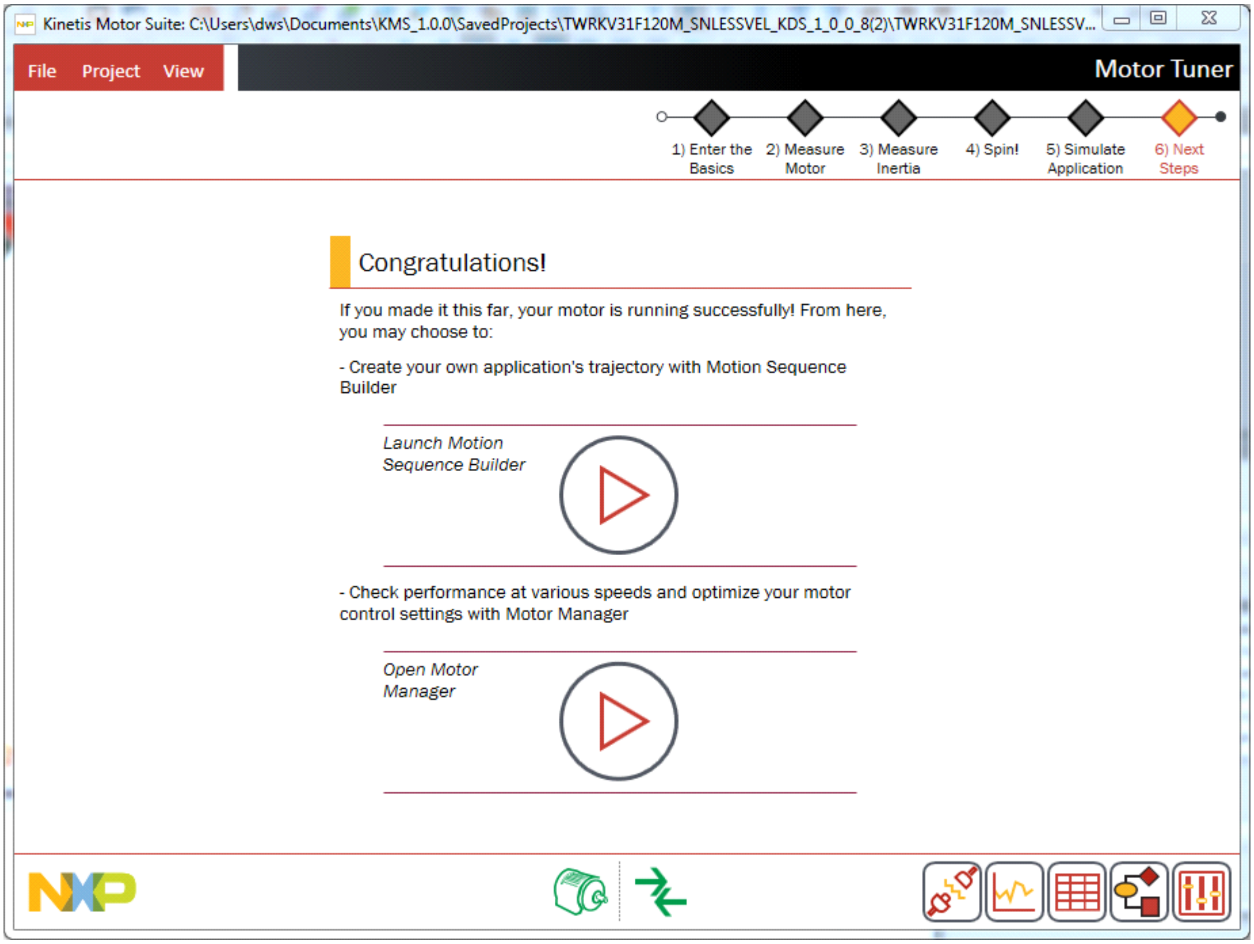


Figure 88. Next steps (velocity control)

6.7.2 Description

After successfully identifying motor & inertia, then spinning simply and in a more application-focused manner, you can be confident that fundamental motor operation is sound. Given this, Motor Tuner suggests that you proceed down one of two paths, as shown in Figure 88:

1. Motion Sequence Builder: Choose this path to build complex motion sequences and automatically generate application code. Detailed information can be found in Section 9, "Motion Sequence Builder".
2. Motor Manager: Choose this path to fine-tune your motor performance and operation. Detailed information can be found in Section 7, "Motor Manager"

7 Motor Manager

Motor Manager provides a superset of Motor Tuner capabilities, offering greater configuration options.

Table 20. Motor Manager

Page	Applicable control types	Description
Identify	All	Aggregates the identification steps of Motor Tuner: • entering basic information, • measuring electrical characteristics of motor • measuring inertia • updating KMS firmware reference project with new values
Speed Control	All	Facilitates placing motor into speed control mode and specifying different speeds and trajectories
Position Control	Sensored position	Facilitates placing motor into position control mode and specifying point to point movements
Motion Sequences	All	Allows running of complex motion sequences defined in Motion Sequence Builder as well as test calculations that can be used to build up a motion sequence
Torque Control	Sensorless velocity Sensored velocity	Enables the motor to run in a mode where current is controlled but speed is not
Protection & Hardware	All	Defines the software protections that are employed as well as the device settings to enable modification for custom hardware
Advanced Tuning	All	Provides access to variables of use to expert users: firmware operating frequencies, current loops, etc.
Dashboard	All	Aggregates and updates the values of key system variables
CPU Utilization	All	Tracks and displays processor utilization

7.1 Identify

Use this page (Figure 89) to identify and save key motor and system parameters required by KMS. This page effectively aggregates the first three steps of Motor Tuner.

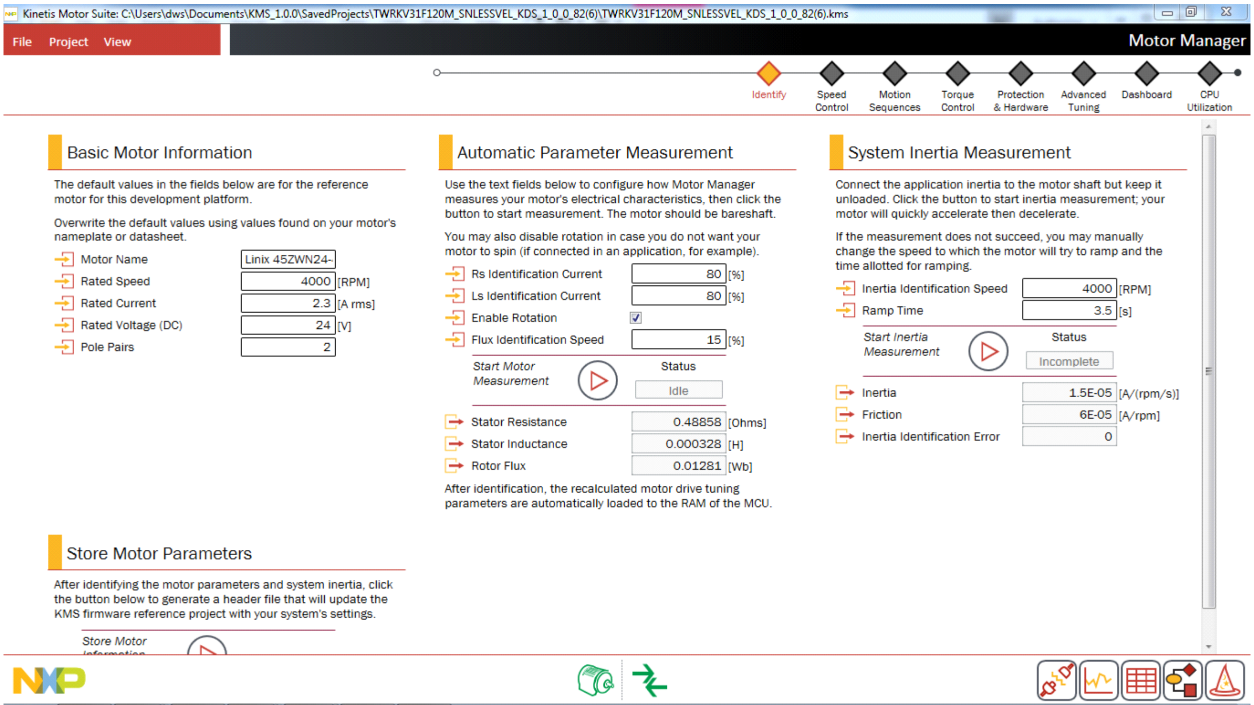


Figure 89. Identify page

7.1.1 Basic motor information

7.1.1.1 Screen

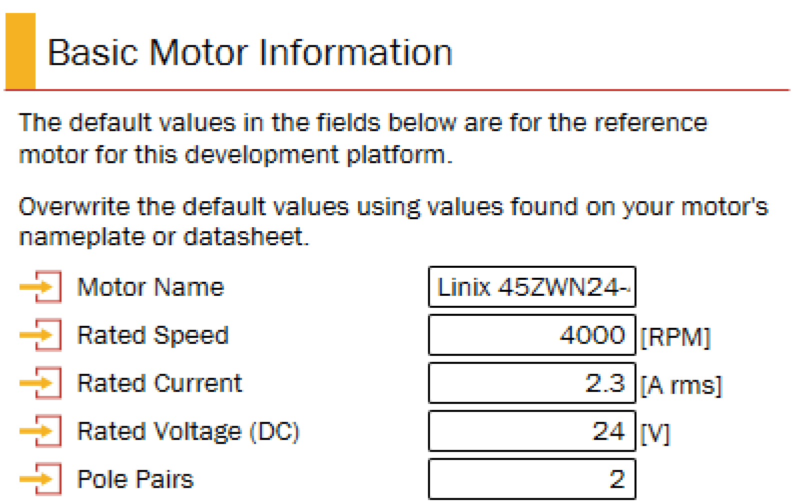


Figure 90. Basic motor information (sensorless velocity)

7.1.1.2 Description

KMS requires the user to enter certain basic information about the selected motor. This information is shown in Figure 90. Basic motor parameters can be found on the motor's name plate, or in the motor data sheet. KMS uses the basic motor parameters to conduct Automatic Parameter Measurement, and to set the hardware configuration and protection settings.

7.1.1.3 Parameters

Table 21. Basic motor parameters

Parameter	Description	Firmware variable
Motor Name	Enter a name for the motor for convenience. This is not required.	• N/A
Rated Speed	The maximum speed at which the rated torque can be delivered.	• N/A
Rated Current	The RMS (not peak) current for which the motor is rated.This value is used to establish the maximum output of the speed controller.	• N/A
Rated Voltage (DC)	Rated DC bus voltage of the motor. If specified on datasheet as VAC, multiply by square root 2 to arrive at VDC value.	• N/A
Pole Pairs	The number of pairs of magnetic poles in one mechanical revolution.	• N/A
Encoder Lines*	Number of lines (pulses) on encoder wheel	• N/A

*Sensored control (velocity or position)

7.1.2 Automatic parameter measurement

7.1.2.1 Screen

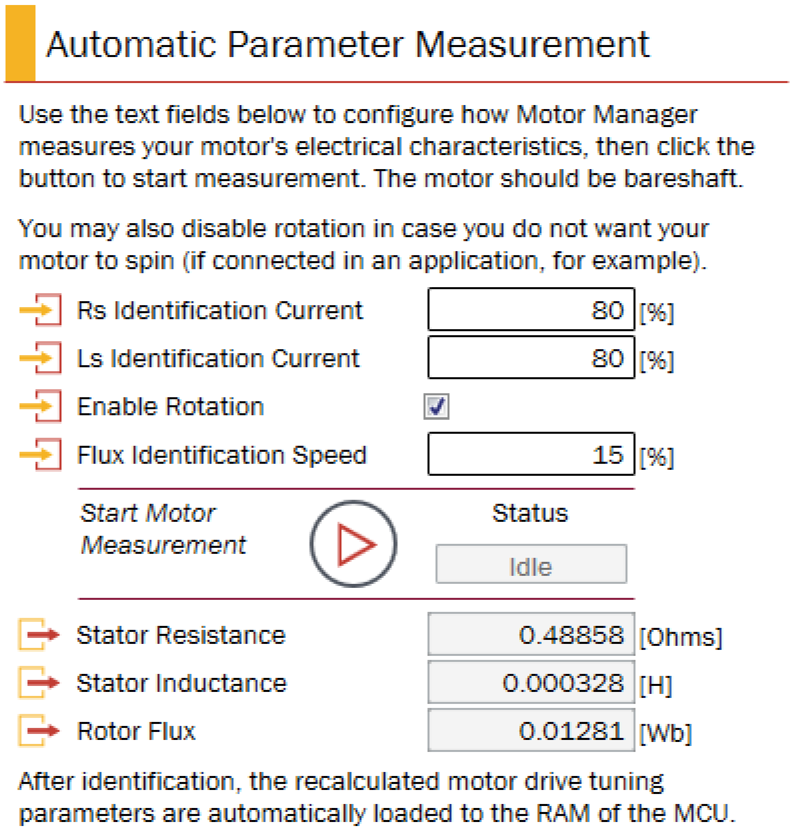


Figure 91. Automatic parameter measurement

7.1.2.2 Description

KMS automates the process of determining key characteristics of a motor. After the Basic Motor Parameters are entered, the user initiates Automatic Parameter Measurement (Figure 91). This step should be performed with a bare motor shaft (nothing attached to shaft). The motor identification status indicator tracks the progress of the parameter measurement process by displaying the parameter currently in the process of being identified.

Once complete, the values for Stator Resistance, Stator Inductance, and Rotor Flux are displayed. KMS may automatically adjust configuration parameters if it encounters a motor that is difficult to identify. Users may also adjust these parameters (see Table 22 for a description of configuration parameters).

These identified motor values are critical to the calculation of proper scaling, configuration and tuning parameters of the motor drive algorithm. After identification, the new drive values are calculated and automatically downloaded to the MCU RAM.

- Refer to the KMS API Reference Manual for a more detailed description of the motor measurement routines.

7.1.2.3 Parameters

Table 22. Automatic parameter measurement parameters

Parameter	Description	Firmware variable
RS Identification Current	Percentage of the rated current used to energize the stator and measure stator resistance. KMS may automatically adjust this value if it encounters a motor that is difficult to identify.	• scm.config.relativeRsCurrent
LS Identification Current	Percentage of the rated current used to energize the stator and measure stator inductance. KMS may automatically adjust this value if it encounters a motor that is difficult to identify.	• scm.config.relativeLsCurrent
Enable/Disable rotation	Disable rotation when the motor cannot be decoupled from the application (as with a compressor). This requires manual entry of a rotor flux value on the Advanced Tuning page.	• N/A
Flux Identification Speed	During Flux Identification, the motor spins at this percentage of the rated speed. KMS may automatically adjust this value if it encounters a motor that is difficult to identify.	• scm.config.relativePmFluxFrequency
Stator Resistance	The per phase winding resistance of your motor	• scm.output.statorRes
Stator Inductance	The per phase winding inductance of your motor at rated current	• scm.output.statorInd
Rotor Flux	The rotor flux (or permanent magnet flux linkage) of your motor	• scm.output.pmFlux

7.1.3 System inertia measurement

7.1.3.1 Screen

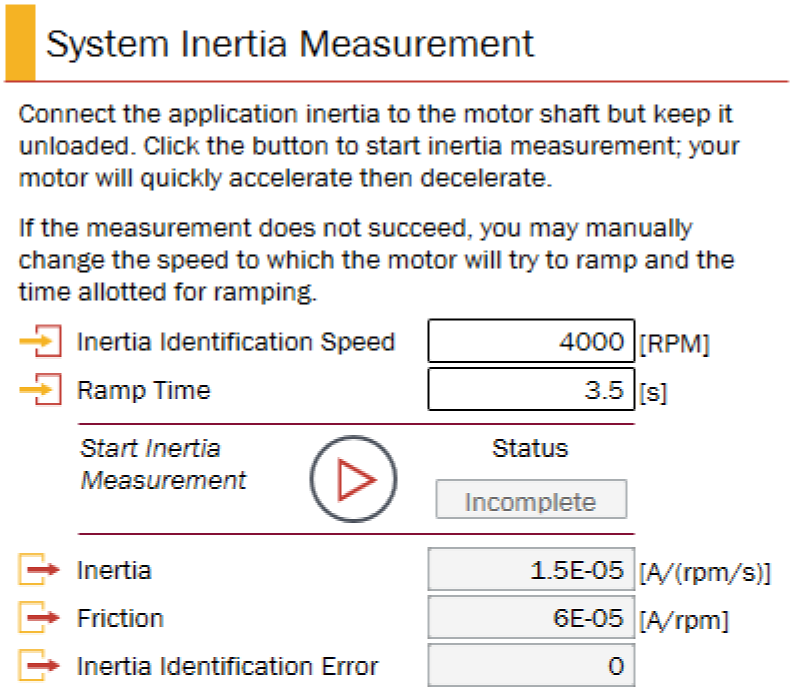


Figure 92. System inertia measurement

7.1.3.2 Description

After the motor parameters are identified, the user performs *System Inertia Measurement* (Figure 92). For this step, the user attaches the application inertia to the shaft, but keeps the motor unloaded. For example, in a washing machine application, the empty drum is the inertia whereas the clothes are the load; the drum should be attached for inertia measurement but the clothes should not be.

Inertia is an important input to KMS' advanced motion controller. The controller must provide enough torque to overcome the system's inertia.

Users may specify the Inertia Identification Speed and Ramp Time. Table 24 describes these inputs. KMS automatically adjusts these parameters if it encounters an Inertia Identification Error or motor fault.

If KMS encounters an Inertia Identification Error, the Inertia Identification Speed and Ramp Time can be manually adjusted as described in Table 23. In Motor Tuner, these configuration updates occur automatically.

Table 23. Common inertia identification adjustments

Error code	2003	2004		2006
Meaning	Bad estimation value	Process timeout		Motor stops during test
Motor behavior	N/A	Motor spins	Motor starts slowly	N/A
Solution	Decrease Ramp Time	Decrease Inertia Identification Speed	Decrease Ramp Time	Decrease Ramp Time
Commonly occurs in these applications	Automotive pumps	Washing machines	Compressors	High friction/ cogging force

- Refer to the KMS API Reference Manual for a more detailed description of the inertia measurement routine.

7.1.3.3 Parameters

Table 24. Automatic parameter measurement parameters

Parameter	Description	Firmware variable
Inertia Identification Speed	Speed that the motor attempts to reach during the inertia identification process. Ensure that this speed is greater than 5 times the Startup Speed Threshold (sensorless operation). KMS might automatically adjust this value as part of the Inertia Identification process.	• inertia.config.goalSpeed
Ramp Time	Rate at which the current will be increased as part of the Inertia identification process. Decreasing the ramp time value accelerates the motor more quickly. It is advised for motors with large friction that this should be reduced. KMS might automatically adjust this value as part of the inertia identification process.	• inertia.config.torqueRampTime_sec
Inertia	This inertia is not in SI units. Rather, it represents the ability of your system to accelerate. The larger this value, the more torque is required for your system to accelerate.	• speed.config.inertia
Inertia*		• position.config.inertia
Friction	Automatically identified system viscous friction. Provided as information.	• speed.config.friction
Friction*		• position.config.friction
Inertia Identification Error	Error code from firmware indicating that inertia measurement has not successfully completed. See Table 23 for typical error codes and recommended remedies.	• inertia.output.error

*Sensored position only

7.1.4 Store motor parameters

7.1.4.1 Screen

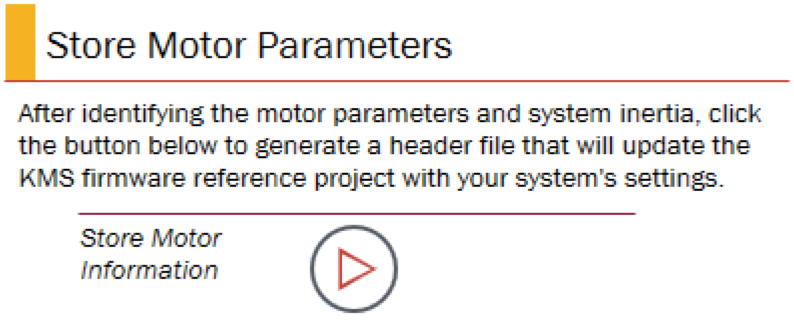


Figure 93. Store motor parameters

7.1.4.2 Description

This feature (Figure 93) stores the motor parameters and system inertia and friction values in a header file (system.h) that can be used in the KMS firmware reference project. This is an important step to ensure that the firmware is updated to reflect your motor's operation, not that of the default motor.

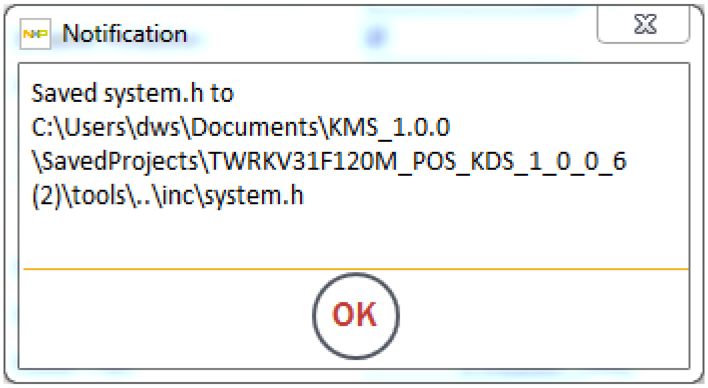


Figure 94. Example notification that system settings have been updated

7.2 Speed control

Use this page (Figure 95) to spin your motor, tune the speed controller, customize motor startup, and specify how the motor should move from point A to point B. The page differs slightly according to whether you are operating in velocity or position control.

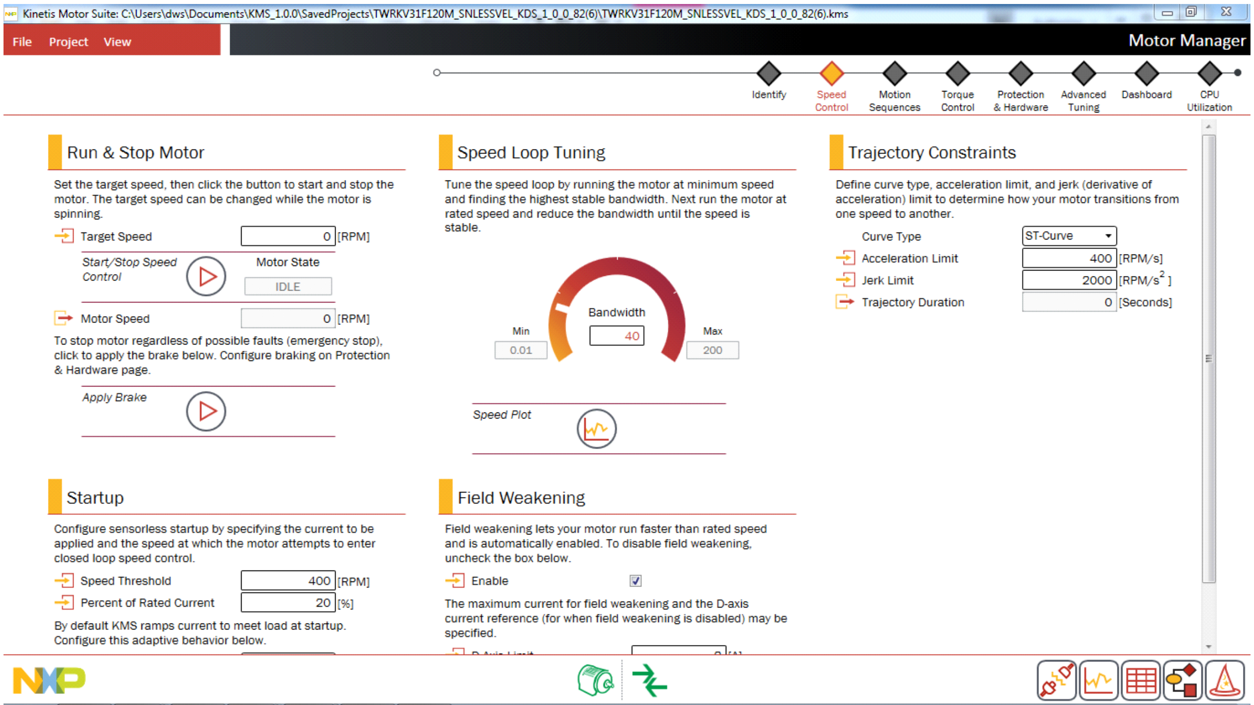


Figure 95. Speed control page (velocity control)

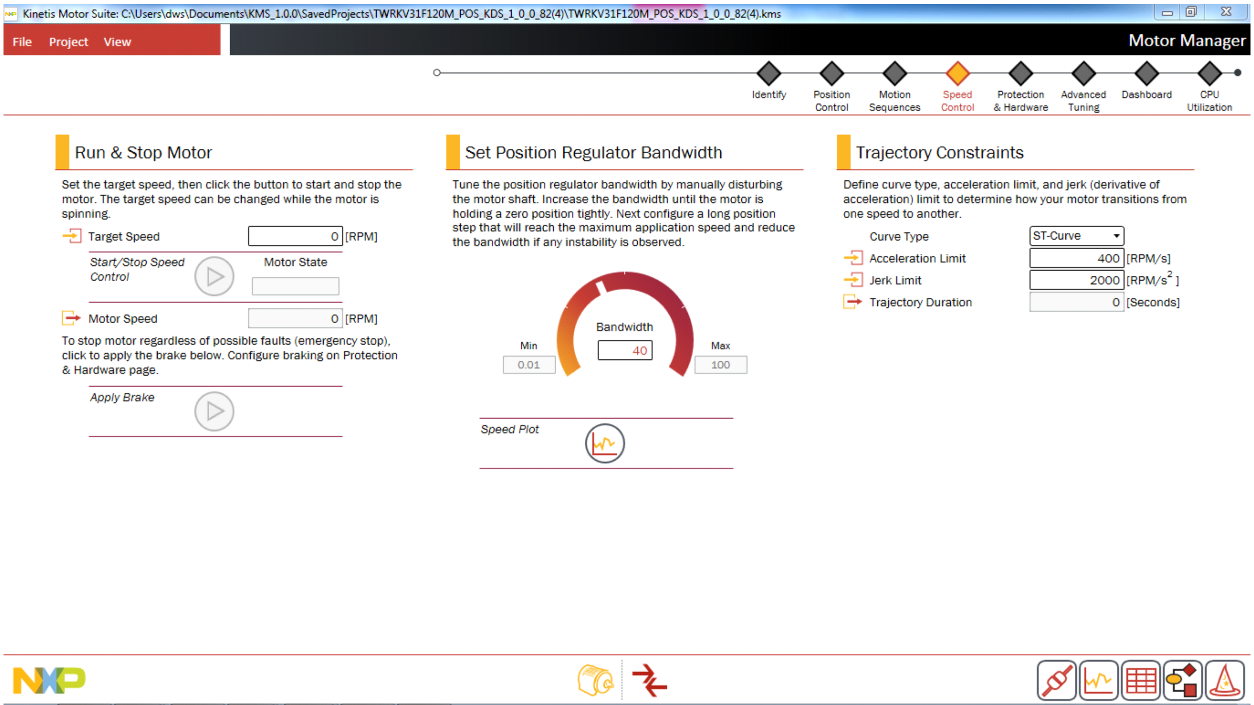


Figure 96. Speed control page (position control)

7.2.1 Run & stop motor

7.2.1.1 Screen

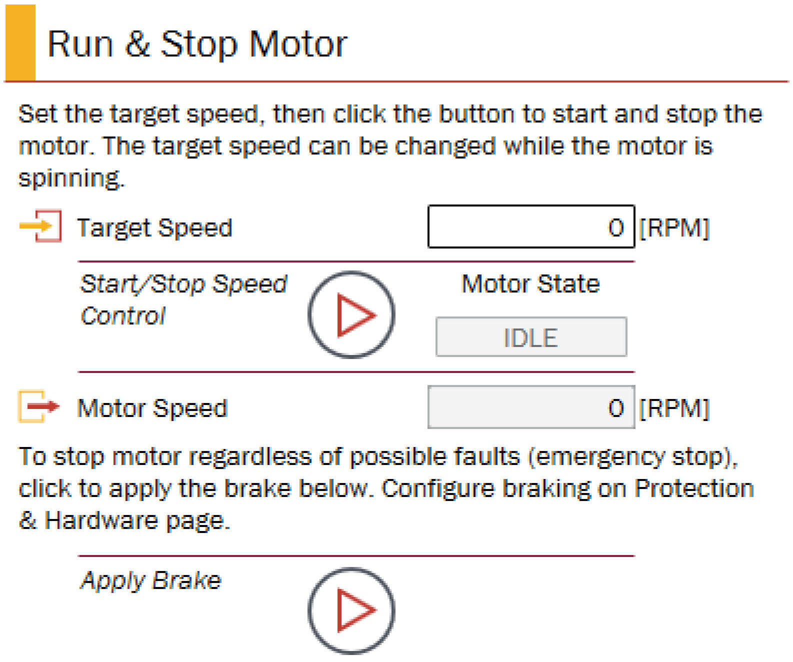


Figure 97. Run & stop motor

7.2.1.2 Description

Enter a target speed, and use the button to start or stop the motor. From here you can also view the actual motor speed, and apply a brake. There are four different types of brakes available in the system (see Section 7.6.2, "Braking"). The braking configuration is specified on the Protection & Hardware page (Section 7.6, "Protection & hardware").

7.2.1.3 Parameters

Table 25. Run & stop motor parameters

Parameter	Description	Firmware variable
Target Speed	Desired motor speed, measured in revolutions per minute (RPM).	• user.command.targetSpeed
Start/Stop Speed Control	Places motor into speed control operating mode. Toggles to allow motor to be placed into idle (stopped) operation.	• user.state = 7 for speed control • user.state = 0 for idle
Motor Speed	Feedback speed of motor based on sensorless angle estimator.	• est.output.rotorSpeed_50Hz

Motor Speed*	Feedback speed of motor based on quadrature encoder feedback.	• enc.output.rotorSpeed_50Hz
Apply Brake	Places motor into brake mode, which stops the motor in accordance with user-specified braking configuration. Toggles to allow motor to be placed into (stopped) operation.	• user.state = 9 to brake • user.state = 0 for idle

*Sensored control (velocity or position)

7.2.2 Speed loop tuning (velocity control only)

7.2.2.1 Screen

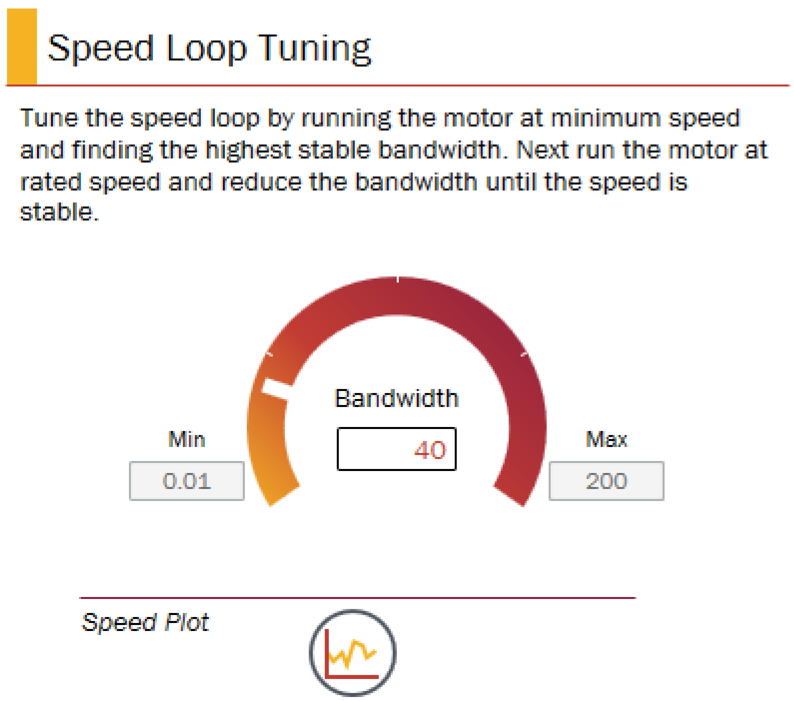


Figure 98. Set speed regulator bandwidth

7.2.2.2 Description

7.2.2.2.1 Background

KMS offers differentiated performance and ease of use in motion control. KMS features a proprietary control algorithm, which actively estimates system disturbances and compensates for them in real time. Disturbances may include:

- Uncertainties (e.g. - resonant mode)
- Nonlinear friction
- Changing loads

- Environmental changes

KMS presents better disturbance rejection and trajectory tracking performance than an industry standard PI speed controller. It can tolerate a wide range of inertia change, enabling improved accuracy and minimized mechanical system duress.

KMS also features a single tuning parameter, bandwidth (Figure 98), which determines the stiffness of the system and dictates how aggressively the system rejects disturbances. With single coefficient tuning, KMS allows the user to quickly test and tune velocity control from soft to stiff response. The bandwidth typically works across the entire variable speed and load range of an application, reducing complexity and system tuning. Once tuned, the controller works over a wide range of speeds and loads.

7.2.2.2.2 Tuning

KMS provides a default value for speed regulator bandwidth but this can be adjusted for a specific or challenging application using the following tuning process:

1. With the motor running at 15% of rated speed or above, click to open the Speed Plot and click the Run button to start sampling.
2. Disturb the rotor speed by introducing a light load shock (by gently grabbing the shaft or resisting the motion on the attached inertia).
3. If the rotor speed shows significant oscillation, increase the Bandwidth in increments of 10-20 radians per second (rad/s) and repeat the light load shock test. (You may need to adjust the sampling rate and y-axis range to zoom in on the motor speed behavior, see Section 8.1, "Software Oscilloscope")
4. Increase Bandwidth until motor speed is regulated despite load shock but the motor remains stable.
5. While using the identified Bandwidth, increase the Target Speed to the motor's Rated Speed.
6. Verify that the motor responds well to light load shock and operates stably at Rated Speed with this Bandwidth. If the motor is unstable, reduce Bandwidth by 10 or 20 rad/s and repeat verification
7. If the application requires a speed higher than the rated speed, increase the Target Speed gradually, checking the speed regulation at each step. For very high speeds (deep Field Weakening), a lower Bandwidth may be required for stable operation.
8. Once the KMS speed controller has been tested and has demonstrated control at the extremes, the motor runs across the operating range.

The motor is now tuned. The motor speed feedback is displayed and the motor should spin to the desired speed. The motor can now be run with the normal application loads and tested at various speeds.

7.2.2.3 Parameters

Table 26. Run & stop motor parameters

Parameter	Description	Firmware variable
Bandwidth	The motion regulator is tuned by adjusting a single parameter, bandwidth, measured in radians per second (rad/s). Bandwidth determines how aggressively the controller compensates for disturbance. Increasing this value increases the stiffness of the speed controller. If the motor begins to oscillate or vibrate, decrease this value 10-15%.	• speed.config.Iq20Bw_radps

7.2.3 Trajectory constraints

7.2.3.1 Screen

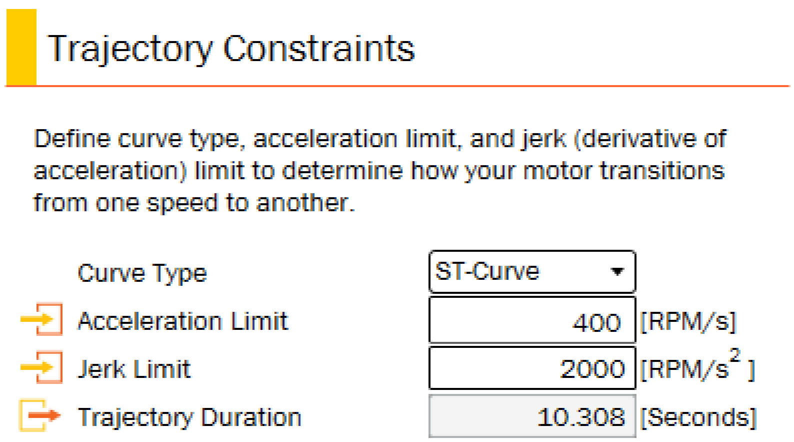


Figure 99. Trajectory constraints

7.2.3.2 Description

KMS includes a motion profile generator that generates constraint-based, time-optimal motion trajectory curves. It removes the need for look-up tables and runs in real-time to generate the desired motion profile. It supports basic ramp profiles as well as advanced s-curve and a proprietary st-curve. The proprietary st-curve features a continuous jerk to provide additional smoothing on the trajectory. Trapezoidal, s-curve, and st-curve are compared in Figure 100 and Figure 101.

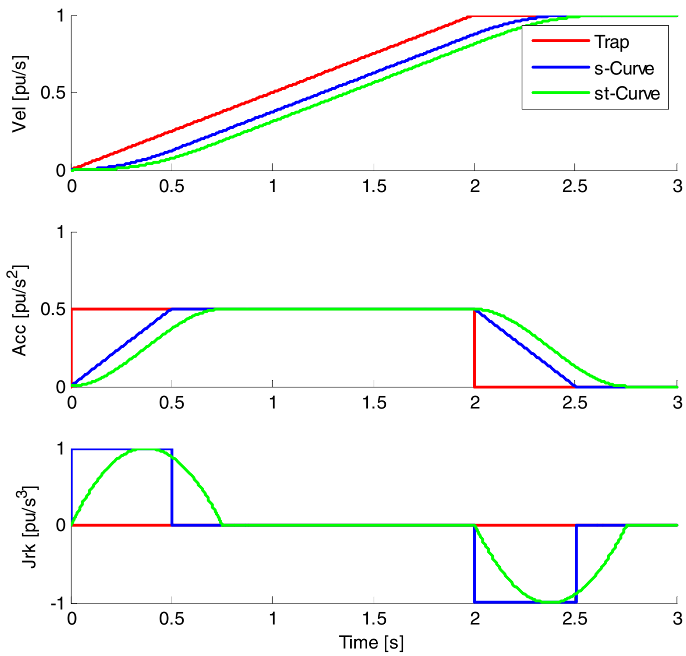
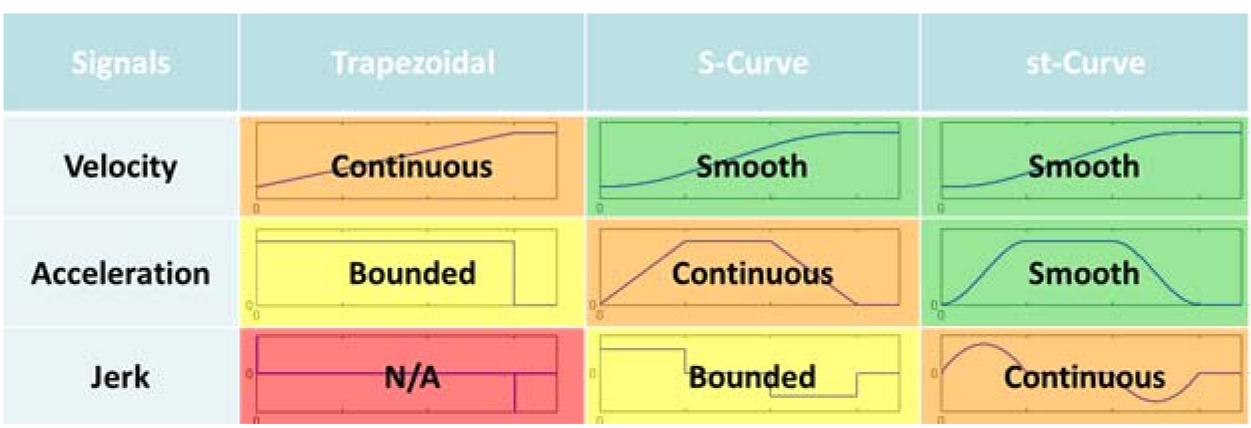


Figure 100. Differences among available curve types

The st-curve represents the smoothest motion, which is critical for systems that are sensitive to large amounts of jerk. Jerk represents the rate of change of acceleration. A larger jerk increases the acceleration at a faster rate. Steps, or sharp movement between two speeds, can cause systems to oscillate. The bigger the step in speed, the greater this tendency for the system to oscillate. Control over jerk can round the velocity corners, reducing oscillation. As a result, acceleration can be set higher. Controlling the jerk in your system leads to less mechanical stress on your system components and can lead to better reliability and less failing parts.



Signals	Trapezoidal	S-Curve	st-Curve
Velocity	Continuous	Smooth	Smooth
Acceleration	Bounded	Continuous	Smooth
Jerk	N/A	Bounded	Continuous

Figure 101. Summarized differences among available curve types

You can easily set up and test speed trajectories, as shown in Figure 99. The Trajectory Duration changes based on the Curve Type, Acceleration Limit, and Jerk Limit. The selected Curve Type is used when you run a Plan from the Motion Sequences page.

NOTE

KMS does not make a distinction between acceleration and deceleration in velocity control because each transition between different speeds can be configured with a different limit on the rate of change of speed. The direction of the speed transition does not matter.

7.2.3.3 Parameters

Table 27. Trajectory constraints parameters

<table>
<tr><th>Parameter</th><th>Description</th><th>Firmware variable</th></tr>
<tr><td>Curve Type</td><td rowspan="2">The user selects from three curve types:
• Trapezoidal enables instantaneous transitions; constant acceleration. Note that the acceleration and Jerk limits do not have an impact on this profile.
• s-curve linearly increases the acceleration until it reaches the acceleration limit.
• st-curve, LineStream's proprietary curve type, linearly increases the acceleration and jerk until it reaches the user-specified limits, enabling the smoothest possible motion.</td><td>• trajvel.config.curve</td></tr>
<tr><td>Curve Type*</td><td>• trajpos.config.curve</td></tr>
<tr><td>Acceleration Limit</td><td>Acceleration is the rate of change of speed. This value represents the maximum allowed rate of change of speed when using trapezoid, s-curve, and st-curve.</td><td>• user.command.limitAcc</td></tr>
<tr><td>Jerk Limit</td><td>Jerk is the rate of change of motor acceleration. This value represents the maximum allowed rate of change of acceleration when using s-curve and st-curve.</td><td>• user.command.lq20LimitJerk</td></tr>
<tr><td>Trajectory Duration</td><td rowspan="2">Amount of time the current trajectory (speed to speed in velocity control or point to point in position control) take to complete. This value is updated with each specification of a new trajectory.</td><td>• trajvel.output.profileTime_tick</td></tr>
<tr><td>Trajectory Duration*</td><td>• trajpos.output.profileTime_tick</td></tr>
</table>

*Sensored position

7.2.4 Startup (sensorless velocity control only)

7.2.4.1 Screen

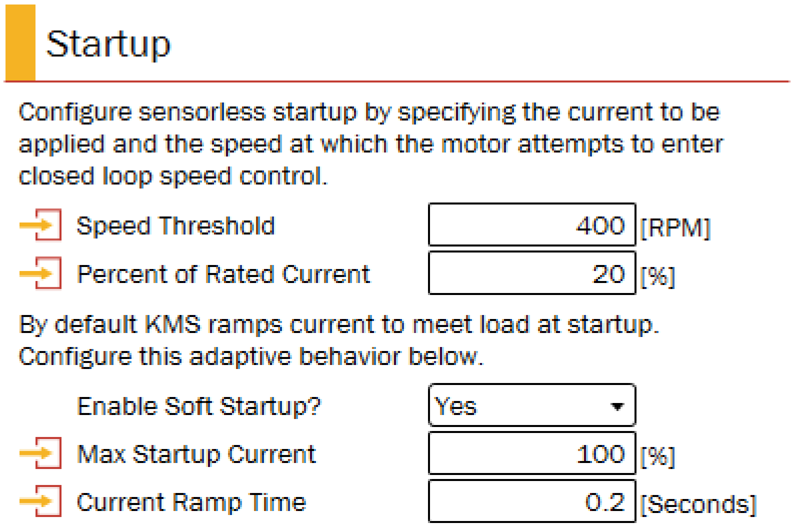


Figure 102. Startup step

7.2.4.2 Description

The user can specify the speed at which the controller attempts to switch between open and closed loop control (Figure 102). This transition requires valid information for the sensorless observer and thus is hardware dependent. The default value is 10% of the rated speed value entered by the user.

The user may also specify the percentage of rated current to be used during startup. Increasing current may allow the system to deal better with high-load startup.

By default, KMS employs a soft startup technique in which the amount of current applied adapts to meet identified load. If the motor is commanded to start but no motion is detected, the current ramps up in accordance with the configuration parameters shown in Figure 102. This is intended to allow startup to occur at the lowest possible current level. This may be switched off in preference of a more standard fixed current level startup.

- Refer to the KMS API Reference Manual for additional information on the sensorless startup algorithm.

7.2.4.3 Parameters

Table 28. Startup parameters

Parameter	Description	Firmware variable
Speed Threshold	Threshold at which the KMS switches from open- to closed-loop control in sensorless velocity operation.	• est.config.FOCLowSpeed

Percent of Rated Current	Percent of the rated current that KMS initially uses during the startup process. This is the minimum value of current to be used during the startup process.	• startup.config.percentMin
Enable Soft Startup?	In some applications, the Soft Startup mechanism may not be desirable. It can be disabled here. When Soft Startup is disabled, the Percent of Rated Current is the current that will be used continuously.	• startup.config.enableSoftStart
Max Startup Current	This is the maximum current that can be used during Soft Startup. Soft Startup cannot increase the current beyond this value. This value can be set higher than 100%.	• startup.config.percentMax
Current Ramp Time	Rate at which current is increased during Soft Startup. This is the amount of time it takes for current to be increased from the specified minimum to specified maximum.	• startup.config.risingTime

7.2.5 Field weakening

7.2.5.1 Screen

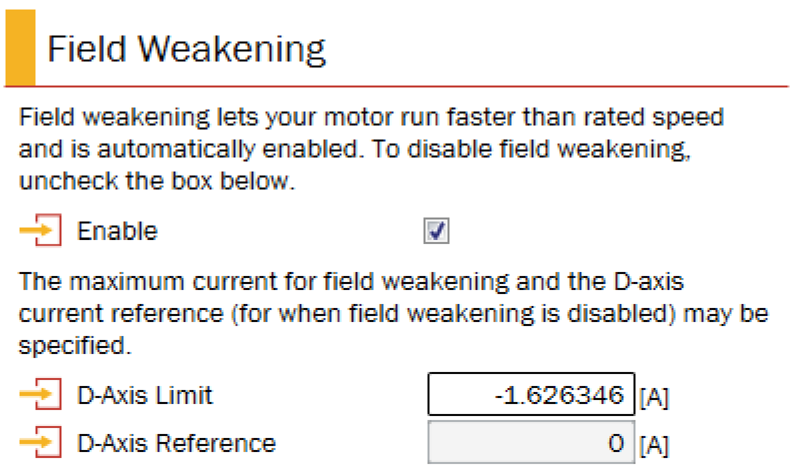


Figure 103. Field weakening step

7.2.5.2 Description

Field weakening enables the motor to run faster than rated speed at a cost of torque. This is achieved by strategically applying current along the direct (D-) axis

This behavior is enabled by default and operated automatically in KMS to enable the user to operate seamlessly near the speed boundary. A limit on the amount of D-axis current that can be applied is also configured by default, based on the motor's electrical characteristics. The user may change this value.

The user may also choose to disable automatic field weakening and either avoid running in this region or define an explicit value that should be commanded for D-axis current.

- Refer to the KMS API Reference Manual for additional information on the implementation of field weakening.

7.2.5.3 Parameters

Table 29. Field weakening parameters

Parameter	Description	Firmware variable
Enable	When checked, the automatic field weakening algorithm provides a reference for the D-axis current. When unchecked, the D-axis reference can be directly specified.	• user.config.FWEnabled
D-Axis Limit	Maximum amount of current that can be applied along the D-axis for field weakening operation.	• fw.config.pi.outMin
D-Axis Reference	Reference provided for D-axis current. When field weakening is enabled, this value is provided for information only. When field weakening is not enabled, the D-axis reference can be directly specified.	• fw.output.IdRef

7.2.6 Position regulator bandwidth (position control only)

7.2.6.1 Screen

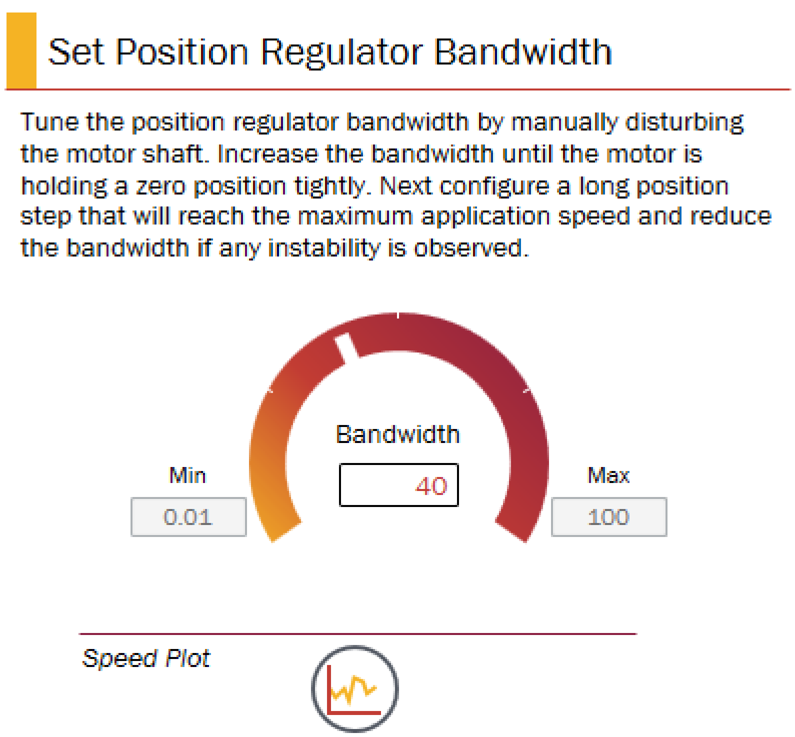


Figure 104. Position regulator bandwidth

7.2.6.2 Description

In position control, KMS uses a single value of bandwidth to control both position and velocity. For this reason, the Speed Control page displays the Position Regulator Bandwidth step - there is no specific speed regulator to tune.

7.2.6.3 Parameters

Table 30. Position regulator bandwidth parameters

Parameter	Description	Firmware variable
Bandwidth	The motion regulator is tuned by adjusting a single parameter, bandwidth, measured in radians per second (rad/s). Bandwidth determines how aggressively the controller compensates for disturbance. Increasing this value increases the stiffness of the position controller. If the motor begins to oscillate or vibrate, decrease this value 10-15%.	• position.config.Iq20Bw_radps

7.3 Position control (sensored position control only)

Use this page to tune the position controller, specify how the motor should move from point A to point B, and set up alignment parameters.

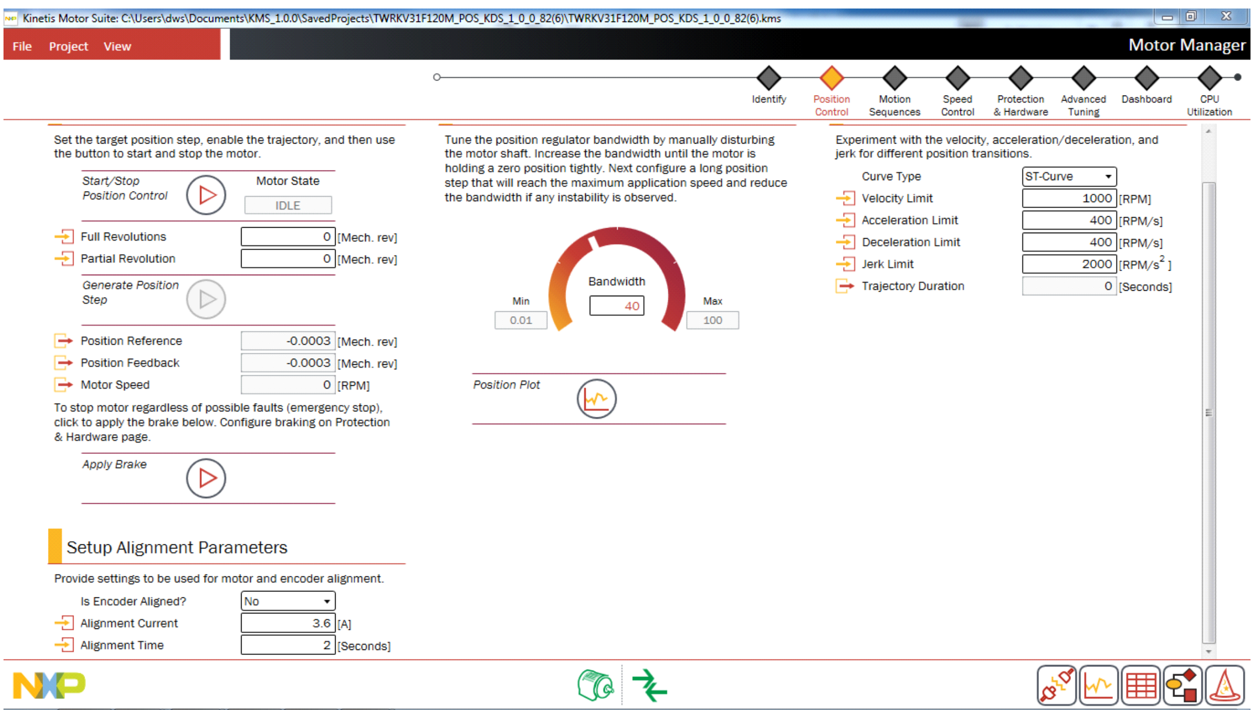


Figure 105. Position control page

7.3.1 Run & Stop Motor

7.3.1.1 Screen

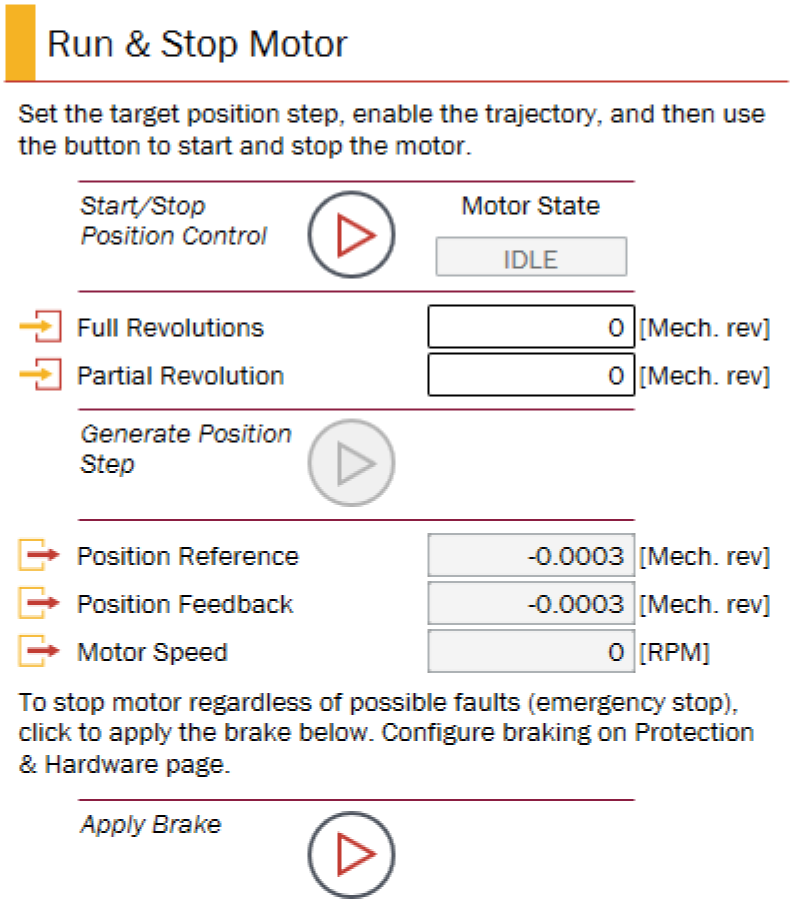


Figure 106. Run & stop motor step (position control)

7.3.1.2 Description

Much like the Run & Stop Motor step on the Speed Control page, the Run & Stop Motor step on the Position Control page focuses on commanding and observing motion. However, due to the more complicated nature of position control, there are additional actions available.

First, the motor must be placed into position control mode; this requires an intermediate alignment state. Next, specify the number of full and partial mechanical revolutions that the motor should turn. Then, click to Generate Position Step (only enabled after motor is in position control mode) to spin the motor. At this point, the feedback information updates to validate that actual motion is as desired.

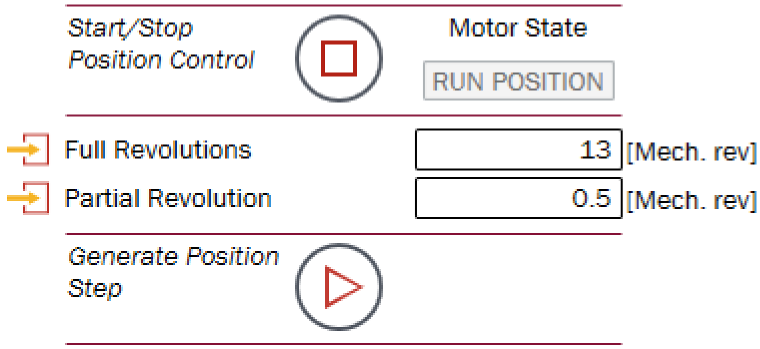


Figure 107. Motor in position control mode, ready to execute movement

Apply the brake to stop the motor at any time.

7.3.1.3 Parameters

Table 31. Run & stop motor parameters

Parameter	Description	Firmware variable
Start/Stop Position Control	Places motor into position control operating mode. Toggles to allow motor to be placed into idle (stopped) operation.	• user.state = 10 for position control • user.state = 0 for idle
Full Revolutions	Number of whole mechanical revolutions that the motor should traverse in its trajectory. This is a relative position step, so it always begins from the previous position.	• user.command.posStepInt_mrev
Partial Revolution	Number of fractional revolutions that the motor should traverse in its trajectory. This is a relative position step, so it always begins from the previous position.	• user.command.posStepFrac_mrev
Generate Position Step	Begin running the position trajectory.	• user.command.runTrajectory = 1 to run • user.command.runTrajectory = 0 to stop
Position Reference	Instantaneous reference position for the motor. This is provided by the trajectory block.	• trajpos.output.refPos_mrev
Position Feedback	Instantaneous feedback position of the motor. This is provided by the encoder module.	• enc.output.rotorAngle_Mrev
Motor Speed	Feedback speed of motor based on quadrature encoder feedback.	• enc.output.rotorSpeed_50Hz
Apply Brake	Places motor into brake mode, which stops the motor in accordance with user-specified braking configuration. Toggles to allow motor to be placed into (stopped) operation.	• user.state = 9 to brake • user.state = 0 for idle

7.3.2 Set Position Regulator Bandwidth

7.3.2.1 Screen

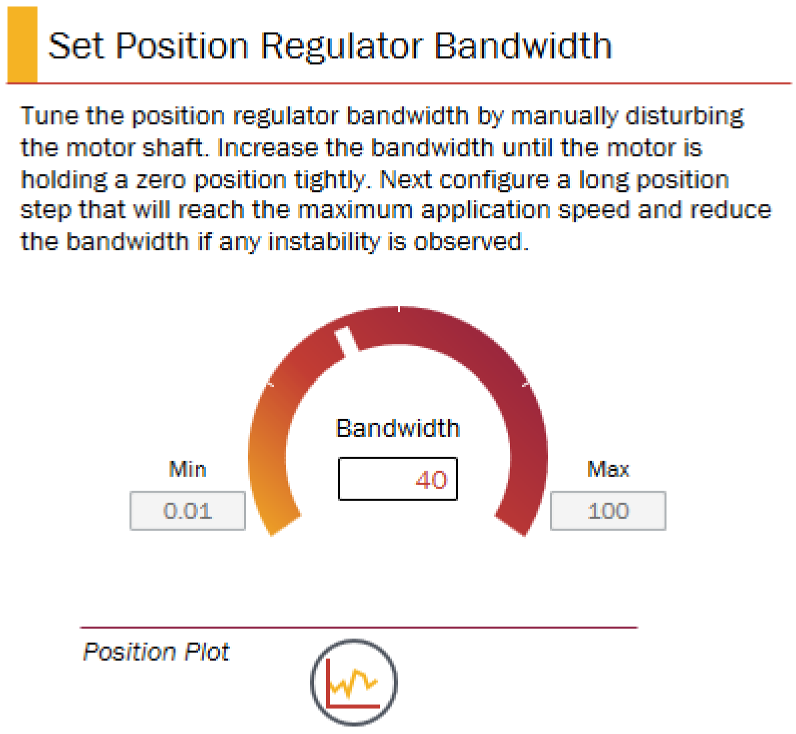


Figure 108. Set position regulator bandwidth step (position control)

7.3.2.2 Description

7.3.2.2.1 Background

KMS offers differentiated performance and ease of use in motion control. KMS features a proprietary control algorithm, which actively estimates system disturbances and compensates for them in real time.

Disturbances may include:

- Uncertainties (e.g. - resonant mode)
- Nonlinear friction
- Changing loads
- Environmental changes

KMS presents better disturbance rejection and trajectory tracking performance than an industry standard PI position controller. It can tolerate a wide range of inertia change, enabling improved accuracy and minimized mechanical system duress.

KMS also features a single tuning parameter, bandwidth, which determines the stiffness of the system and dictates how aggressively the system rejects disturbances. With single coefficient tuning, KMS allows the user to quickly test and tune position control from soft to stiff response. The bandwidth typically works across the entire dynamic range of an application, reducing complexity and system tuning. Once tuned, the controller works over a wide range of dynamics.

7.3.2.2.2 Tuning

KMS provides a default value for position regulator bandwidth but this can be optimized for a specific application using the following tuning process

1. In the Run & Stop Motor step, click to start position control. The Motor State should change from "IDLE" to "RUN POSITION". Click to open the Position Plot and click the Run button to start sampling.
2. Disturb the motor by gently turning the shaft or the attached inertia.
3. Assess shaft stiffness by gently grabbing the shaft and attempting to turn. If stiffer position control is desired, increase the Bandwidth in increments of 10-20 radians per second (rad/s), assessing shaft stiffness at each setting.
4. After the system holds position well in static conditions, verify operation in motion. Input the number of Full or Partial Revolutions desired to assess position control performance, then click to Generate Position Step and run motor. If the motor was unable to turn, it is likely coupled with a large inertia. In the case, increase Alignment Current and Alignment Time incrementally until motor executes the desired mechanical revolutions.
5. Adjust the Velocity, Acceleration, Deceleration, and Jerk limits to adjust the motion path from point A to point B. Lower limits result in a slow execution of the Position Step, whereas larger limits accelerate the execution of the Position Step.

Verify that motor responds well to light load shock and operates stably at both high speed and low speed trajectories. If the motor is unstable, reduce the bandwidth by 10 or 20 rad/s and repeat verification.

7.3.2.3 Parameters

Table 32. Set position regulator bandwidth parameters

Parameter	Description	Firmware variable
Bandwidth	The position controller is tuned by adjusting a single parameter, bandwidth, measured in radians per second (rad/s). Bandwidth determines how aggressively the controller compensates for disturbance. Increasing this value increases the stiffness of the position controller. If the motor begins to oscillate or vibrate, decrease this value 10-15%.	• position.config.lq20Bw_radps

7.3.3 Set up Position Trajectory

7.3.3.1 Screen

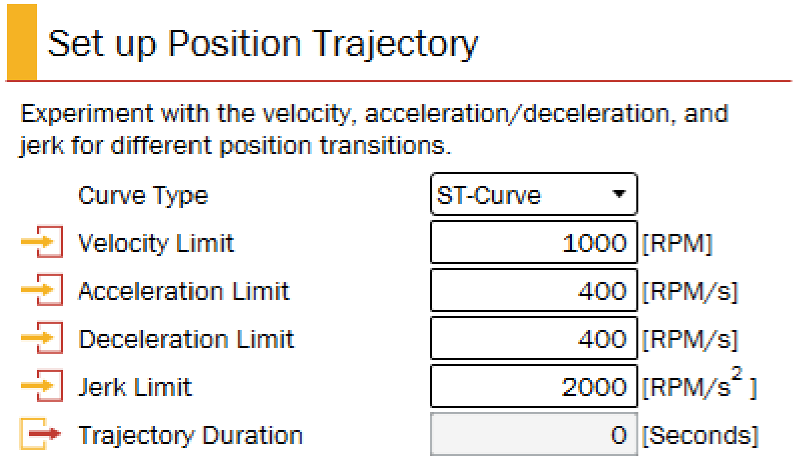


Figure 109. Set up position trajectory step (position control)

7.3.3.2 Description

As in Speed Control, KMS provides the ability to configure kinematic trajectory constraints, but there are additional options for position control.

7.3.3.3 Parameters

Table 33. Set up position trajectory parameters

Parameter	Description	Firmware variable
Curve Type	The user selects from three curve types: • Trapezoidal enables instantaneous transitions; constant acceleration. Note that the acceleration and Jerk limits do not have an impact on this profile. • s-curve linearly increases the acceleration until it reaches the acceleration limit. • st-curve, a proprietary curve type, linearly increases the acceleration and jerk until it reaches the user-specified limits, enabling the smoothest possible motion.	• trajpos.config.curve
Velocity Limit	Restricts the speed that the motor can reach while making a point-to-point transition.	• user.command.limitVel

Acceleration Limit	In contrast to speed control, position control differentiates acceleration vs. deceleration, because in a point-to-point transition (vs. speed to speed), there is intrinsically an initial ramp-up from the starting point (acceleration) and an ultimate ramp-down (deceleration) to reach the endpoint. Thus two different limits are configurable for each trajectory.	• user.command.limitAcc
Deceleration Limit		• user.command.limitDec
Jerk Limit	Restricts the rate of change of acceleration and deceleration in a position movement	• user.command.lq20LimitJerk
Trajectory Duration	Amount of time the current trajectory (point to point in position control) take to complete. This value is updated with each specification of a new trajectory.	• trajpos.output.profileTime_tick

7.3.4 Setup Alignment Parameters

7.3.4.1 Screen

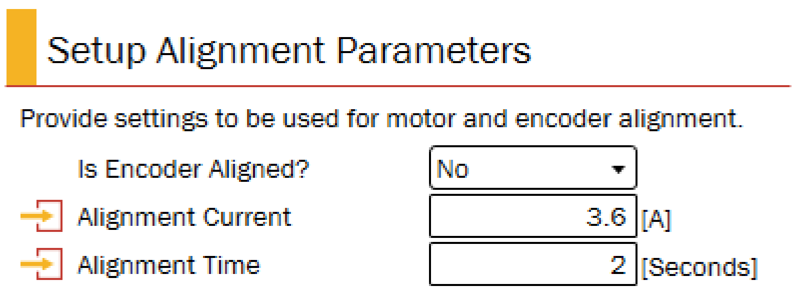


Figure 110. Setup alignment parameters step before alignment

7.3.4.2 Description

The encoder must be aligned to provide accurate rotor angular position information to KMS. This alignment occurs whenever KMS is placed into a control mode requiring encoder feedback. For position control, this is typically achieved by using the Start Position Control button in the Run & Stop Motor step. Determining how this alignment occurs is accomplished in the Setup Alignment Parameters step.

7.3.4.3 Parameters

Table 34. Setup alignment parameters

Parameter	Description	Firmware variable
Is Encoder Aligned?	Reflects current status of system. Toggles from No to Yes after motor has been placed into position control mode. This value can be manually set back to No in order to perform the alignment again.	• user.internal.encoderAligned

Alignment Current	Amount of current applied to the D-axis in order to force the motor to a known position to perform encoder alignment.	• user.config.alignmentCurrent
Alignment Time	Duration allotted for application of current to align rotor with encoder	• user.config.alignmentCounts

7.4 Motion sequences

From this page (Figure 111), users can:

- access Motion Sequence Builder to construct a complex motion sequence
- run the motion sequence loaded to the MCU
- test various trajectories to ensure that the motion sequence meets required operation.

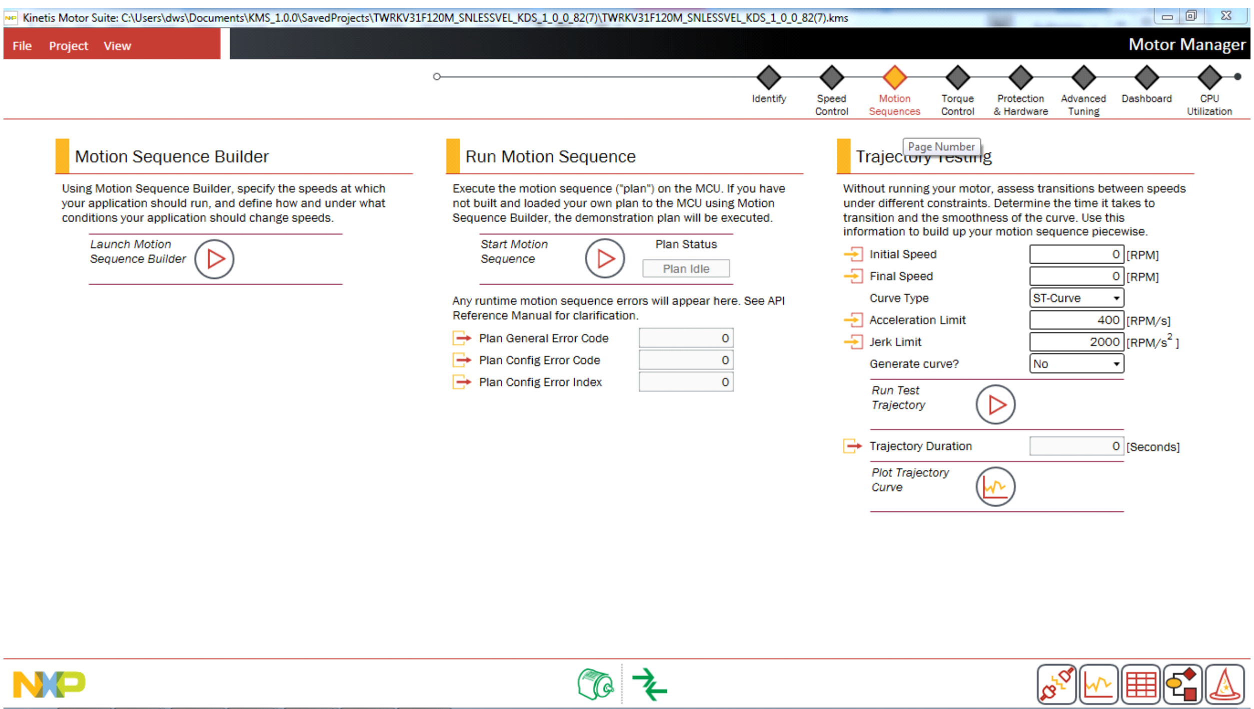


Figure 111. Motion sequences page (velocity control)

7.4.1 Motion Sequence Builder

7.4.1.1 Screen

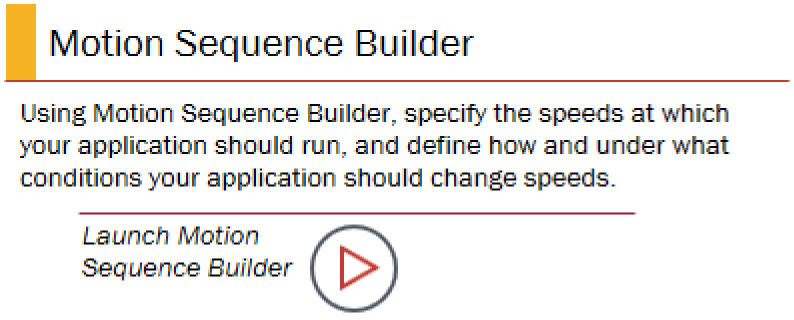


Figure 112. Run KMS Motion Sequence Builder

7.4.1.2 Description

KMS Motion Sequence Builder (Section 9, "Motion Sequence Builder") allows users to build complex motion sequences through an easy-to-use graphical interface. Motion Sequence Builder may be accessed from this page (Figure 112), from the activation bar, or from the View menu.

7.4.2 Run motion sequence

7.4.2.1 Screen

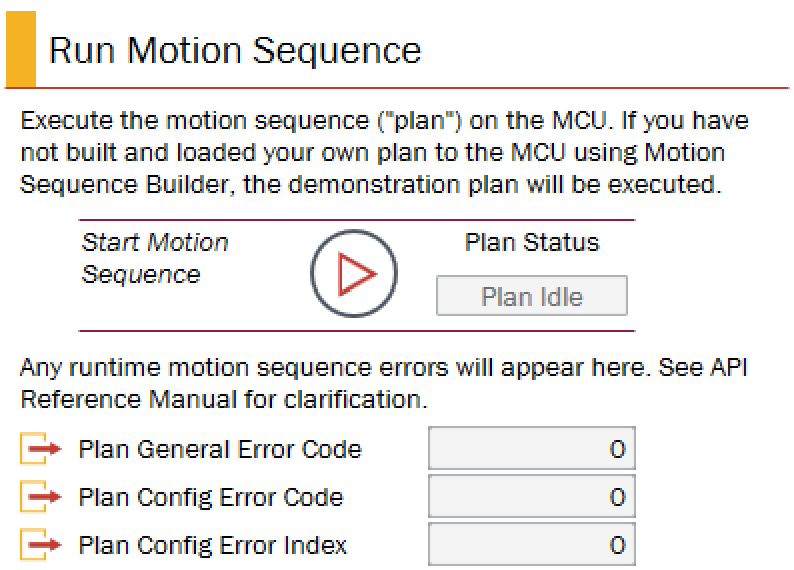


Figure 113. Run motion sequence

7.4.2.2 Description

Each KMS-enabled MCU includes a simple demonstration plan that simulates the motion of a washing machine. Users can also build a motion sequence, or "plan," using Motion Sequence Builder, then download the plan to the MCU. Users can run the plan from this page (Figure 113). Users can specify the Curve Type to be used by the plan from the Speed or Position Control pages (depending on control type)

7.4.2.3 Parameters

Table 35. Run motion sequence parameters

Parameter	Description	Firmware variable
Start Motion Sequence	Places motor into "run plan" operating mode. This executes the motion sequence loaded on the MCU. Toggles to allow motor to be placed into idle (stopped) operation.	• user.state = 8 for run motion sequence • user.state = 0 for idle
Plan General Error Code	Plan configuration function that caused the error. See the KMS API Reference Manual for a list of the error codes that correspond to specific functions.	• VelPlan.ERR_ID
Plan General Error Code*		• PosPlan.ERR_ID
Plan Config Error Code	The specific configuration error that needs to be addressed. See the KMS API Reference Manual for a list of all error codes.	• VelPlan.CfgError.ERR_code
Plan Config Error Code*		• PosPlan.CfgError.ERR_code
Plan Config Error Index	Identifies the index of the function call that caused the configuration error. This count is zero indexed.	• VelPlan.CfgError.ERR_idx
Plan Config Error Index*		• PosPlan.CfgError.ERR_idx

*Sensored position

7.4.3 Trajectory testing

7.4.3.1 Screen

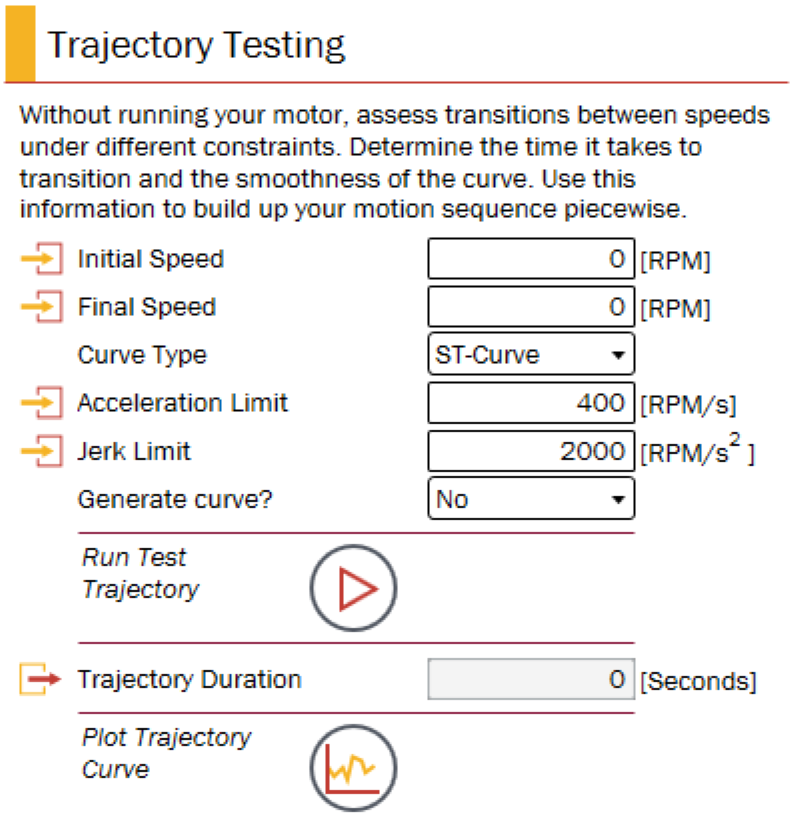


Figure 114. Trajectory testing

7.4.3.2 Description

Users can evaluate the dynamic system performance by changing the Curve Type, Acceleration and Jerk limits (Figure 114). High Acceleration and Jerk limits enable fast transitions, whereas low limits enable slow, smooth motion. Trajectory Testing calculates the trajectory duration, allowing users to find the transition limits that work best for the target application.

By choosing to generate the test curve and plot the trajectory, the expected behavior of the motor can be visualized without actually running the motor.

7.4.3.3 Parameters

Table 36. Trajectory testing parameters

Parameter	Description	Firmware variable
Initial Speed	Speed from which the trajectory should begin. This determines the very first reference in the motion profile.	• trajvelTEST.config.startSpeed

Final Speed	Speed at which the trajectory should end. This determines the very last reference in the motion profile.	• trajvelTEST.config.targetSpeed
Full Revolutions*	Number of whole mechanical revolutions that the motor should traverse in its trajectory. This is a relative position step, so it always begins from the previous position. In this test scenario, the previous position is defined to be zero.	• trajposTEST.config.PosStepInt_mrev
Partial Revolution*	Number of fractional mechanical revolutions that the motor should traverse in its trajectory. This is a relative position step, so it always begins from the previous position. In this test scenario, the previous position is defined to be zero.	• trajposTEST.config.PosStepFrac_mrev
Curve Type	The user selects from three curve types: • Trapezoidal enables instantaneous transitions; constant acceleration. Note that the acceleration and Jerk limits do not have an impact on this profile. • s-curve linearly increases the acceleration until it reaches the acceleration limit. • st-curve, LineStream's proprietary curve type, linearly increases the acceleration and jerk until it reaches the user-specified limits, enabling the smoothest possible motion.	• trajvelTEST.config.curve
Curve Type*		• trajposTEST.config.curve
Acceleration Limit	Acceleration is the rate of change of speed. This value represents the maximum allowed rate of change of speed when using trapezoid, s-curve, and st-curve.	• trajvelTEST.config.limitAcc
Acceleration Limit*		• trajposTEST.config.limitAcc
Deceleration Limit*	Deceleration is the rate of change of speed. This value represents the maximum allowed rate of change when reducing speed when using trapezoid, s-curve, and st-curve.	• trajposTEST.config.limitDec
Jerk Limit	Jerk is the rate of change of motor acceleration. This value represents the maximum allowed rate of change of acceleration when using s-curve and st-curve.	• trajvelTEST.config.lq20LimitJerk
Jerk Limit*		• trajposTEST.config.lq20LimitJerk
Generate Curve?	• Selecting Yes calculates the incremental references for the defined trajectory and enables visualization in the Software Oscilloscope. This allows visual inspection of the calculated references. • Selecting No only calculates the trajectory duration.	• trajvelTEST.config.test
Generate Curve?*		• trajposTEST.config.test
Run Test Trajectory		• N/A
Trajectory Duration	Amount of time the current trajectory (speed to speed in velocity control or point to point in position control) take to complete. This value is updated with each specification of a new trajectory.	• trajvelTEST.output.profileTime_tick
Trajectory Duration*		• trajposTEST.output.profileTime_tick

*Sensored position

7.5 Torque control (velocity control only)

7.5.1 Screen

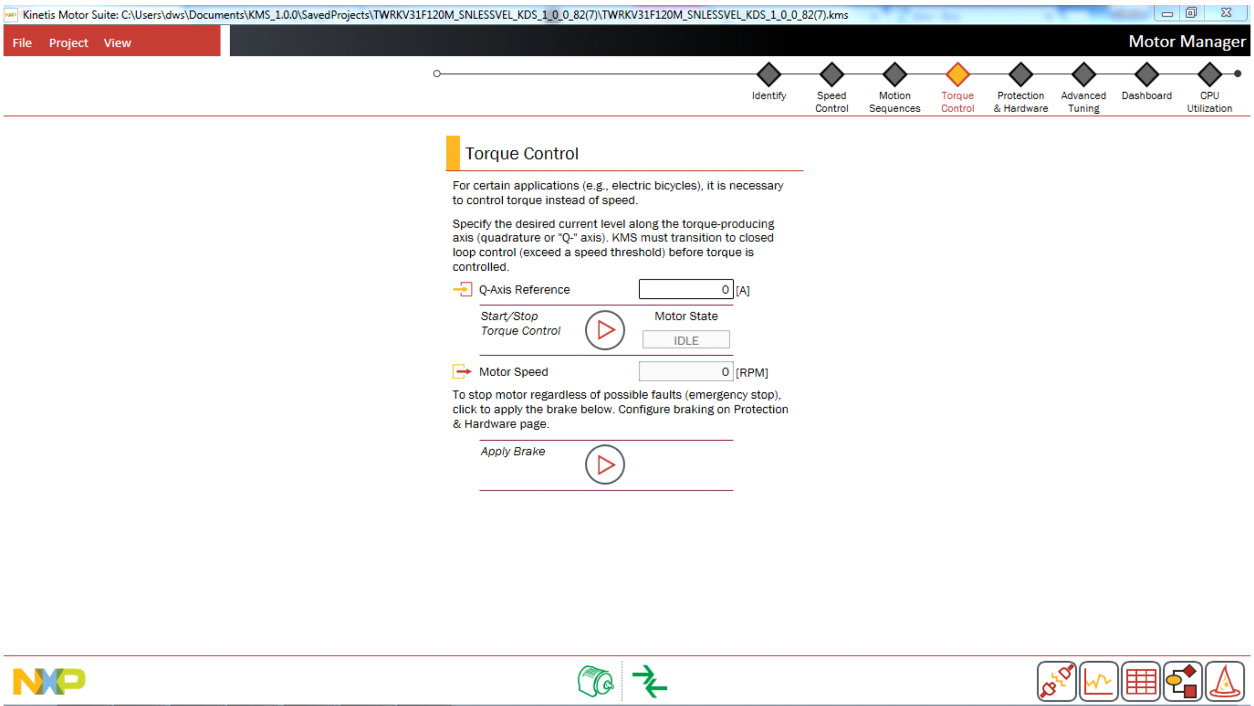


Figure 115. Torque control page

7.5.2 Description

Some applications, such as traction systems and e-bikes, require Torque Control (rather than Speed Control). For these types of applications, users can set the limit of current applied along the torque producing (Q-) axis (Figure 115).

- Refer to the KMS API Reference Manual for additional information regarding how to utilize the torque control mode available in KMS.

7.5.3 Parameters

Table 37. Torque control parameters

Parameter	Description	Firmware variable
Q-Axis Reference	Sets the amount of current used to produce torque. Increasing this value compels the motor to apply more torque.	• user.command.statorRefCurrent.q

Start/Stop Torque Control	Places motor into torque control operating mode. Toggles to allow motor to be placed into idle (stopped) operation.	• user.state = 6 for torque control • user.state = 0 for idle
Motor Speed	Feedback speed of motor based on sensorless angle estimator.	• est.output.rotorSpeed_50Hz
Motor Speed*	Feedback speed of motor based on quadrature encoder feedback.	• enc.output.rotorSpeed_50Hz
Apply Brake	Places motor into brake mode, which stops the motor in accordance with user-specified braking configuration. Toggles to allow motor to be placed into (stopped) operation.	• user.state = 9 to brake • user.state = 0 for idle

*Sensored control (velocity or position)

7.6 Protection & hardware

While KMS is pre-configured to work out of the box with Kinetis development platforms, it is intended to allow the user to adapt to their own power hardware. The primary explanation for how to do this can be found in Application Note 5254: Adapting KMS for Custom Hardware.

The **Protection & Hardware** page (Figure 116) is prominently featured in this process. It enables easy setting of motor current and DC bus voltage feedback scalings, which affect the configuration of the motor drive. Other hardware-dependent parameters that may be defined in the KMS GUI are PWM deadtime, sampling delay, and gain calibration factors.

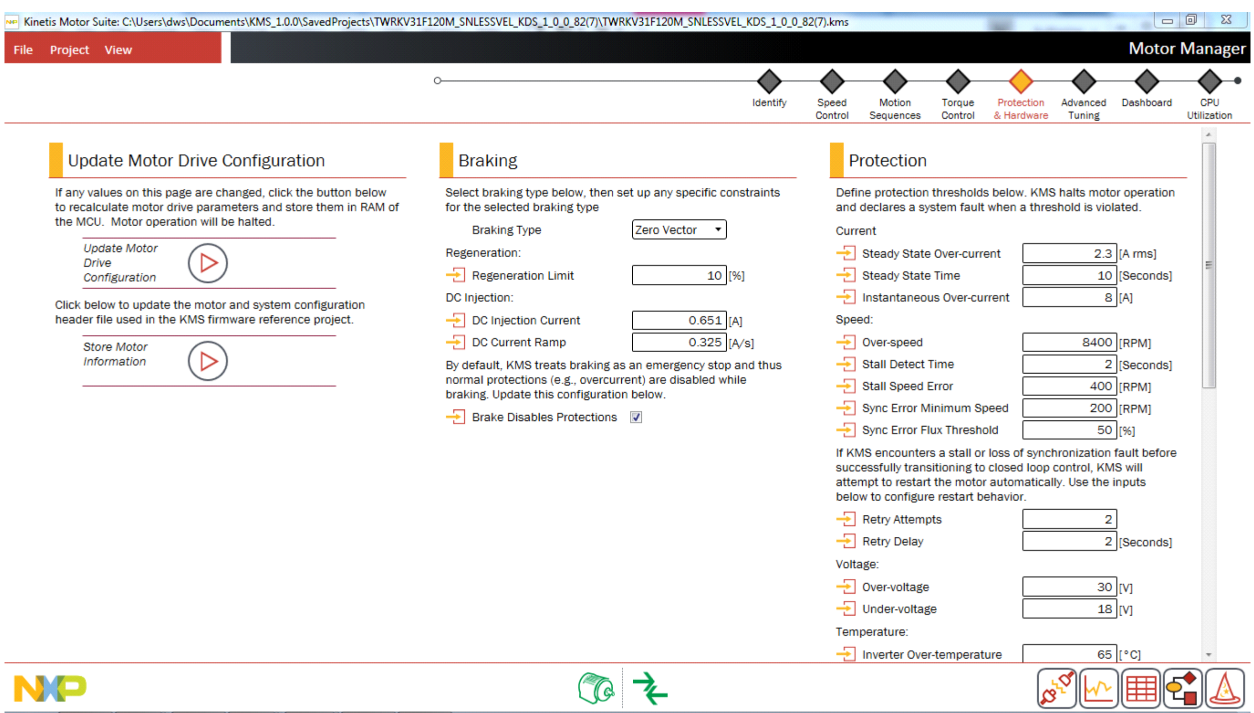


Figure 116. Protection & hardware page

7.6.1 Update motor drive configuration

7.6.1.1 Screen

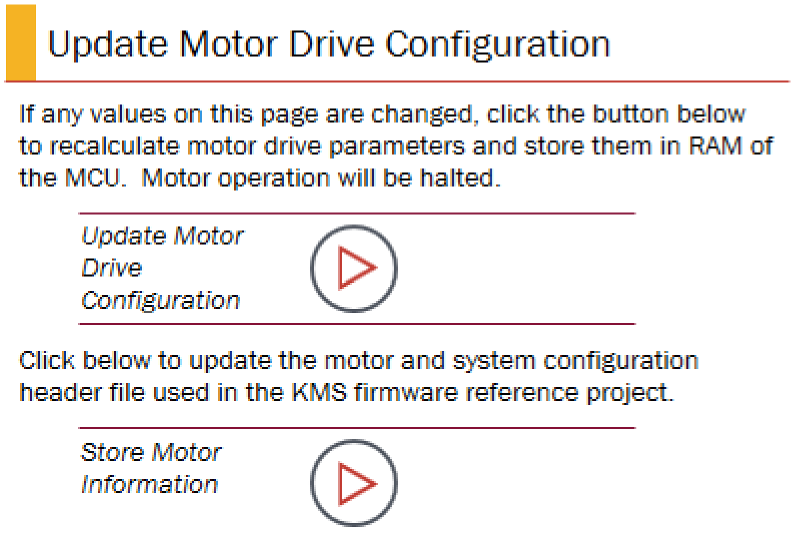


Figure 117. Update motor drive configuration

7.6.1.2 Description

The Motor Drive Configuration must be recalculated (Figure 117) if any value on the Protection and Hardware page is changed. The new values can also be reflected in the KMS reference project by clicking to Store Motor Information.

7.6.2 Braking

7.6.2.1 Screen

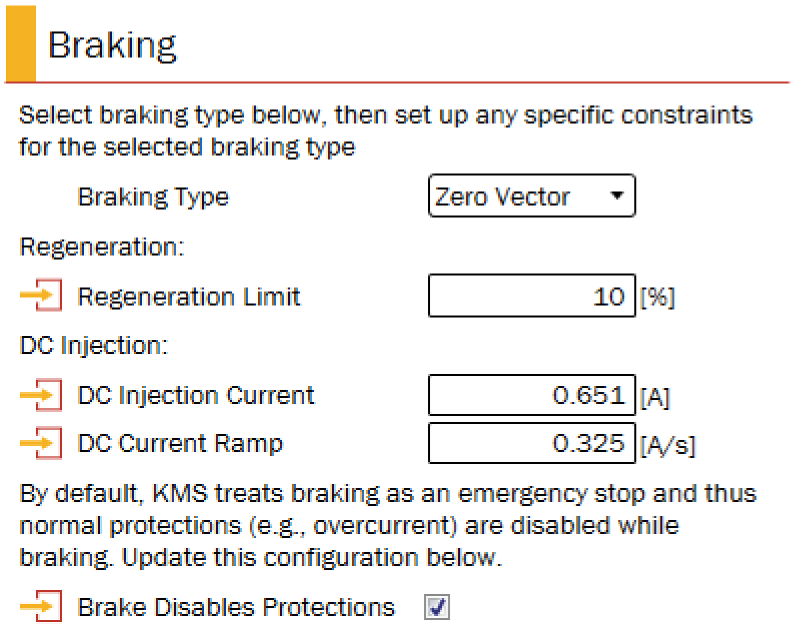


Figure 118. Braking configuration

7.6.2.2 Description

KMS offers four braking options:

Table 38. Braking options

Brake option	Description	Configurable parameters
Zero vector	The low side transistors are turned on and the current circulates freely through the motor and inverter low side devices. Leg shunt sensors monitor the current. This is the default braking type.	None
Regeneration	Synonymous with Ramp to Stop. The motor slows to zero rpm in accordance with specified limits on Acceleration and Jerk. The DC bus voltage should rise. The user can monitor bus voltage through the **Dashboard** page, or with the **Real Time Debugging** tools.	• Acceleration Limit • Jerk Limit • Current Regeneration Limit
Coast	When the Coast brake is applied, the PWM signals are cut off from the inverter. The motor is slowed by the system's load and inertia.	None
DC Injection	DC current can be injected into the motor windings, which provides a braking force to the rotor.	• DC Injection Current (amplitude) • DC Current Ramp Rate

The brake is most often applied and removed on the Speed Control page (Section 7.2.1, "Run & stop motor"). The process for doing this is as follows:

1. Before starting motor operation, select the Braking Type.
2. Configure any related braking settings (see Table 38).
3. Click to Update Motor Drive Configuration. This is required to send the updated braking configuration to the MCU.
4. On **Speed Control** page, click to Start Motor and operate at desired speed.
5. With motor running at desired speed, click to Enable Brake and stop motor.

Figure 119. Apply brake button

6. Disable brake by clicking the Stop Brake button

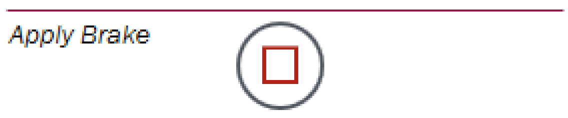


Figure 120. Disable brake

NOTE

Braking operates as an emergency stop. That is, when the brake is applied, software protection thresholds are not enforced and faults are not declared. The objective of braking is first and foremost to stop the motor.

7.6.3 Parameters

Table 39. Braking parameters

Parameter	Description	Firmware variable
Braking Type	Selects the type of braking that should be used if the brake is applied. See the descriptions in Table 38 for the available braking modes.	• brake.config.brakingType
Regeneration Limit	Limits applied to speed or position controller output signal when in regeneration braking mode. This reduces the amount of current used by the motor when braking.	• brake.config.regenIqRefLim
DC Injection Current	Amount of current to use when in DC Injection braking mode. This current is placed along the D-axis and the motor angle is fixed to zero degrees.	• brake.config.dcInjectIdRef

DC Current Ramp	Rate at which current is increased from zero when in DC Injection braking mode.	• brake.config.dcInjectMaxDelta
Brake Disables Protections	• If checked, when the brake is engaged any faults (except hardware faults) are ignored. This is to treat the brake as an emergency stop. • If unchecked, faults operate as normal during braking and may place the motor in the idle state instead of strictly enforcing the desired brake operation.	• user.config.disableFaultsinBrake

7.6.4 Protection

7.6.4.1 Screen

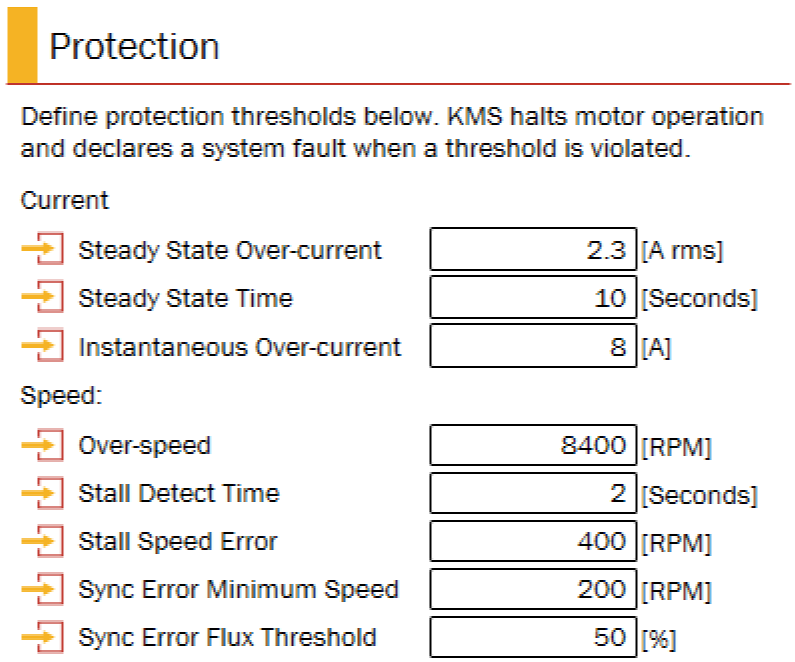


Figure 121. Protection features (1 of 2)

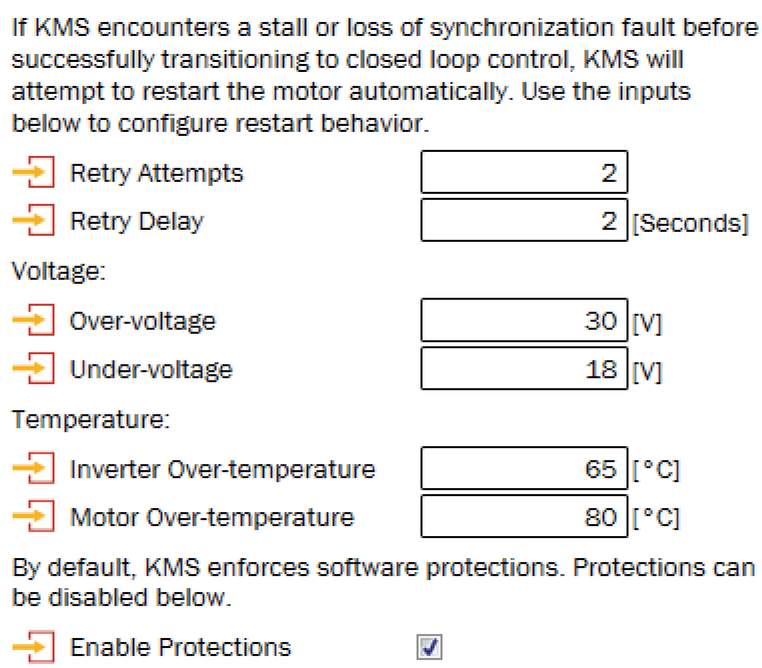


Figure 122. Protection features (2 of 2)

7.6.4.2 Description

KMS allows for system-specific protection thresholds to ensure safe operation. For convenience, these values are predefined but can be manually configured from the *Protection Features* heading (Figure 121 and Figure 122).

Protections are enabled or disabled as a set. If an individual protection should not be applied, the value should be made unreachable.

NOTE

The *Protection Features* are automatically adjusted according to entered information by KMS during *Automatic Parameter Measurement*. To change them manually after this adjustment, the values should be overwritten and the Recalculate Motor Drive Parameters button should be clicked.

7.6.4.3 Parameters

Table 40. Protection parameters

Parameter	Description	Firmware variable
Steady State Over-current	Threshold for detecting operation above the maximum steady state current of the motor. This fault is designed to prevent the motor from overheating due to current demand.	• dsm.faultThresholds.overCurrentRMS
Steady State Over-current Time	The time that the system is permitted to stay in an over-current situation before the system generates an error	• dsm.faultThresholds.overCurrentRMSCounter
Instantaneous Over-current	Threshold for detecting operation above the maximum peak current. This fault is triggered on a cycle-by-cycle basis and is designed to prevent a very large current from flowing through the power devices.	• dsm.faultThresholds.overCurrentPeak
Over-speed	Threshold for the maximum speed of the motor. Operation above this value triggers immediate fault declaration.	• dsm.faultThresholds.overSpeed
Stall Detect Time	The amount of time the motor is allowed to deviate from desired speed by the Stall Speed Error Threshold before a fault is declared	• dsm.faultThresholds.stallCounter
Stall Speed Error Threshold	Maximum allowable speed error (difference between reference and feedback speeds). This fault is designed to detect conditions where the motor cannot achieve the goal speed.	• dsm.faultThresholds.stallSpeedError
Sync Error Min Speed	Speed above which sync error detection is allowed.	• dsm.faultThresholds.minSyncSpeed
Sync Error Flux Threshold	Minimum motor flux that indicates the motor is spinning. If the estimated flux drops below this value (defined as a percentage of the flux measured during motor measurement), a fault is declared. This fault is designed to detect when the motor is not rotating but the feedback speed indicates that it is.	• dsm.faultThresholds.pmFluxSyncThreshWb

Retry Attempts	Number of attempts KMS will make automatically to start motor if synchronization or stall error is declared before successful transition to closed loop control	• user.config.startupRetryAttempts
Retry Delay	Time between automatic attempts to restart after failed startup	• user.config.startupRetryDelay
Over-voltage	Maximum allowed DC bus voltage. This fault is designed to protect the power electronics from voltage regeneration.	• dsm.faultThresholds.overVoltage
Under-voltage	Minimum allowed DC bus voltage. This fault is designed to prevent the motor from running when the DC bus is insufficient to run the motor.	• dsm.faultThresholds.underVoltage
Inverter Over-temperature	Maximum allowed inverter temperature. This fault is designed to prevent the inverter from experiencing an over temperature condition. This fault is implemented on HVP-MC3PH only.	• dsm.faultThresholds.overTempInverter
Motor Over-temperature	Maximum allowed motor temperature. This fault is designed to prevent the motor from experiencing an over temperature condition. This fault requires custom hardware.	• dsm.faultThresholds.overTempMotor
Enable Protections	• If checked, faults operate as described. • If unchecked, KMS does not stop the motor control upon detection of a fault. Faults are still detected but do not trigger any action from KMS. – This does not apply to hardware faults.	• dsm.faultThresholds.enableFaults

7.6.5 Hardware configurations

7.6.5.1 Screen

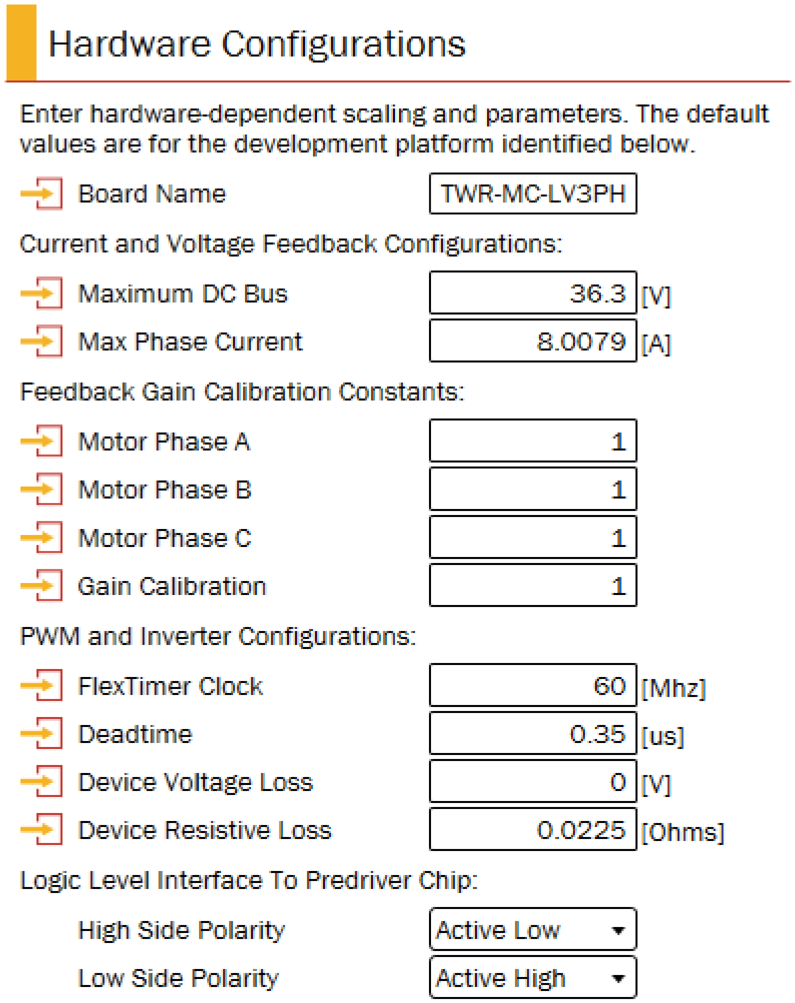


Figure 123. Hardware configuration settings

7.6.5.2 Description

KMS adjusts the *Hardware Configuration* (Figure 123) based on the automatically measured parameters. As users begin to build their own power hardware, it may be necessary to further adjust the *Hardware Configurations*.

7.6.5.3 Parameters

Table 41. Hardware configuration parameters

Parameter	Description	Firmware variable
Board Name	Unique name for the motor drive board. For information purposes only.	• N/A
Maximum DC Bus	Value on the DC Bus when the ADC reads the maximum value. This value establishes the base voltage for the system.	• FULL_SCALE_VOLTAGE
Max Phase Current	Value on the Phase Current Sense when the ADC reads the maximum value. This value establishes the base current for the system.	• FULL_SCALE_CURRENT
Current Feedback Gain Calibration (Motor Phases A, B, C)	Adjustment to the overall gain of the ADC feedback signal. Used for fine-tuning the ADC readings for specific measurements.	• feedback.calib.gains.kIa • feedback.calib.gains.kIb • feedback.calib.gains.kIc
Gain Calibration	Adjustment to the overall gain of the ADC feedback signal. Used for fine-tuning the ADC readings for specific measurements.	• feedback.calib.gains.kVdc
FlexTimer Clock	Frequency of the clock provided to the FlexTimer Module	• flashSysParams.sysFreqHz
Deadtime	Power device deadtime (found on power device datasheet).	• flashSysParams.dTperiod
Device Voltage Loss	Voltage drop across the power device (found on the power device datasheet).	• est.config.V_igbt
Device Resistive Loss	Resistance through the power device (found on the power device datasheet).	• est.config.R_igbt
High Side Polarity	Logic level interface for the high side switch of the pre-driver chip or power module. Note that changing this value from the GUI enables this setting to be passed to the KMS system.h file, but the code must be manually edited for this configuration to take effect. This is to minimize the risk to KMS development platforms: setting this incorrectly can destroy the motor drive power stage.	• flashSysParams.enableHighSideActiveLow
Low Side Polarity	Logic level interface for the low side switch of the pre-driver chip or power module. Note that changing this value from the GUI enables this setting to be passed to the KMS system.h file, but the code must be manually edited for this configuration to take effect. This is to minimize the risk to KMS development platforms: setting this incorrectly can destroy the motor drive power stage.	• flashSysParams.enableLowSideActiveLow

7.7 Advanced tuning

The **Advanced Tuning** page (Figure 124) allows for optimization of the motor drive settings. After changing any of the parameters in this step, be sure to click the button to Update Motor Drive Configuration. The motor is stopped and updated drive parameters are downloaded to RAM.

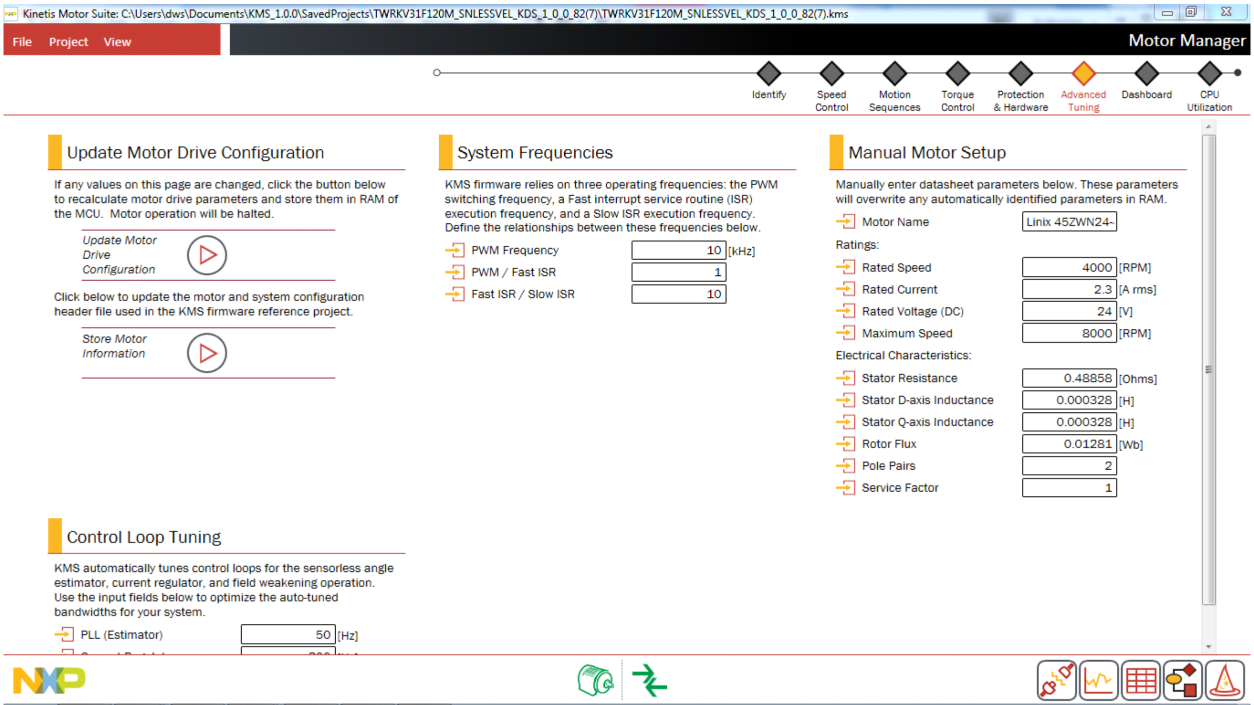


Figure 124. Advanced tuning page

7.7.1 Update motor drive configuration

7.7.1.1 Screen

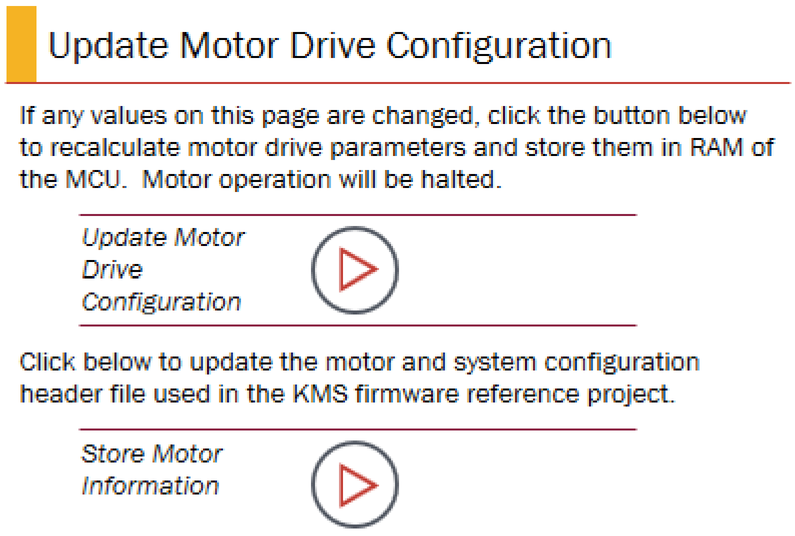


Figure 125. Update motor drive configuration

7.7.1.2 Description

Motor Drive Parameters must be recalculated (Figure 125) if any value on the Advanced Tuning page is changed. From here, the new parameters can also be saved in the appropriate KMS reference project header file.

7.7.2 System frequencies

7.7.2.1 Screen

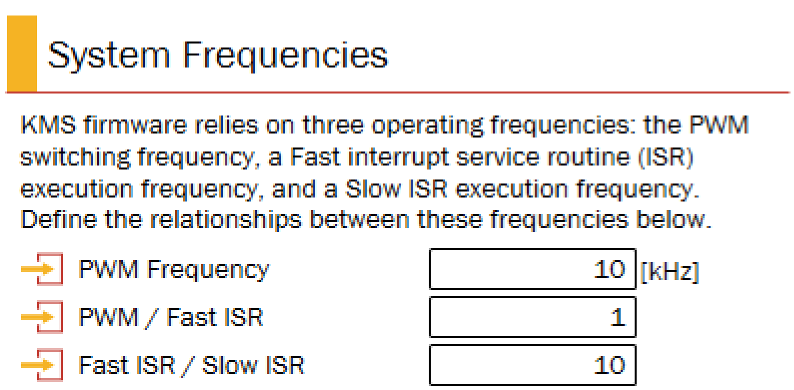


Figure 126. System frequencies

7.7.2.2 Description

The user may manually configure the pulse width modulation frequency (PWM Frequency) and related execution frequencies (Figure 126).

- Refer to the KMS API Reference Manual for additional information on how the ISRs are configured and what code is executed in which ISR.

7.7.2.3 Parameters

Table 42. System frequencies parameters

Parameter	Description	Firmware variable
PWM Frequency	Rate at which Pulse Width Modulation occurs	• flashSysParams.pwmFreqHz
PWM / Fast ISR	Ratio between PWM switching frequency and the rate at which the core motor control code is executed; the Fast ISR is also known as the PWM ISR	• flashSysParams.FocPwmDecimation
Fast ISR / Slow ISR	Ratio between the rate at which core motor control code and slower application code is executed	• flashSysParams.fastTicksPerSlowTick

7.7.3 Manual motor setup

7.7.3.1 Screen

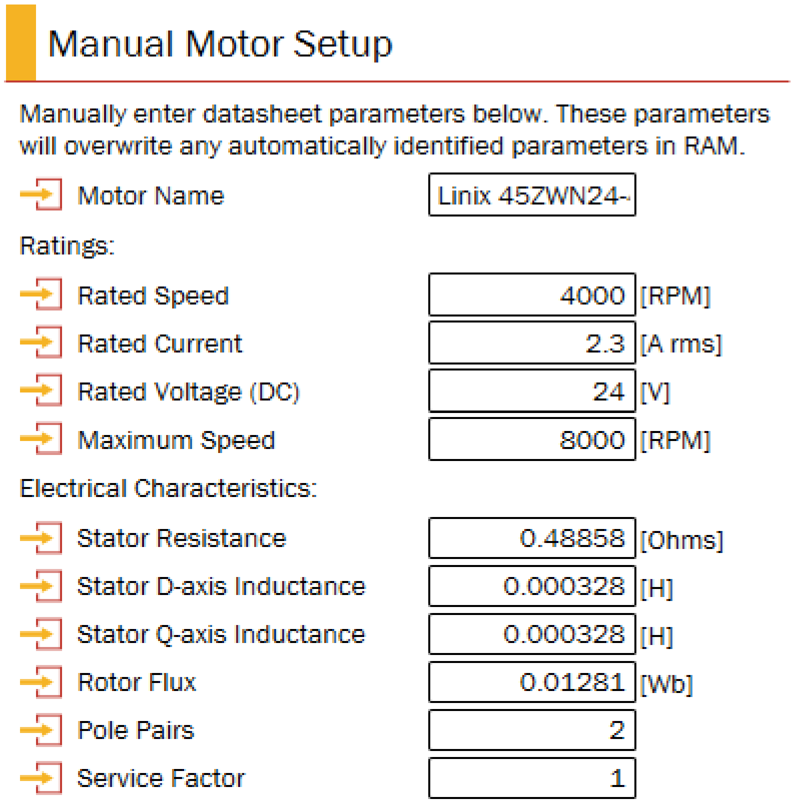


Figure 127. Manual motor setup

7.7.3.2 Description

If a motor is connected to an end application, it may not be possible for KMS to automatically identify the motor parameters. In this case, the user may manually enter motor parameters (Figure 127). These parameters overwrite any automatically identified parameters in RAM. Be sure to Update Motor Drive Configuration if any of these values change.

7.7.3.3 Parameters

Table 43. Manual motor setup parameters

Parameter	Description	Firmware variable
Motor Name	Enter a name for the motor for convenience. This is not required.	• N/A
Rated Speed	The maximum speed at which the rated torque can be delivered.	• flashSysParams.motorParams.ratedSpeed

Rated Current	The RMS (not peak) current for which the motor is rated. This value is used to establish the maximum output of the motion controller.	• flashSysParams.motorParams.ratedCurrent
Rated Voltage (DC)	Rated DC bus voltage of the motor. If specified on datasheet as VAC, multiply by square root 2 to arrive at VDC value.	• flashSysParams.motorParams.ratedVoltage
Maximum Speed	Maximum speed the application can run.	• flashSysParams.motorParams.maxSpeed
Stator Resistance	The per phase winding resistance of your motor.	• flashSysParams.motorParams.statorRes
Stator D-Axis Inductance	The per phase winding d-axis inductance of your motor at rated current.	• flashSysParams.motorParams.statorDInd
Stator Q-Axis Inductance	The per phase winding q-axis inductance of your motor at rated current. By default, this is assumed equivalent to d-axis inductance (non-salient motor condition) but may be manually edited if known.	• flashSysParams.motorParams.statorQInd
Rotor Flux	The rotor flux (or permanent magnet flux linkage) of your motor.	• flashSysParams.motorParams.pmFlux
Pole Pairs	The number of pairs of magnetic poles in one mechanical revolution.	• flashSysParams.motorParams.polePairs
Service Factor	Allows the motor to run at a higher current than maximum. This increases the maximum output of the motion controller.	• N/A

7.7.4 Control loop tuning

7.7.4.1 Screen

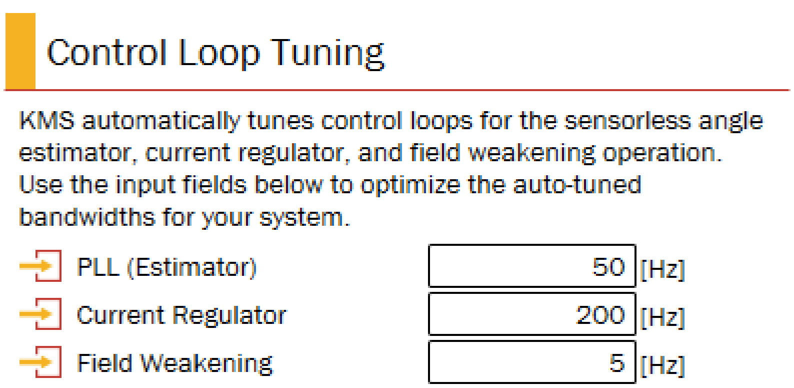


Figure 128. Control loop tuning

7.7.4.2 Description

The sensorless observer and current regulators are automatically tuned when KMS identifies the motor and system parameters. However, the user may manually adjust both the PLL and Current Regulator Bandwidth (Figure 128).

Field weakening allows the motor to run faster than the rated speed at the cost of torque. The system may require a different bandwidth when it moves into field weakening. Be sure to Update the Motor Drive Configuration if any of these values are changed.

- Refer to the KMS API Reference Manual for additional detail on the implementation of these advanced features.

7.7.4.3 Parameters

Table 44. Control loop tuning parameters

Parameter	Description	Firmware variable
PLL (Estimator)*	Gain for the sensorless angle estimator. Information from motor measurement is used to set a default value to enable sensorless operation. This value may need to be adjusted for challenging motors or reduced if the application has a large inertia.	• N/A
Current Regulator	Gain for the current controllers. After motor parameter identification, this value is tuned using the pole-zero cancellation technique. For some applications it may need to be increased to enable higher dynamic performance.	• N/A
Field Weakening**	Gain for the field weakening controller. This value is designed to provide smooth performance in the field weakening region.	• N/A

*Sensorless velocity

**Velocity (sensored or sensorless)

7.8 Dashboard

Think of the Dashboard as your motor performance observation center. After commanding the motor to spin, the user can monitor all key values from a single page (Figure 129).

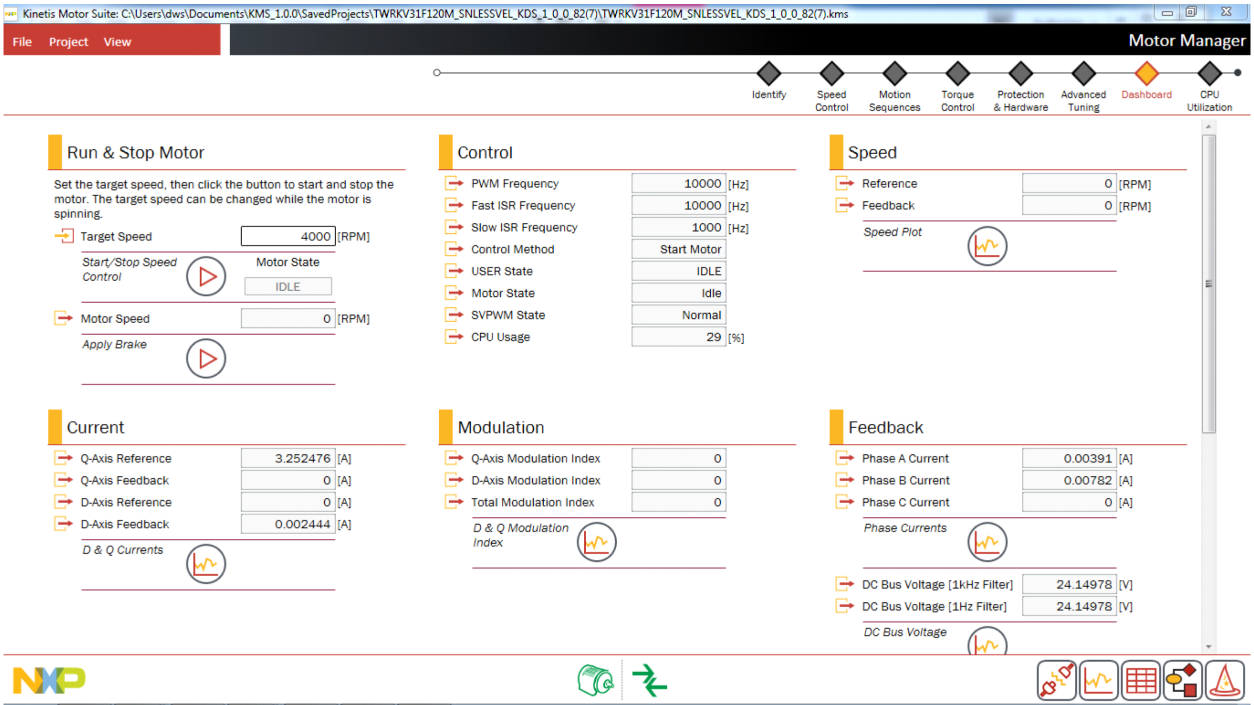


Figure 129. Dashboard page

7.8.1 Run & Stop Motor

7.8.1.1 Screen

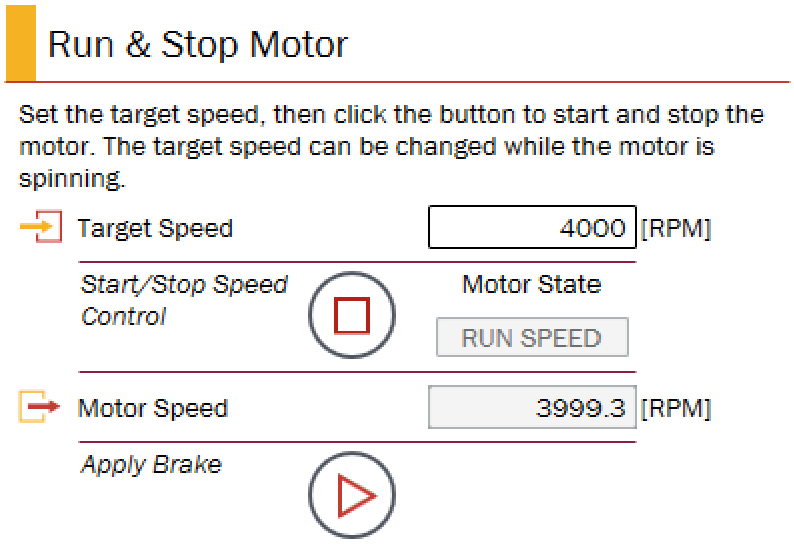


Figure 130. Run & Stop Motor step (sensorless velocity)

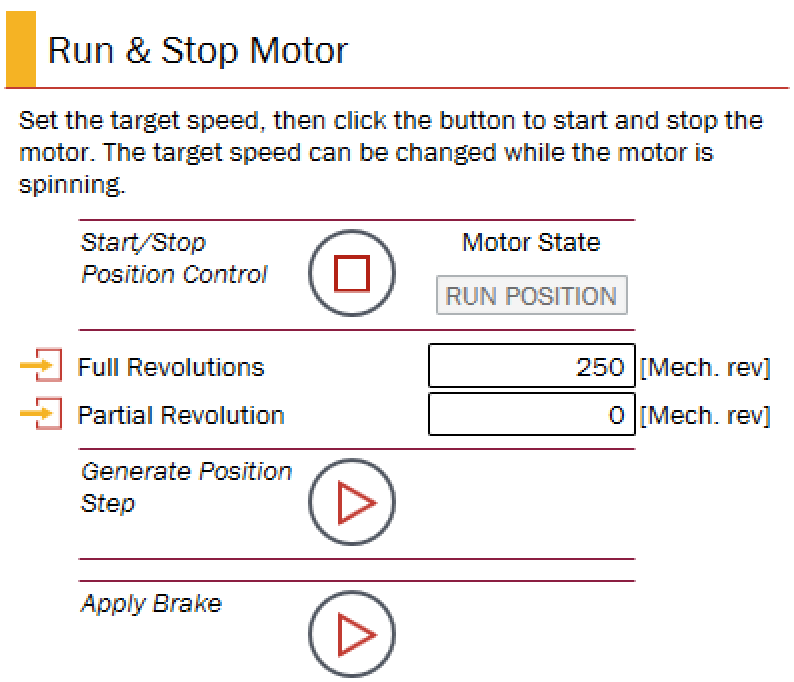


Figure 131. Run & Stop Motor step (sensored position)

7.8.1.2 Description

The Run & Stop Motor step enables the user to toggle motor operation on and off while on the Dashboard page. This is provided as a convenience to avoid having to jump from page to page while trying to view updates to values on the Dashboard.

7.8.1.3 Parameters

Table 45. Run & stop motor parameters

Parameter	Description	Firmware variable
Target Speed	Desired motor speed, measured in revolutions per minute (RPM).	• user.command.targetSpeed
Start/Stop Speed Control	Places motor into speed control operating mode. Toggles to allow motor to be placed into idle (stopped) operation.	• user.state = 7 for speed control • user.state = 0 to return to idle
Start/Stop Position Control	Places motor into position control operating mode. Toggles to allow motor to be placed into idle (stopped) operation.	• user.state = 10 for position control • user.state = 0 to return to idle
Motor Speed	Feedback speed of motor based on sensorless angle estimator.	• est.output.rotorSpeed_50Hz
Motor Speed*	Speed of motor based on quadrature encoder feedback.	• enc.output.rotorSpeed_50Hz
Apply Brake	Places motor into brake mode, which stops the motor in accordance with user-specified braking configuration. Toggles to allow motor to be placed into (stopped) operation.	• user.state = 7 for brake • user.state = 0 to return to idle
Full Revolutions**	Number of whole mechanical revolutions that the motor should traverse in its trajectory. This is a relative position step, so it always begins from the previous position.	• user.command.posStepInt_mrev
Partial Revolution**	Number of fractional revolutions that the motor should traverse in its trajectory. This is a relative position step, so it always begins from the previous position.	• user.command.posStepFrac_mrev
Generate Position Step**	Begin running the position trajectory.	• user.command.runTrajectory = 1 to run • user.command.runTrajectory = 0 to stop

*Sensored control (velocity or position)
**Sensored position

7.8.2 Control

7.8.2.1 Screen

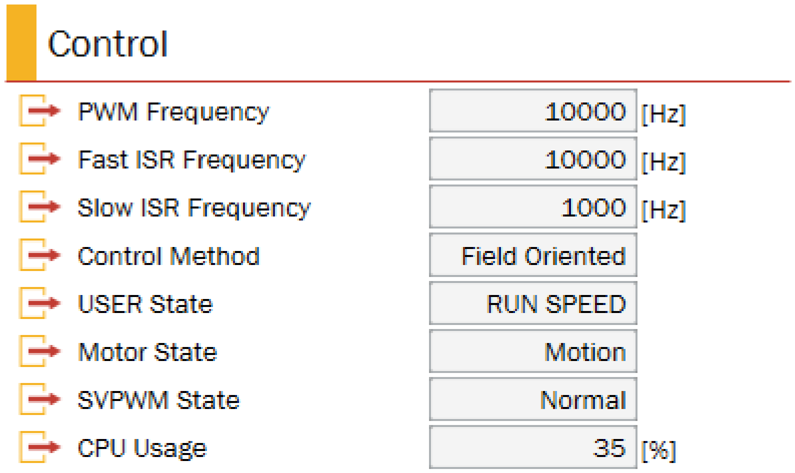


Figure 132. Control step (sensorless velocity)

7.8.2.2 Description

The Control step makes explicit the mode of operation of KMS at any point in time. Parameters include:

- absolute values for system frequencies (vs. the relationships among them described on the Advanced Tuning page)
- control method (used most critically to determine when the motor switches from open loop to closed loop speed control in sensorless operation)
- user state (commanded operating mode: speed, position, self-commissioning, etc.)

7.8.2.3 Parameters

Table 46. Control parameters

Parameter	Description	Firmware variable
PWM Frequency	The frequency of the Pulse Width Modulator (PWM), the hardware switching frequency.	• flashSysParams.pwmFreqHz
Fast ISR Frequency	The frequency at which the main motor control code is executed.	• flashSysParams.focFreqHz
Slow ISR Frequency	The frequency at which the slower mechanical control loops are executed (motion control and application code).	• flashSysParams.slowFreqHz

Control Method*	Control method currently being used. • Startup indicates that the sensorless estimator is not providing the angle source (speed loop is not closed) • FOC indicates that the sensorless estimator is providing the angle source.	• startup.output.controlType
USER State	Indicates the user control mode (speed control, position control, brake mode, etc.). Describes which references are being provided to core motor control.	• user.state
Motor State	Indicates the field oriented control (FOC) operating mode (e.g. current control, voltage control, motion control). This shows which FOC blocks are active. Refer to the KMS API Reference Manual block diagrams for additional detail.	• dsm.state
SVPWM State	Indicates if the SVPWM block is operating in the normal region or the overmodulation region.	• svpwm.output.state
CPU Usage	Percent of CPU consumed by the FOC algorithm. Reflects the sum of the Fast ISR usage and the Slow ISR usage.	• N/A

*Sensorless velocity

7.8.3 Speed

7.8.3.1 Screen

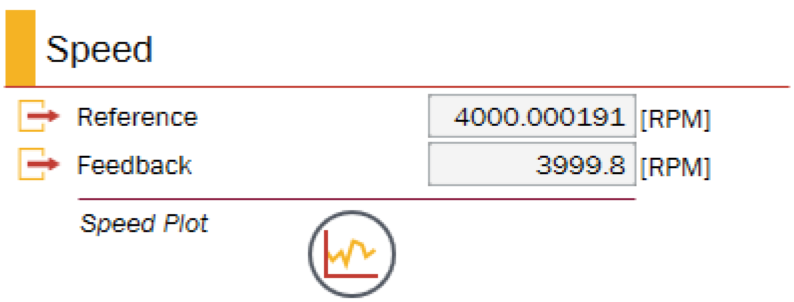


Figure 133. Speed step (sensorless velocity)

7.8.3.2 Description

The Speed step provide targeted information on the motion of the system, actual vs. commanded.

7.8.3.3 Parameters

Table 47. Speed parameters

Parameter	Description	Firmware variable
Reference	Instantaneous speed that the motor should be spinning. This is provided by the trajectory block and is not equal to the goal speed until the trajectory is completed.	• trajvel.output.refSpeed
Feedback	Feedback speed of motor based on sensorless angle estimator.	• est.output.rotorSpeed_50Hz
Feedback*	Feedback speed of motor based on quadrature encoder feedback.	• enc.output.rotorSpeed_50Hz

*Sensored velocity

7.8.4 Position & Speed (sensored position only)

7.8.4.1 Screen

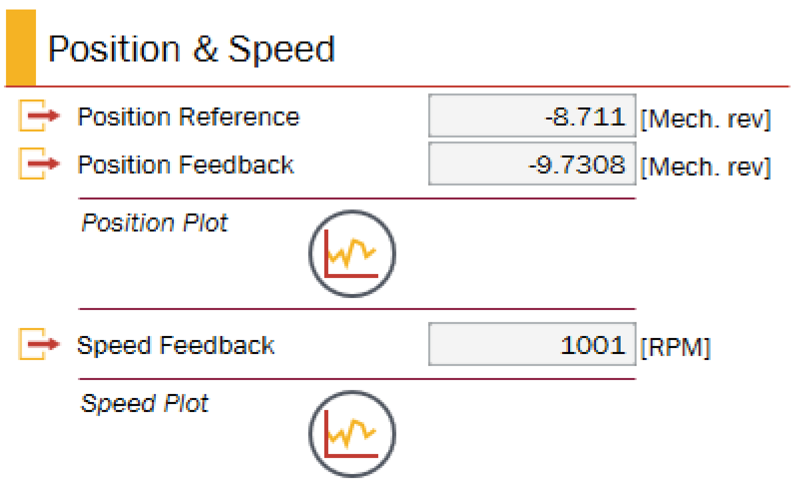


Figure 134. Postion & speed step

7.8.4.2 Description

The Position & Speed step provides targeted information on the motion of the system, actual vs. commanded.

7.8.4.3 Parameters

Table 48. Position & speed parameters

Parameter	Description	Firmware variable
Position Reference	Instantaneous reference position for the motor. This is provided by the trajectory block.	• trajpos.output.refPos_mrev
Position Feedback	Instantaneous feedback position of the motor. This is provided by the encoder module.	• enc.output.rotorAngle_Mrev
Speed Feedback	Feedback speed of motor based on quadrature encoder feedback.	• enc.output.rotorSpeed_50Hz

7.8.5 Current

7.8.5.1 Screen

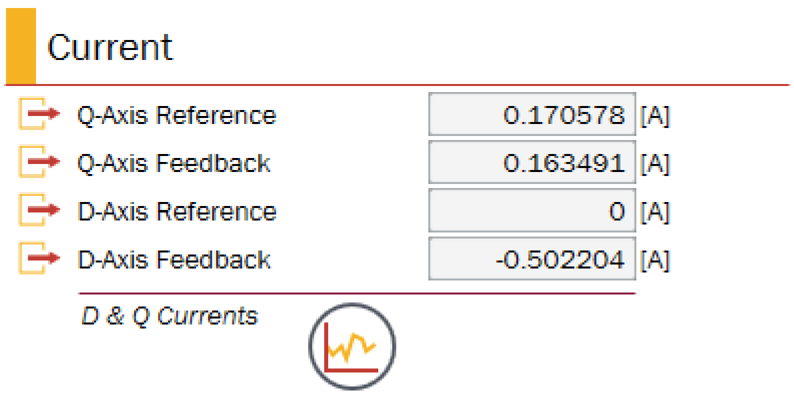


Figure 135. Current step

7.8.5.2 Description

The Current step displays information regarding the commanded and actual currents applied along the direct and quadrature axes. Among other things, this information may be used to validate operation in the normal and field weakening modes: the D-axis reference should be zero in normal operation to maximize torque, while it should be non-zero to push speed higher in field weakening.

7.8.5.3 Parameters

Table 49. Current parameters

Parameter	Description	Firmware variable
Q-Axis Reference	Amount of current requested by the speed controller. This is the current that produces torque in the motor.	• speed.output.IqRef
Q-Axis Reference**	Amount of current requested by the position controller. This is the current that produces torque in the motor.	• position.output.IqRef

Q-Axis Feedback	Feedback current flowing through the motor. This is the amount of torque-producing current in the motor.	• park.output.vector.q
D-Axis Reference*	Amount of current requested by the field weakening controller. This is the current that allows the motor to run faster than rated speed.	• fw.output.IdRef
D-Axis Feedback	Feedback current flowing through the motor. This is the amount of flux producing current in the motor.	• park.output.vector.d

*Velocity (sensored or sensorless)
**Sensored position

7.8.6 Modulation

7.8.6.1 Screen

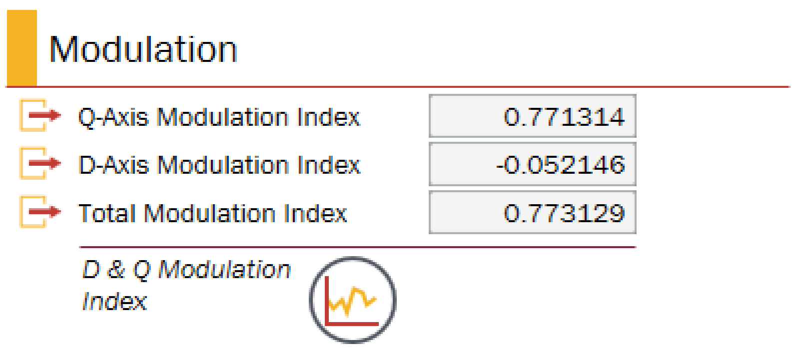


Figure 136. Modulation step (sensorless velocity)

7.8.6.2 Description

The values in the Modulation step represent the percent of DC Bus being requested by the current controller. Modulation value is provided along both the direct and the quadrature axes, and as a total magnitude.

7.8.6.3 Parameters

Table 50. Modulation parameters

Parameter	Description	Firmware variable
Q-Axis Modulation Index	Percentage of DC bus applied to the motor in order to adjust the torque-producing current.	• current.output.statorRefVoltageDQ.q
D-Axis Modulation Index	Percentage of DC bus applied to the motor in order to adjust the flux-producing current.	• current.output.statorRefVoltageDQ.d
Total Modulation Index	Total percentage of DC bus applied to the motor.	• ipark.output.magnitude

7.8.7 Feedback

7.8.7.1 Screen

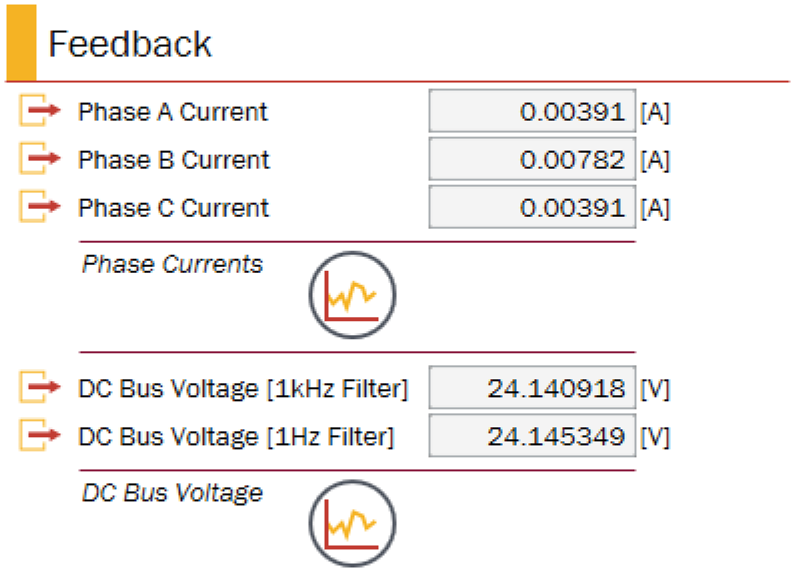


Figure 137. Feedback step

7.8.7.2 Description

The Feedback step provides insight into the key signals being sensed by KMS to operate field oriented control - in particular, the motor phase currents.

7.8.7.3 Parameters

Table 51. Feedback parameters

Parameter	Description	Firmware variable
Phase A Current	Instantaneous feedback current flowing through motor phase A.	• feedback.output.Isa
Phase B Current	Instantaneous feedback current flowing through motor phase B.	• feedback.output.Isb
Phase C Current	Instantaneous feedback current flowing through motor phase C.	• feedback.output.Isc
DC Bus Voltage (1 kHz Filter)	DC voltage bus after being filtered at 1kHz. This is the DC voltage bus used for fast motor control calculations.	• feedback.output.Vdc_1kHz
DC Bus Voltage (1 Hz Filter)	DC voltage bus after being filtered at 1Hz. This is the DC voltage bus used for fault detection.	• feedback.output.Vdc_1Hz

7.8.8 Motor

7.8.8.1 Screen

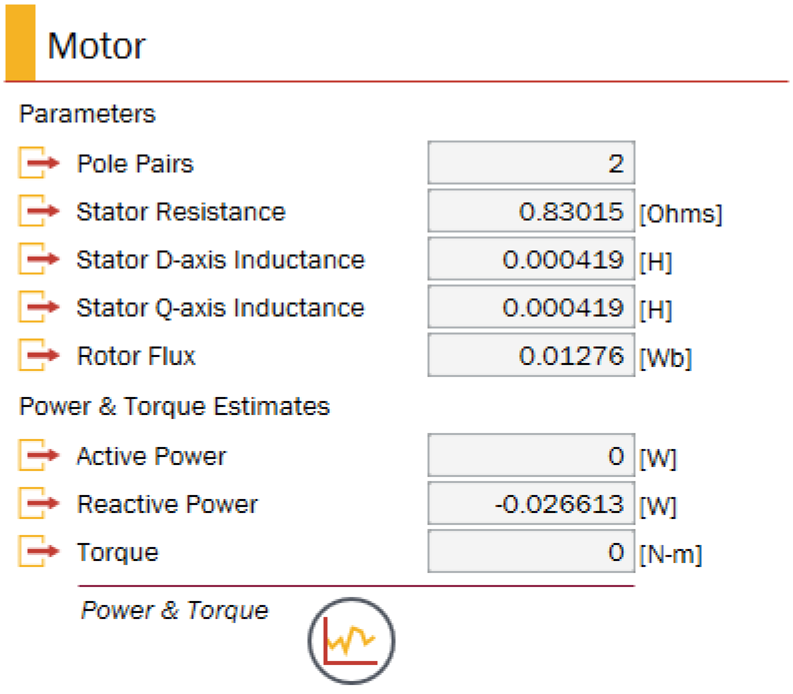


Figure 138. Motor step (sensorless velocity)

7.8.8.2 Description

The Motor step displays basic motor information, both what has been entered by the user and what has been measured by KMS. These values do not update on the Dashboard.

In contrast, real-time power & torque estimates do update in this step.

7.8.8.3 Parameters

Table 52. Motor parameters

Parameter	Description	Firmware variable
Pole Pairs	The number of pairs of magnetic poles in one mechanical revolution.	• flashSysParams.motorParams.polePairs
Stator Resistance	The per phase winding resistance of your motor.	• flashSysParams.motorParams.statorRes
Stator D-Axis Inductance	The per phase winding d-axis inductance of your motor at rated current.	• flashSysParams.motorParams.statorDInd
Stator Q-Axis Inductance	The per phase winding q-axis inductance of your motor at rated current. By default, this is assumed equivalent to d-axis inductance (non-salient motor condition).	• flashSysParams.motorParams.statorQInd

Rotor Flux	The rotor flux (or permanent magnet flux linkage) of your motor.	• flashSysParams.motorParams.pmFlux
Encoder Lines*	Number of lines (pulses) on encoder wheel	• flashSysParams.motorParams.encoderPulses
Active Power	Instantaneous active power being used by the motor. Estimated based on voltage and current.	• fluxest.output.activePower
Reactive Power	Instantaneous reactive power being used by the motor. Estimated based on voltage and current.	• fluxest.output.reactivePower
Torque	Instantaneous torque being provided by the motor. Estimated based on current and motor parameters.	• fluxest.output.torque

7.9 CPU utilization

KMS displays the CPU used by the motor and motion control code. This information is displayed on the CPU Utilization page (Figure 139).

Information is displayed for usage of:

- PWM (fast) ISR
- Slow ISR
- UART (communication) ISR

Pre-configured plots showing usage over time and the ability to clear & restart statistics are provided.

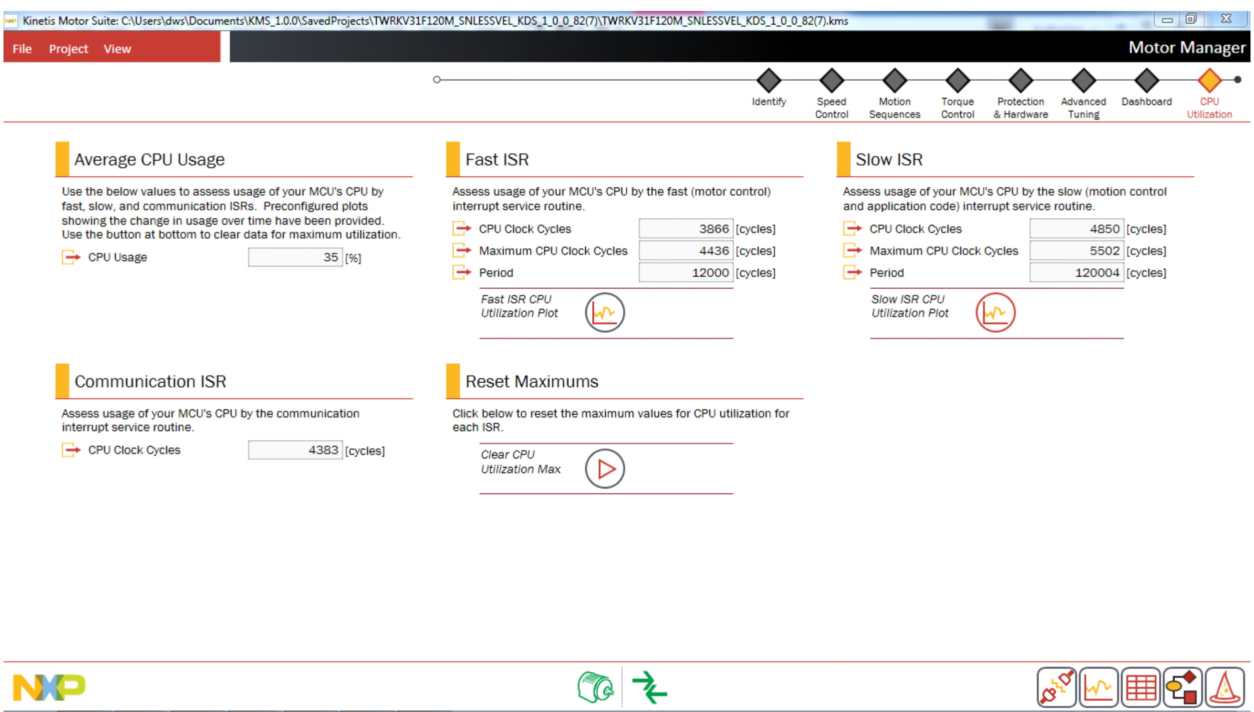


Figure 139. CPU utilization page

7.9.1 Average CPU Usage

7.9.1.1 Screen

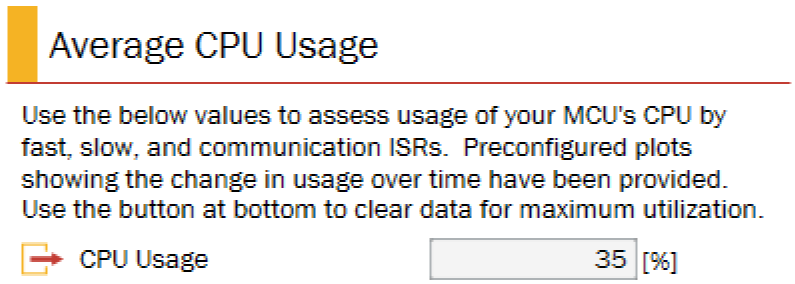


Figure 140. Average CPU Usage step

7.9.1.2 Description

The Average CPU Usage step displays the percentage of the processor used by the fast and slow interrupts combined.

7.9.1.3 Parameters

Table 53. Average CPU usage parameters

Parameter	Description	Firmware variable
CPU Usage	Percent of CPU consumed by the FOC algorithm.	• N/A

7.9.2 Fast ISR

7.9.2.1 Screen

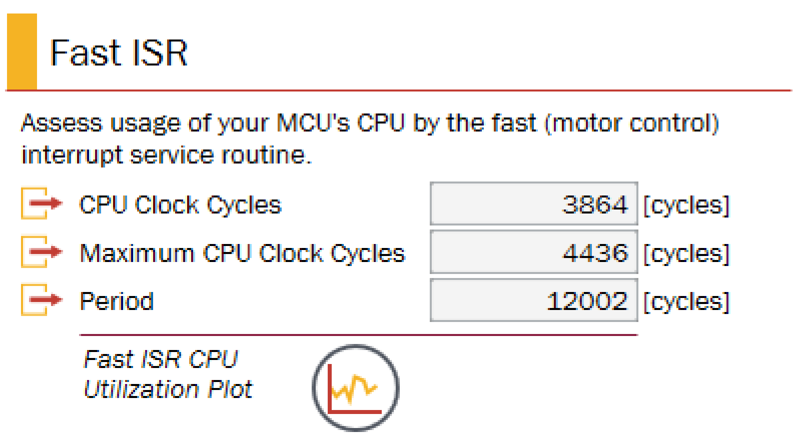


Figure 141. Fast ISR step

7.9.2.2 Description

Core motor control code operates in the fast interrupt service routine (ISR). Values displayed in the Fast ISR step show the cycles used by this routine as well as the cycles available. Refer to the Kinetis Motor Suite API Reference Manual for additional detail on the code that executes in the fast ISR.

7.9.2.3 Parameters

Table 54. Fast ISR parameters

Parameter	Description	Firmware variable
CPU Clock Cycles	Number of CPU cycles used by the Fast ISR.	• CpuUtilization.PwmIsrCycles
Maximum CPU Clock Cycles	Maximum number of CPU cycles used by the Fast ISR since the value was reset.	• CpuUtilization.PwmIsrCyclesMax
Period	Total number of CPU cycles available during the period of the Fast ISR.	• CpuUtilization.PwmIsrPeriod

7.9.3 Slow ISR

7.9.3.1 Screen

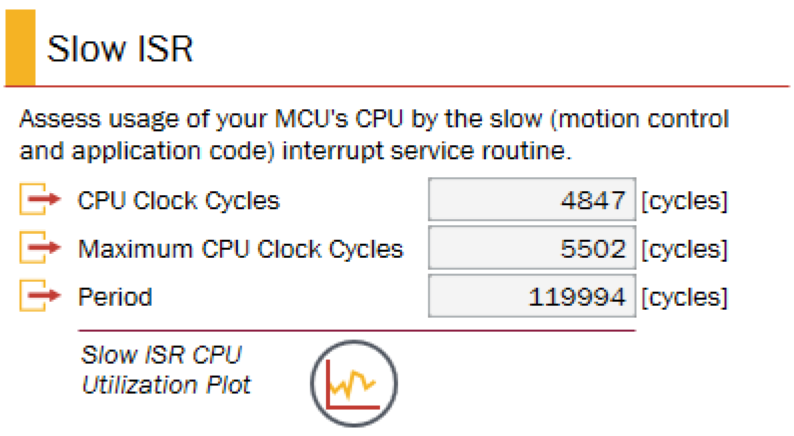


Figure 142. Slow ISR

7.9.3.2 Description

Motion control and application code operates in the slow interrupt service routine (ISR). Values displayed in the Slow ISR step show the cycles used by this routine as well as the cycles available. Refer to the Kinetis Motor Suite API Reference Manual for additional detail on the code that executes in the slow ISR.

7.9.3.3 Parameters

Table 55. Slow ISR parameters

Parameter	Description	Firmware variable
CPU Clock Cycles	Number of CPU cycles used by the slow motion & application control code. This value is inclusive of the Slow ISR being interrupted by the Fast ISR.	• CpuUtilization.SlowIsrCycles
Maximum CPU Clock Cycles	Maximum number of CPU cycles used by the Slow ISR since the value was last reset.	• CpuUtilization.SlowIsrCyclesMax
Period	Total number of CPU cycles available during the period of the Slow ISR.	• CpuUtilization.SlowIsrPeriod

7.9.4 Communication ISR

7.9.4.1 Screen

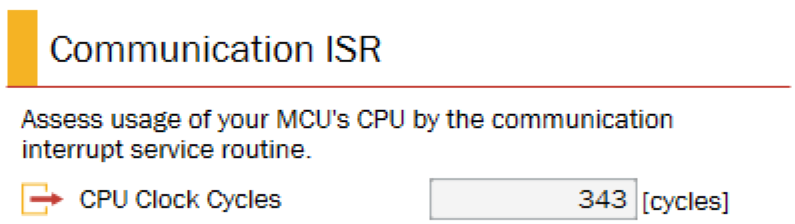


Figure 143. Communication ISR step

7.9.4.2 Description

KMS requires cycles to communicate from the PC to the MCU. The number of cycles, typically negligible in comparison to the cycles used by fast and slow ISR, are displayed in the Communication ISR step.

7.9.4.3 Parameters

Table 56. Communication ISR parameters

Parameter	Description	Firmware variable
CPU Clock Cycles	Number of CPU cycles used the Communication ISR.	• CpuUtilization.UartIsrCycles

7.9.5 Reset Maximums

7.9.5.1 Screen

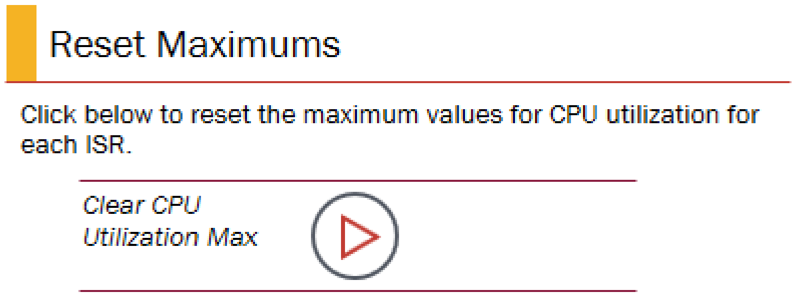


Figure 144. Reset Maximums step

7.9.5.2 Description

The Reset Maximums step allows for a reset of the Maximum CPU Clock Cycles values shown in the Fast and Slow ISR steps. This enables capturing maximum values during different operating points instead of simply providing an overall maximum.

8 Real time debugging

8.1 Software Oscilloscope

The Software Oscilloscope (Figure 145) is a critical tool for an engineer. The ability to visualize and save data for internal variables enables the application engineer to validate and compare configurations in order to optimize the application. The KMS GUI offers comprehensive plotting functionality, based on OxyPlot (http://oxyplot.org/) that provides pre-configured plots for typical operations while allowing the flexibility for the user to create custom plots.

8.1.1 Operation

The Software Oscilloscope is accessible from a number of places within KMS. Once activated, available options include:

- Select a previously configured plot
  - Certain plots are provided out of the box. For example, the Speed Feedback plot, which tracks actual motor speed vs. desired motor speed. These plots are available in the drop-down menu at the bottom left of the **Software Oscilloscope** and are enumerated in Table 57.

Table 57. Preconfigured plots

Name	Plots...
DCBus	Filtered DC bus voltage and fault thresholds for over and under voltage
DQCurrents	Commanded and actual currents along direct and quadrature axes
DQVoltage	Commanded reference voltages along direct and quadrature axes
PhaseCurrents	Motor feedback current along each of the three phase
PowerTorque	Estimates for active and reactive power as well as torque
PwmIsrCpuUtilization	CPU cycles used for the Fast (motor control) ISR compared to available cycles
SlowIsrCpuUtilization	CPU cycles used for the Slow (motion control/application) ISR compared to available cycles
SpeedFeedback	Commanded vs. actual speed
TrajectoryTestPlot	Theoretical kinematic values given user configuration of test trajectory
PosFeedback*	Commanded vs. actual position

*Only available in sensored position control projects.

  - Once the user has configured and saved a plot, this new plot appears as an option in the drop-down menu as well.
- Add, edit, or delete a plot
  - To customize plots, select from one of the three buttons next to the drop-down menu at bottom left. The Add button opens a dialog box to create a new plot from scratch; the Edit button opens

the same dialog box to enable revision of a previously defined plot; the Delete button eliminates the plot selected in the drop-down menu.

- Begin sampling
  - — The Run button initiates sampling from the MCU and starts plotting. A text field to the right of the Run button indicates status. The Run button toggles to a Stop button (to allow for termination of sampling) after sampling commences.
- Apply grid lines, markers, and/or connecting lines to plotted variables
  - — Three toggle buttons change the visualization of plotted data. Background grid lines, markers to indicate sampled data points, and lines connecting data points are individually accessible.
- Save as CSV or PNG
  - — Plotted data may be saved as either a comma separated value file or an image. Two buttons at bottom right present these options.

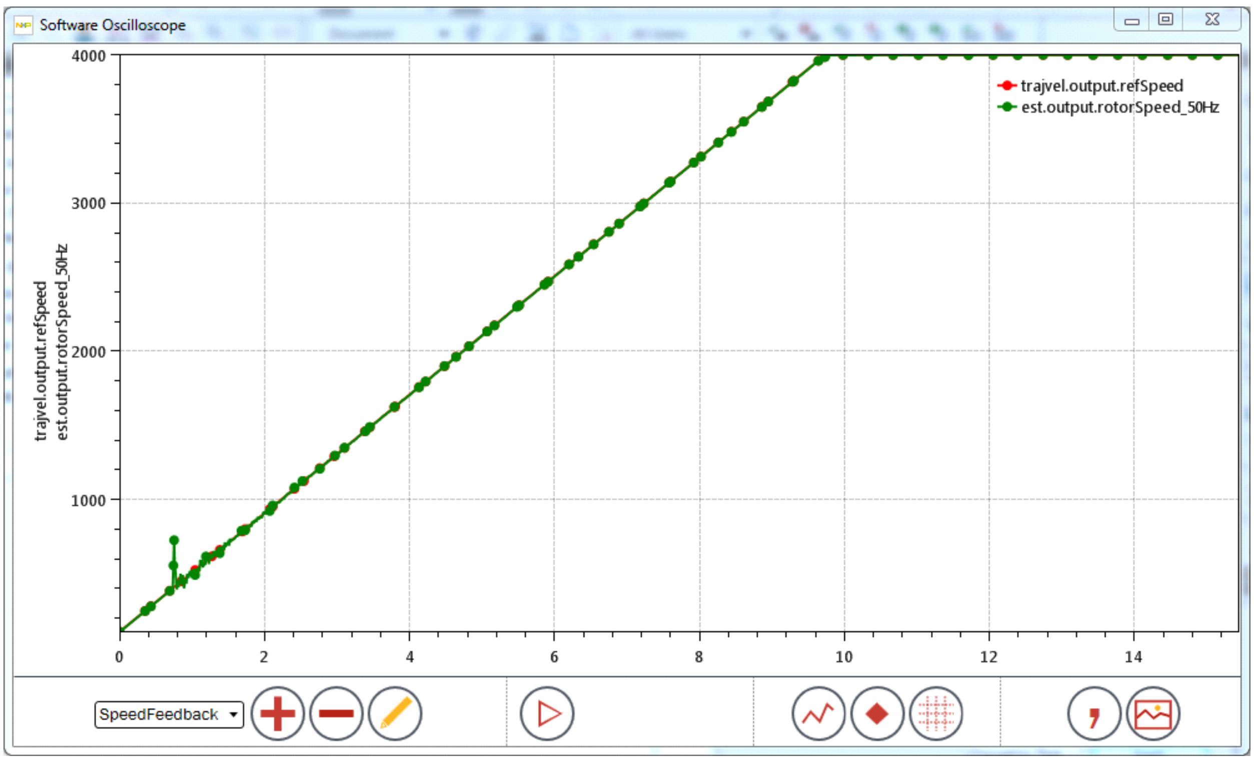


Figure 145. Software Oscilloscope with example data plotted

8.1.2 Define a custom plot

To define a custom plot, the user accesses the *Plot Definition* dialog box (Figure 148) via the Add or Edit buttons.

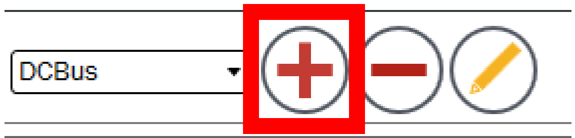


Figure 146. Add button

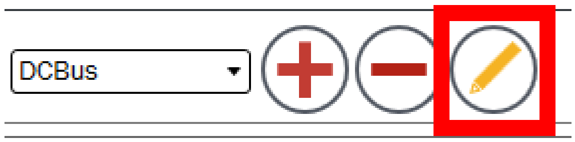


Figure 147. Edit button

All parsed data points from the KMS output file are available and may be included in a plot. Following are the plot attributes:

- Name
  — A user-defined name for the plot so that it may be saved and displayed in the drop-down menu.
- Available Variables
  — The list of available variables. This includes the main motor and motion control variables utilized by the KMS embedded firmware but may also include application-specific variables added by the user in IDE during application development. Note: If a variable is not displayed in the list, the user may need to add a parsed symbol in the **Watch Window**. The variable is subsequently available to use in a plot.
  — After clicking on a variable in the list, the user must click the Add button beneath the Available Points section to push the chosen variable into the plot. This is successful if the chosen variable appears in the Log Points at the bottom of the dialog box.
- Sample Rate
  — The rate at which the plot updates values. This may be defined by electrical Frequency (in hertz) or by interrupt service routine (ISR) Tick, according to user preference. KMS provides bounds on these values and the user must type in the desired value.
- Buffering
  — The Software Oscilloscope offers two modes for capturing data: Fill and Stop, and Continuous. Fill and Stop means that the plot proceeds until the specified buffer has been filled. Continuous means that data capture continues without termination until otherwise commanded. The buffer can be specified according to a quantity of bytes or data points sampled.
- Triggering
  — The Software Oscilloscope allows data capture to commence on a trigger event instead of on a button click to begin sampling. Trigger events are configured by defining the:

   – Variable to be evaluated
   – Trigger Type (Equal, Not_equal, Greater_than, etc.)
   – Value against which the variable's value is being evaluated
   – Pre-trigger behavior (counts prior to trigger being initiated).
  — In this configuration, data capture begins only after the specified condition is met.
- Log Points
  — Reflects the variables chosen for plotting and provides options for how the plotting is presented. A total of four data points may be included in a plot. Four independent axes are available for tracking these data points, each of which may automatically scale the y-axis according the value of the associated variable(s) or may have defined minimum and maximum y-axis values. The x-axis view is determined by the Sample Rate and Buffering configurations.

After defining a plot, the user must click the Update button (Figure 149) at bottom right to ensure that the plot can be accessed at a later time from the drop-down menu.

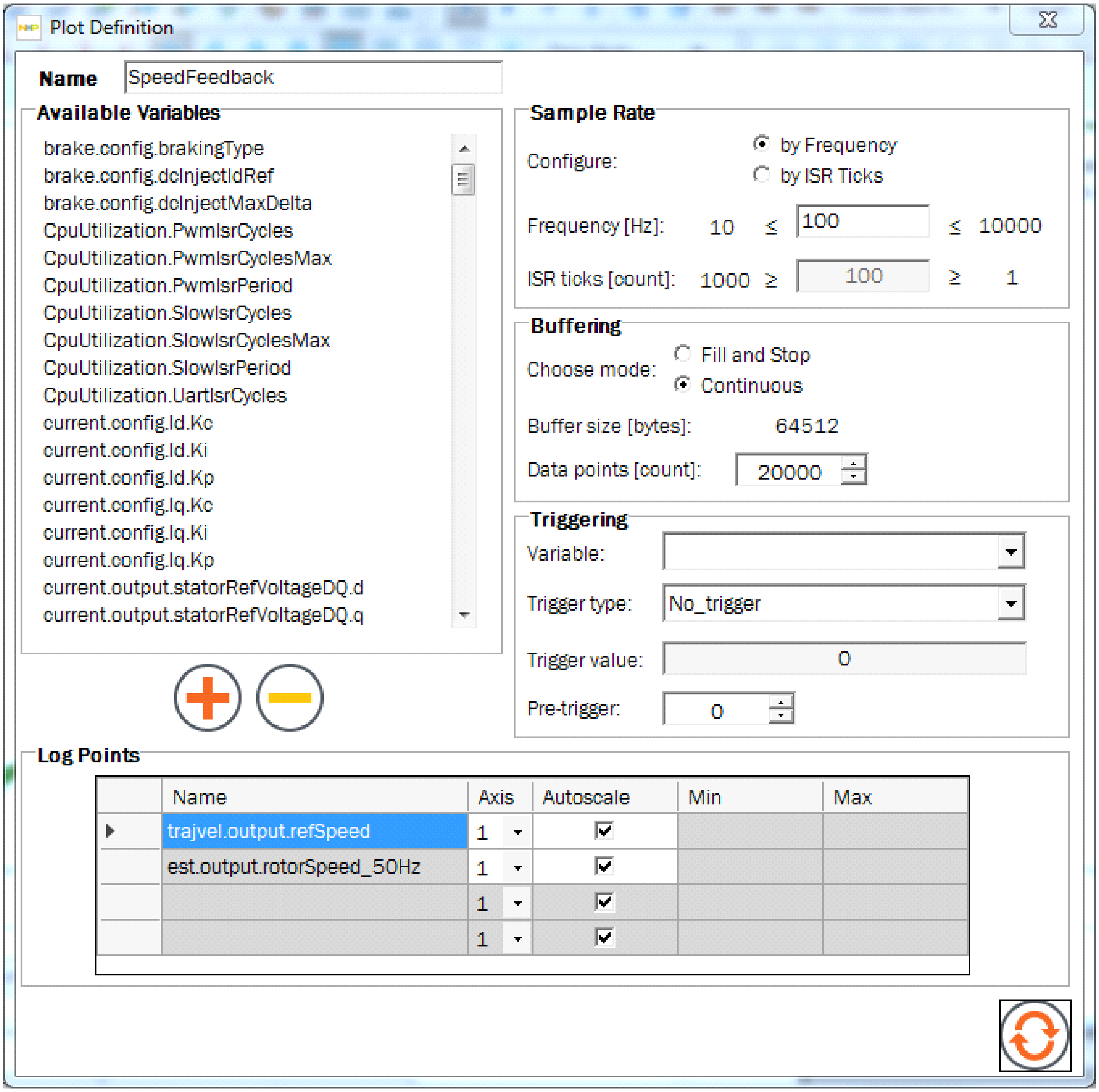


Figure 148. Plot definition

Figure 149. Update button

8.1.3 Interacting with the Software Oscilloscope while plotting

Users can zoom/pan/and reset a single plot axis by positioning the mouse cursor over the axis before starting the zoom/pan. Table 58 identifies the gestures (from http://oxyplot.org/) that can be used to perform these actions:

Table 58. Plot actions

Action	Gesture
Pan	Right mouse button
Zoom	Mouse wheel
Zoom by rectangle	Ctrl+Right mouse button
Reset	Right mouse button double-click
Show ‘tracker’	Left mouse button
Reset axes	‘A’

8.2 Watch Window

Through the **Watch Window** (Figure 151), users may add, edit, and view internal variables in real-time. This tool allows for rapid testing and feedback during application design.

8.2.1 Operation

The Watch Window is available from the View menu and from the button in the activation bar of the KMS GUI.

Figure 150. Click to show Watch Window

Options available within the **Watch Window** include:

- Add, edit, or delete a variable
  - — The **Watch Window** presents three buttons at bottom left to allow the user to view all relevant variables parsed from the MCU. This may include user-defined variables after a compile & download.
  - — The Add button opens a *Variable Definition* dialog box to create a new variable from any of the parsed symbols; the Edit button opens the same dialog box to enable revision of a previously defined data point; the Delete button eliminates the variable from the **Watch Window**. Note: to select a variable, click on the whitespace to the left of the row.
- Begin updating
  - — The Run button initiates sampling from the MCU and starts updating values. The Run button toggles to a Stop button (to allow for termination of sampling) after sampling commences.
- Columns

— The columns that appear in the **Watch Window** may be adjusted by clicking on the Header row. A dialog box appears allowing individual columns to be toggled on and off.

— The columns may be resized by hovering over cell dividing lines and clicking & dragging.

- Cell values

— Only the column “Value” allows an individual cell value to be modified. The Edit button must be used to modify the values of cells in other columns.

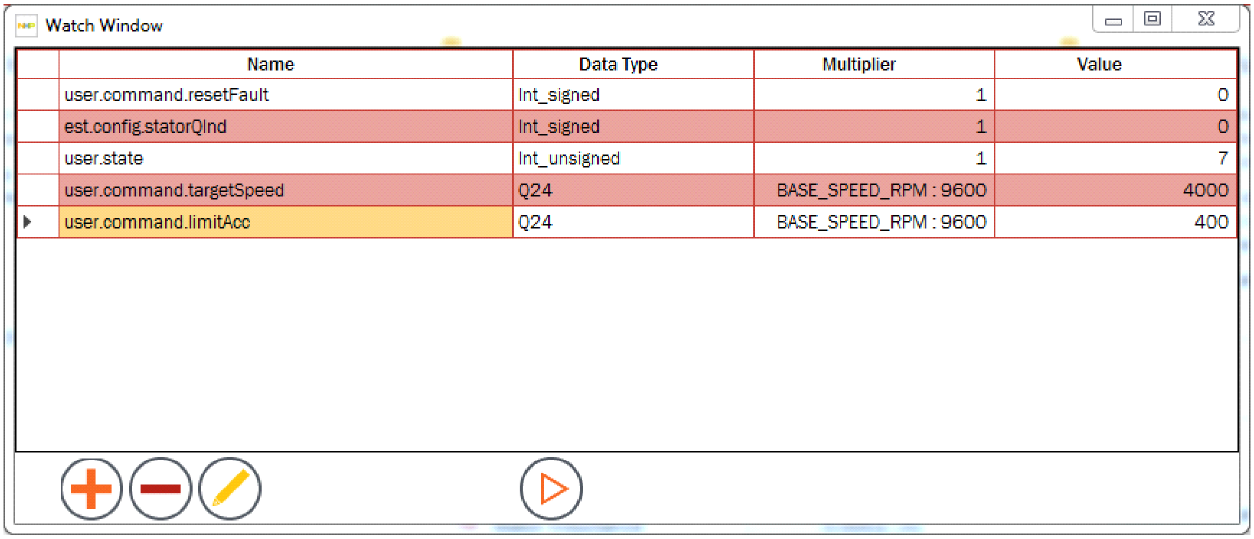


Figure 151. Watch window

8.2.2 Variable definition

Use the Add or Edit buttons to view and customize a variable in the **Watch Window**. When Add or Edit is clicked, the system displays the *Variable Definition* dialog box (Figure 152), which presents all parsed variables from the KMS output file. Certain key variables cannot be edited but the remainder allows for configuration of various attributes:

- Name

— The name of the internal variable derived from the output file associated with KMS.

- Available Variables

— The list of available variables. This includes all relevant motor and motion control variables utilized by the KMS embedded firmware, but may also include application-specific variables added by the user in the IDE.

- Data Type

— The nature of the data being stored; options include signed & unsigned integers, raw bytes, and Q values from Q0-31.

- Data Format

— The format of the data being stored; options include Decimal, Hexadecimal, Binary, and String.

- Multiplier
  — An adjustable scaling factor. For convenience, the KMS system constants are provided but custom value can also be typed into this box.
- Offset
  — An adjustable offset value.
- Units
  — The unit of measurement for the data point.
- Decimal Places
  — The number of places to the right of the decimal point that are visible.

After customizing, the user must click the Update button at bottom center to ensure that the **Watch Window** updates with the new data point.

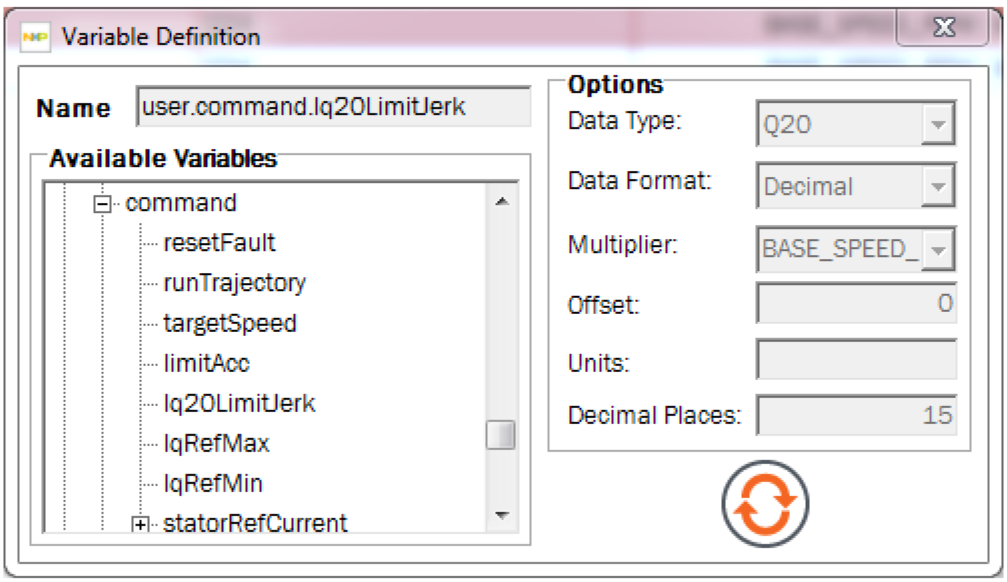


Figure 152. Variable definition dialog box example (uneditable variable)

9 Motion Sequence Builder

After identifying and tuning the motor, use Motion Sequence Builder to generate code for the application's motion sequence from a graphical environment.

A motion sequence, or **Plan,** in the terminology of the Motion Sequence Builder, describes the states at which the application should run under certain conditions, and the nature of transitions between these states.

There are two different implementations of Motion Sequence Builder: one for velocity control and one for position. However, the underlying functionality is common across both, so the velocity control case is considered here, with reference to key differences to position control where appropriate.

One example of a velocity control motion sequence is the operation of a simple ceiling fan. The fan starts at 0 rpm. Each time the speed button is pushed, the fan advances to the next speed.

- 0 to 1000 rpm
- 1000 to 2500 rpm
- 2000 to 4000 rpm
- 4000 to 0 rpm

Figure 153 displays the state diagram generated for the ceiling fan by Graphviz (this is the purpose of the Graphviz installation from Section 3, "Installation"). The ceiling fan example is provided in KMS and is used for explanation purposes throughout this section.

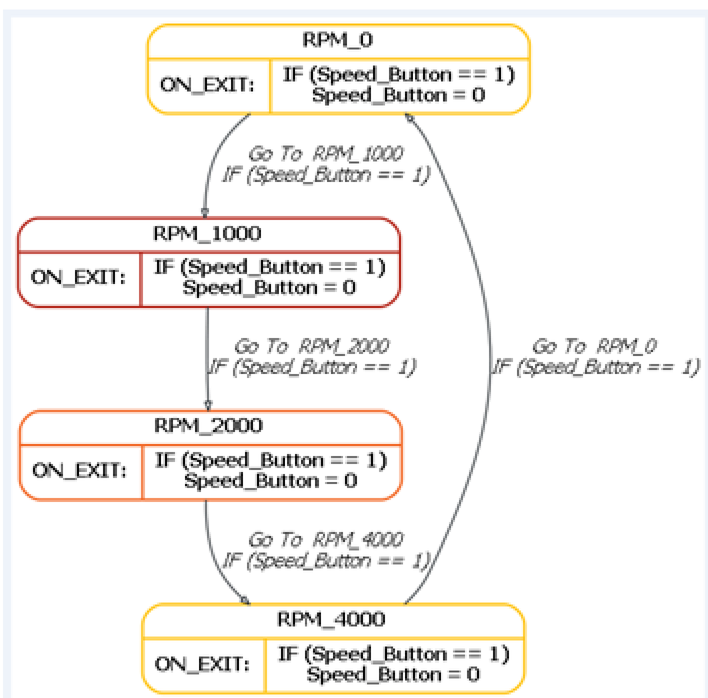


Figure 153. State diagram for simple ceiling fan

Motion Sequence Builder enables graphical configuration of **Plans** and automatically generates code that can be:

- executed from KMS
- edited from the IDE

Code generated from the Motion Sequence Builder by default runs in the slow interrupt service routine (Slow ISR).

Construction of a **Plan** in the Motion Sequence Builder is an optional step and is best suited for applications with well-defined and consistent states of motion.

Once the **Plan** code is modified in the IDE, it cannot be edited in Motion Sequence Builder. Motion Sequence Builder does not recognize manual code adjustments.

Motion Sequence Builder generated code can reference user-created variables to link control of the application's motion with non-motor control components like sensors and actuators.

9.1 Access Motion Sequence Builder

Motion Sequence Builder can be invoked from several places in KMS:

- the KMS View Menu (Figure 154)
- the Motion Sequences page in Motor Manager (Figure 155)
- the Next Steps page in Motor Tuner (Figure 156)
- The activation button in the lower right hand corner of KMS (Figure 157)

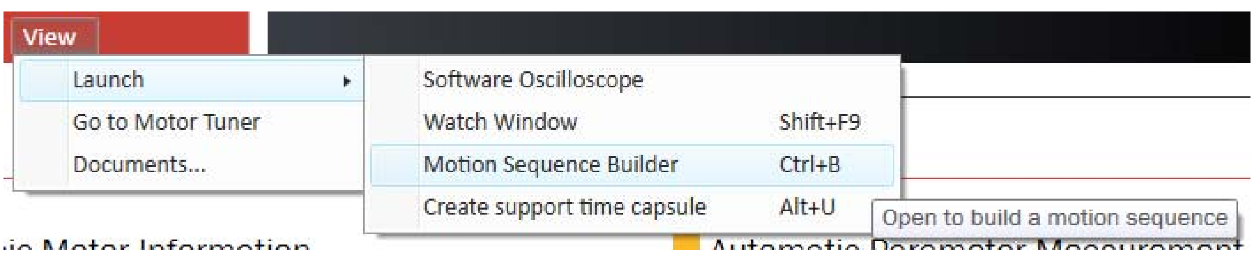


Figure 154. View menu with start Motion Sequence Builder command highlighted

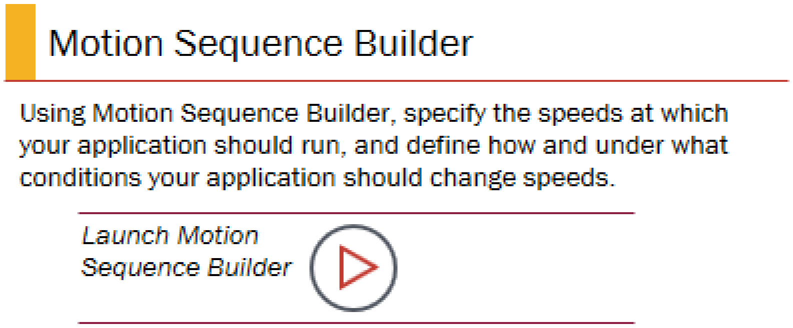


Figure 155. Motor Manager Motion Sequences page

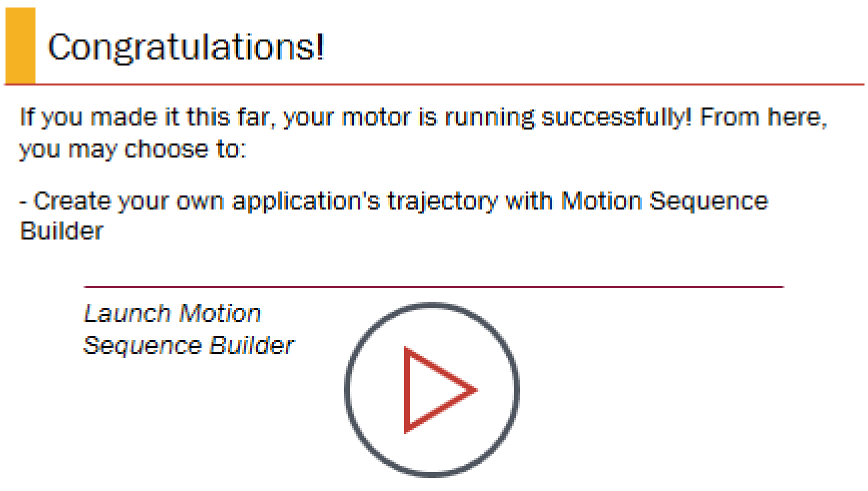


Figure 156. Motor Tuner Next Steps page access to Motion Sequence Builder

Figure 157. Access Motion Sequence Builder from activation button

9.2 Build a motion sequence

When opened, Motion Sequence Builder automatically generates a valid plan that consists of 2 states and 2 transitions (Figure 158).

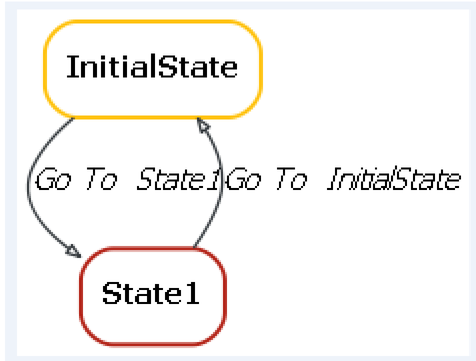


Figure 158. State diagram for default motion sequence

Users can choose to execute a Motion Sequence a single time (box is unchecked), or have the Motion Sequence repeat continuously (Figure 159).

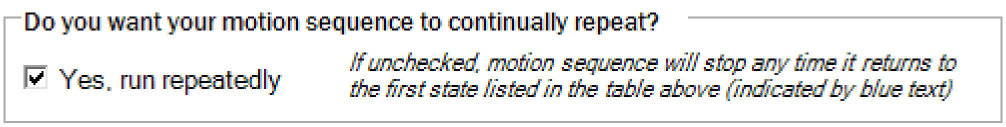


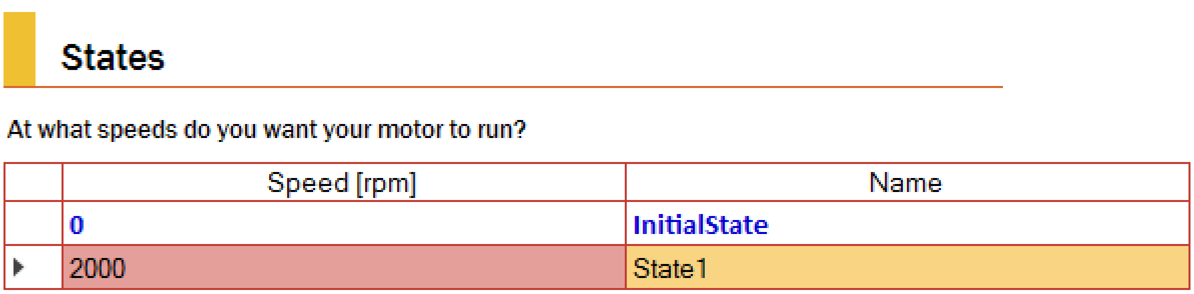
Figure 159. Continuously repeat the motion sequence

If the box is unchecked, the motion sequence will terminate when it returns to the Initial State. If the motion sequence is to run repeatedly, it will wait in the Initial State until a true condition occurs.

9.2.1 States

States describe consistent operations of the motion sequence.

For the velocity control version of Motion Sequence Builder, the user specifies the speed at which the motor runs while in each state (Figure 160).



	Speed [rpm]	Name
	0	InitialState
▸	2000	State1

Figure 160. Define the speed at which the motor runs while in each state (velocity control)

For position control, the user specifies the number of full and partial revolutions that the motor completes - a state is basically a point at a user-specified rotational distance from the current state.

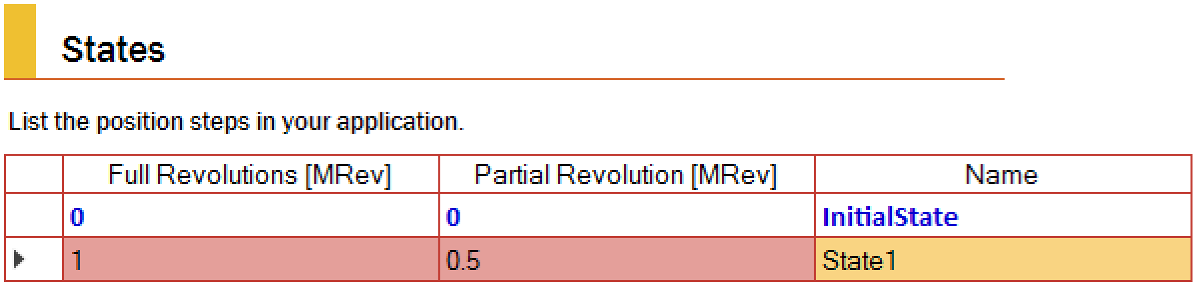


Figure 161. Define the revolutions the motor completes while in each state (position control)

- States are created by clicking the Add button (Figure 162).

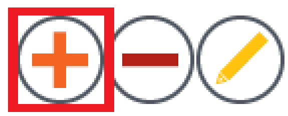


Figure 162. Add a state

- States are removed by selecting the row(s) and clicking the Delete button (Figure 163). Motion Sequence Builder requires at least two states at all times, and prevents the user from deleting more than the minimum requirement (Figure 163).

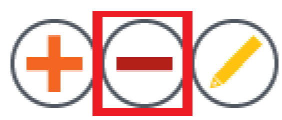


Figure 163. Delete a state

- States are edited by selecting a row and clicking the Edit button (Figure 164). This takes you to the selected state's definition page (see Section 9.2.2, "State definition"). This can also be achieved by clicking on the appropriate state name at top left (Figure 164).

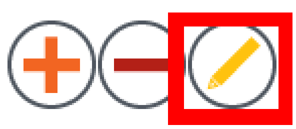


Figure 164. Edit a state

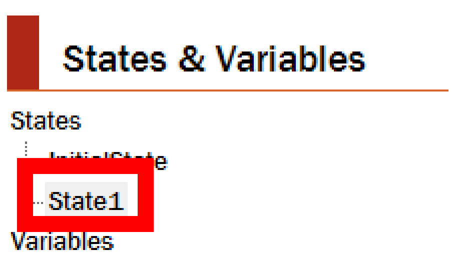


Figure 165. Click on state name to edit

- The first State in the table is the Initial State (Figure 166), and it has special properties. The Motion Sequence stops any time it returns to this state, unless the user has indicated that the plan should run continuously.

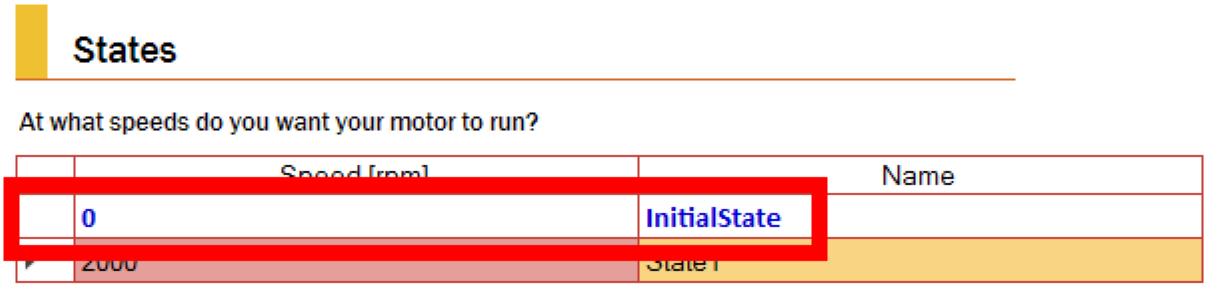


Figure 166. InitialState has special properties

- States may be reordered by selecting the row and moving it up or down using the arrow buttons (Figure 167). The system does not allow a user to replace the Initial State, and presents a notification if the user tries to do so (Figure 168).

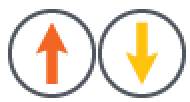


Figure 167. Reorder states

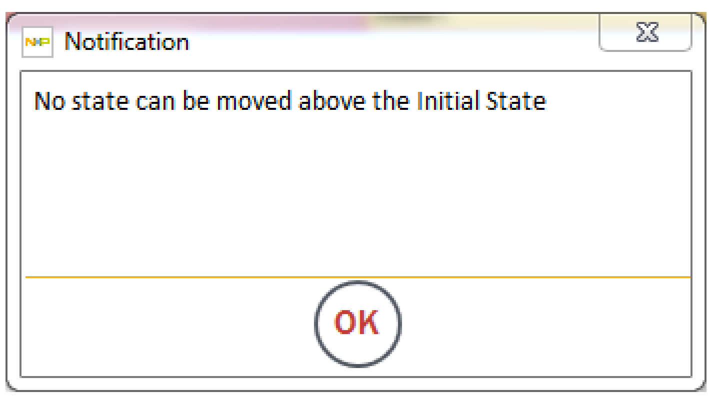


Figure 168. Cannot replace InitialState

- Many motion sequences are circular. For instance, in the ceiling fan example, the fan proceeds from InitialState to Low Speed to Medium Speed to High Speed then back to the InitialState. If the user selects the Connect States button (Figure 169), Motion Sequence Builder automatically generates transitions between all States in the table, creating a circular motion sequence.

Figure 169. Connect states

NOTE

If the user has specified different transitions on the State Definition pages, the Connect States button will overwrite these transitions.

9.2.2 State definition

From the State Definition page (Figure 171), users can specify the behavior of the system while in the selected state. For example, Figure 170 shows that when the system is in the Low_Speed state, the motor runs at 1000rpm for a minimum of 1000ms. After 1000ms, the system begins evaluating the state's If...Then... conditions.

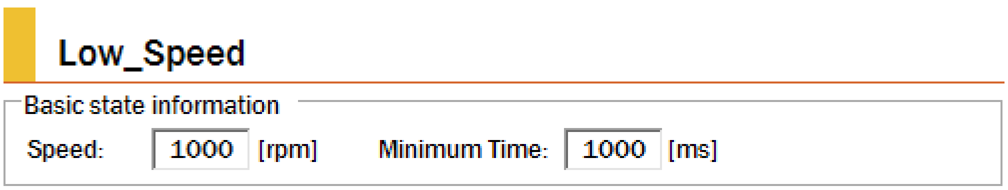


Figure 170. Basic state information

If... statements specify a Condition that must be satisfied. Then... statements specify an action or transition to another speed that will occur when a condition is satisfied. In other words:

- If <condition = True> Then <execute the transition or action>

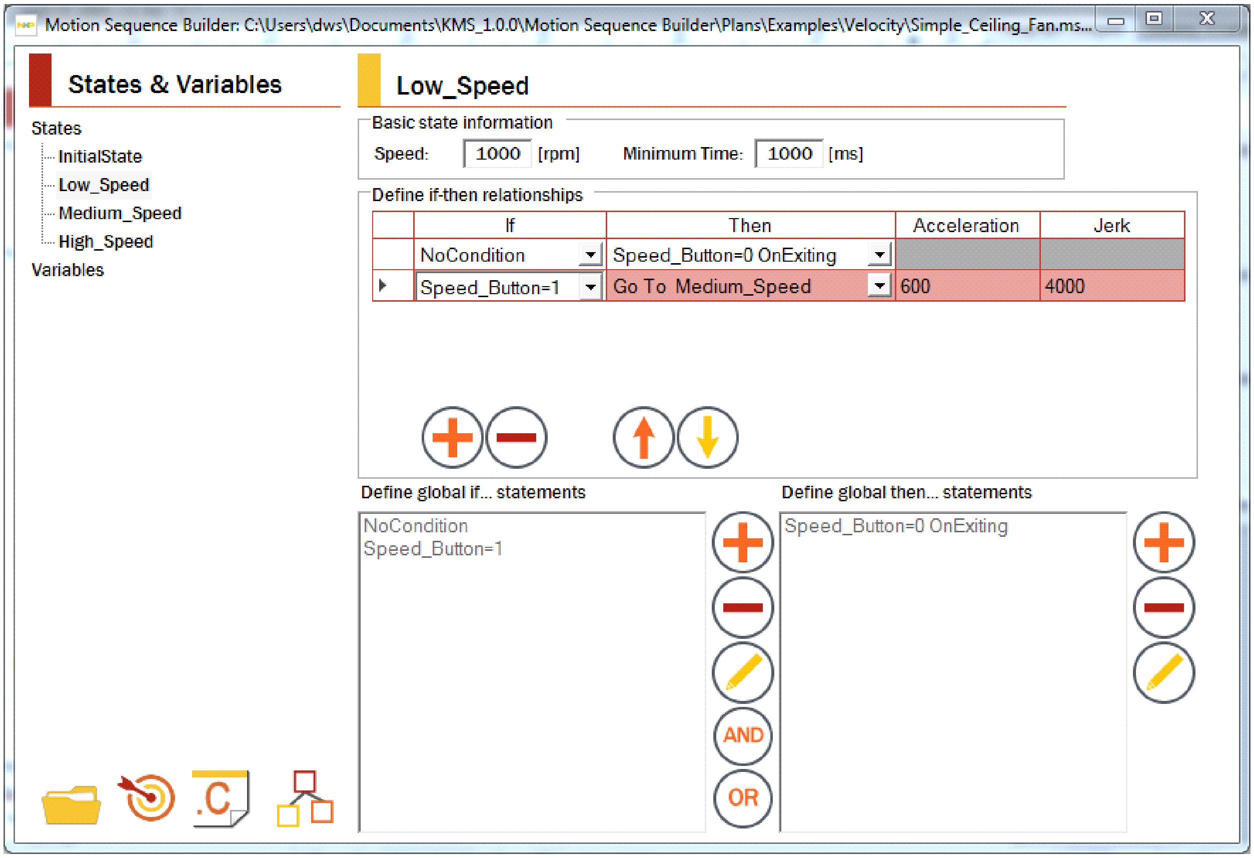
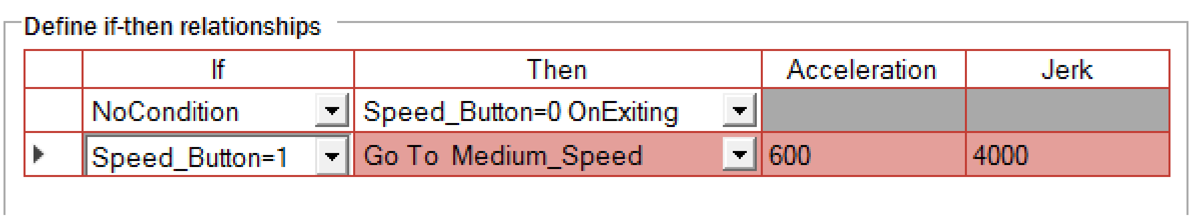


Figure 171. State definition screen

For example, in the Ceiling Fan motion sequence, the fan transitions from Low_Speed to Medium_Speed when the Speed_Button is enabled (set to 1). The system clears the Speed_Button (set to 0) when it leaves the current state. This behavior is represented by the If...Then statements shown in Figure 172.


Define if-then relationships

	If	Then	Acceleration	Jerk
	NoCondition	Speed_Button=0 OnExiting		
▸	Speed_Button=1	Go To Medium_Speed	600	4000

Figure 172. State Definition screen for one state within the simple ceiling fan motion sequence

Once If... and Then... statements are created, they can be reused when defining any other State.

9.2.2.1 Define If... statement conditions

If... statements provide logical checks within the system. The condition(s) in the statement must be satisfied before the motor can transition to another speed, or the system can perform an action. Figure 173 provides an example of global If... statements.

Figure 173. Global if… statements

To determine whether a condition is satisfied, a Variable is compared against a specific value, value range, or another Variable. This returns a true or false value based on the criteria (see Figure 174 and Figure 175).

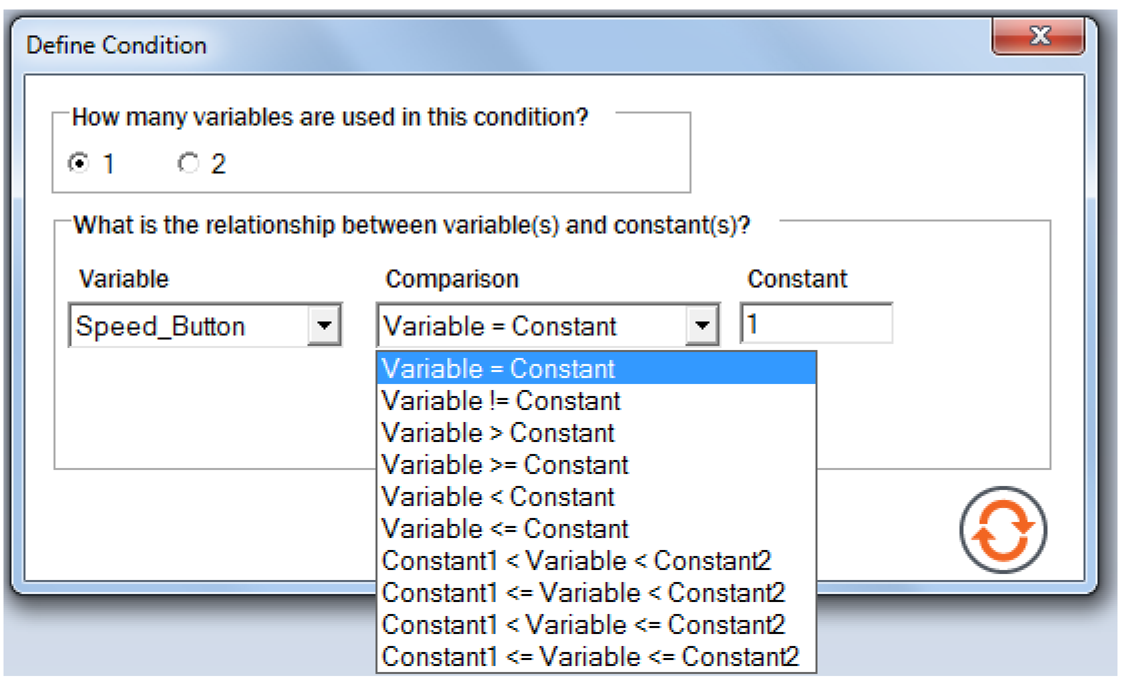


Figure 174. Variable is compared against a specific value or value range

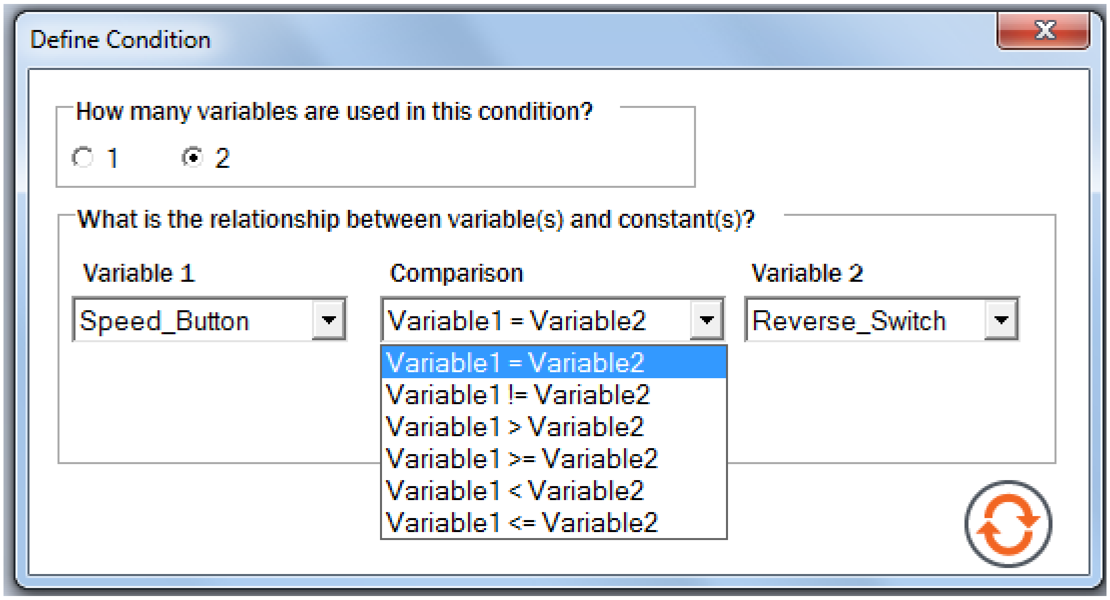


Figure 175. Variable is compared against another variable

- Conditions are created by clicking the Add button (Figure 176).

Figure 176. Add global condition

- Conditions are removed by selecting the condition and clicking the Delete button (Figure 177).

Figure 177. Delete global condition

- Conditions are changed by selecting the condition and clicking the Edit button (Figure 178).

Figure 178. Edit global condition

- NoCondition is a special case. When "No condition" is used with a transition, the motor transitions from the current state to a new state when the minimum time expires (Figure 179). When used with an action, the system performs the action immediately upon entering or exiting a state.

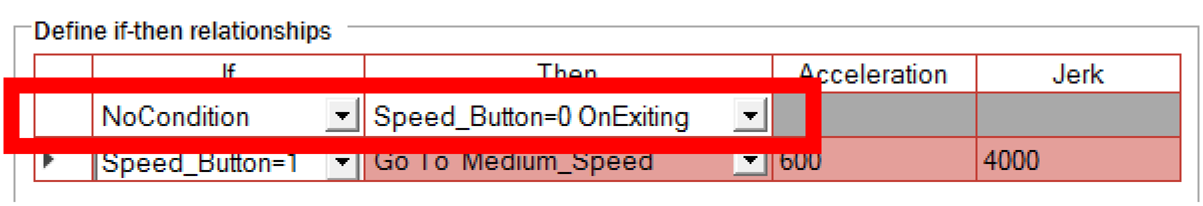


Figure 179. NoCondition

- Complex conditions can be created by selecting two individual conditions and combining them by clicking either the "And" or "Or" button, as shown in Figure 180.

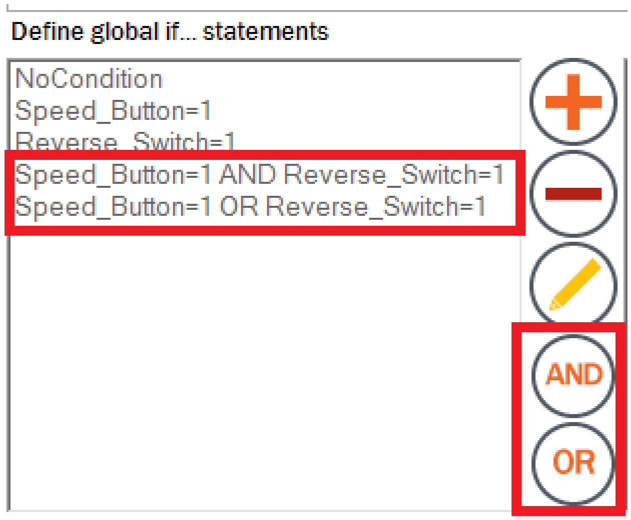


Figure 180. Complex conditions

9.2.2.2 Define then... statement actions

Then... statements specify the actions that occur in the system if a condition is true.

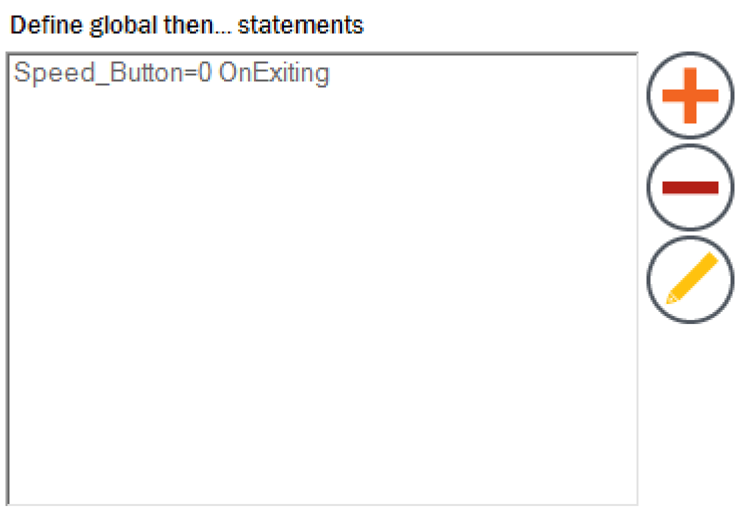


Figure 181. Global then... statement

Actions can set a variable equal to a value, or actions can increment the variable by a number (positive or negative) (see Figure 182).The user specifies whether the action occurs when entering or exiting the state.

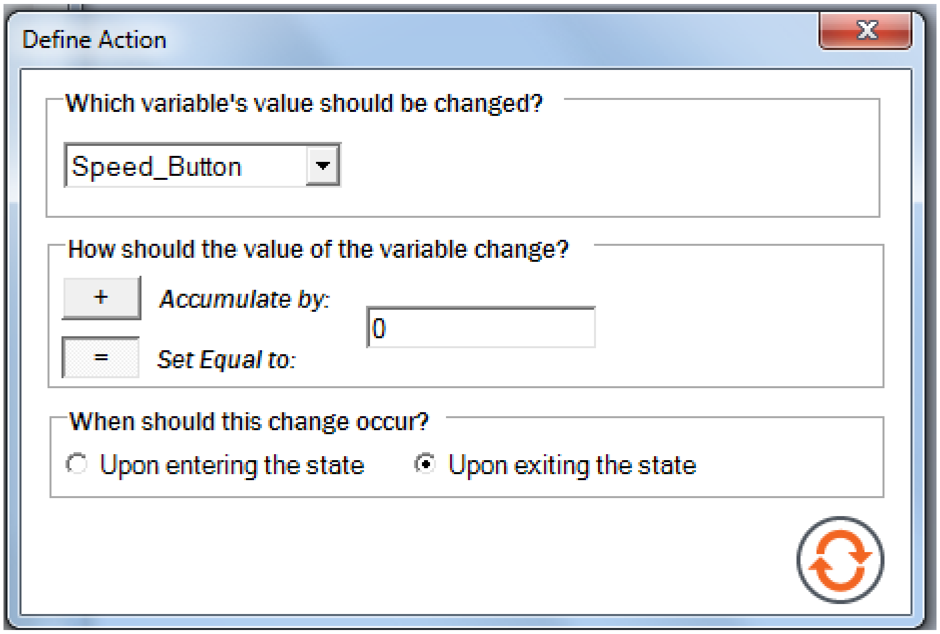


Figure 182. Define action

- Actions are created by clicking the "Add" button (Figure 183).

Figure 183. Add a global action

- Actions are removed by selecting the condition and clicking the "Delete" button (Figure 184).

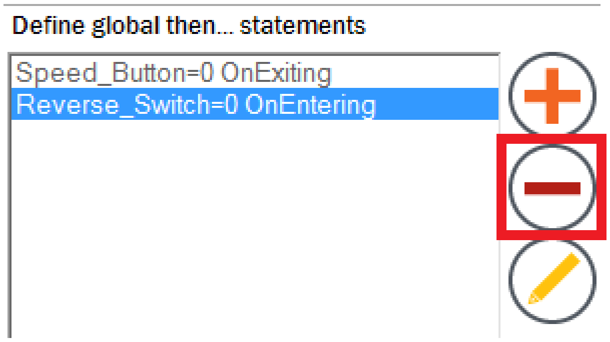


Figure 184. Delete global action

- Actions are changed by selecting the condition and clicking the "Edit" button (Figure 185).

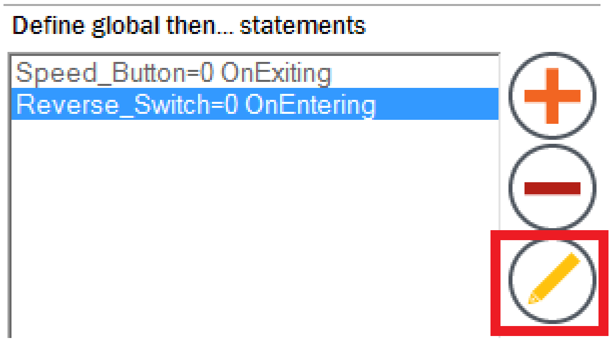


Figure 185. Edit global action

9.2.2.3 Define then... statement transitions

Transitions define the allowable movements between States, and they establish the connections between the States. In Motion Sequence Builder, transitions are structured as "Go to" statements. Motion Sequence Builder automatically generates a "Go to" statement for every State in the system. "Go to" statements may not be created or removed by the user.

Users can select the transitions from the State definition page (Figure 186).

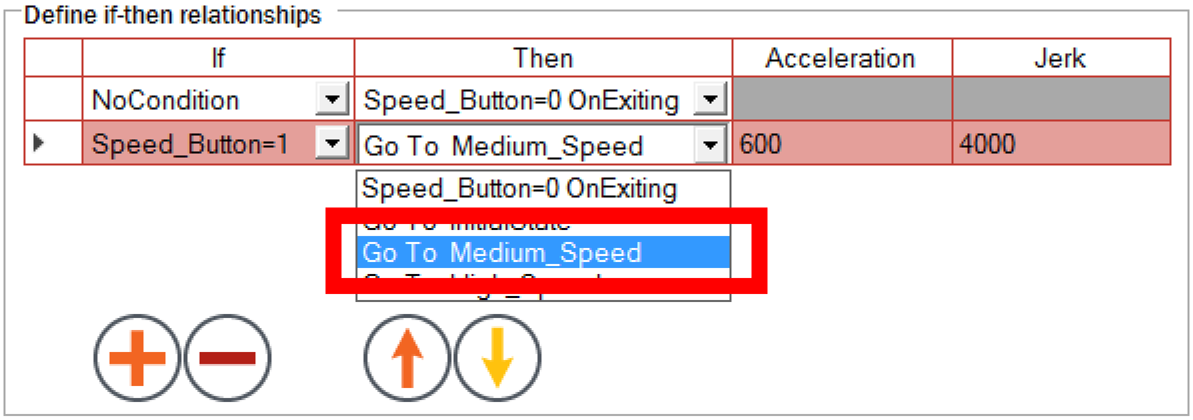


Figure 186. Select transitions from the state definition page

Users can automatically create a circular path between all states by clicking the "Connect States" button from the States page (Figure 187).

Figure 187. Motion Sequence Builder can automatically generate circular transitions

NOTE

If the user has specified transitions on the State Definition pages, the "Connect States" button will overwrite these transitions.

When the user defines a transition, the system automatically populates kinematic limits (Figure 188). The user can change these values. The Test Trajectory function in KMS (Section , "Each KMS-enabled MCU includes a simple demonstration plan that simulates the motion of a washing machine. Users can also build a motion sequence, or “plan,” using Motion Sequence Builder, then download the plan to the MCU. Users can run the plan from this page (Figure 113). Users can specify the Curve Type to be used by the plan from the Speed or Position Control pages (depending on control type)") can be used to arrive at the ideal parameters for the desired transition time.

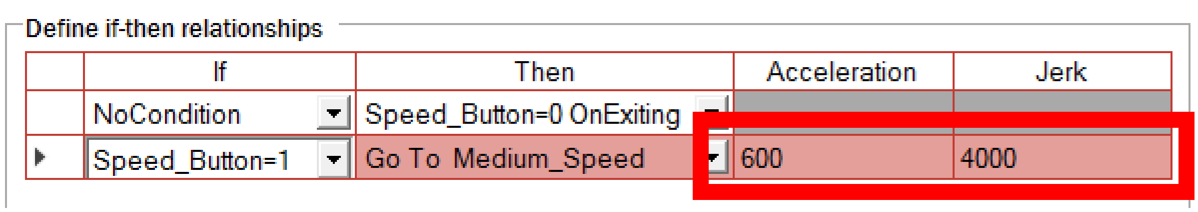


Figure 188. Define acceleration and jerk limits for transitions

9.2.3 Define variables

Variables add flexibility to the motion sequence. They allow the motion part of the application to interact with the rest of the application. For example, in the ceiling fan application, there is an external button that controls the speed of the fan. The button is included as a variable in the motion sequence.

Variables are created, removed, and modified from the Variable Definition screen (Figure 189).

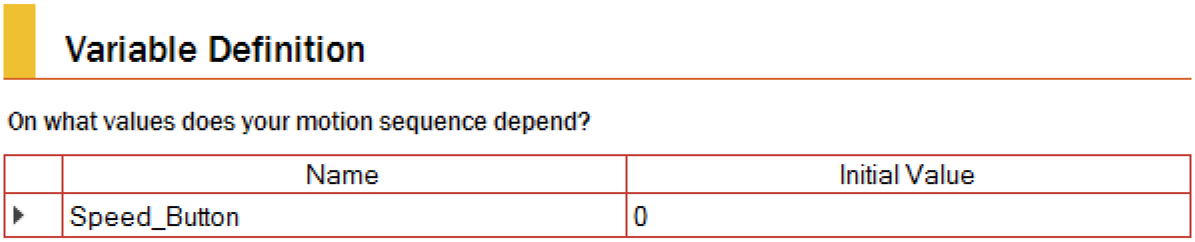


Figure 189. Variable definition screen

- Variables are created by clicking the Add button (Figure 190).

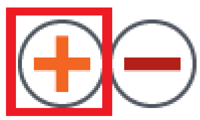


Figure 190. Add variable

- Variables are removed by selecting the row and clicking the "Delete" button (Figure 191). Note: When a variable is deleted, all If... and Then… statements associated with the variable are also deleted.

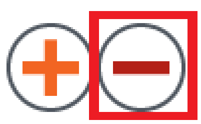


Figure 191. Delete variable

- Variables are renamed by selecting the variable name cell and entering the desired name. The name change is automatically reflected throughout the system.

9.3 Compile and run a motion sequence

After designing a plan in Motion Sequence Builder, the user can click the Generate Code button (Figure 192) to see the representation of the state machine in embedded C code.

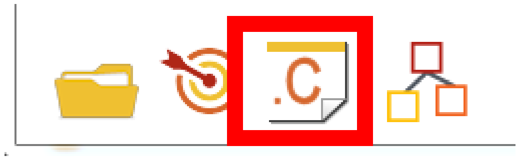


Figure 192. Generate and display source code

More importantly, Motion Sequence Builder can push this new motion sequence directly to the MCU for validation of operation. By clicking the Run on Target button (Figure 193) at the bottom left corner, the user initiates a command line build and download. This process relies on KMS knowing the location of KSDK and the preferred IDE. If the location of the preferred IDE has not previously been specified, KMS prompts for its path.

Figure 193. Run on target button

As part of the compile, KMS automatically saves the source code underlying the new motion sequence in the KMS project under the src (.c file) and inc (.h file) folders.

At this point, KMS leverages IDE functionality without opening the IDE to execute a build of the relevant KSDK and KMS files. The image is then downloaded to the MCU. The result is the same as opening the reference project workspace file within the IDE, building each project, and downloading the compiled code to the MCU.

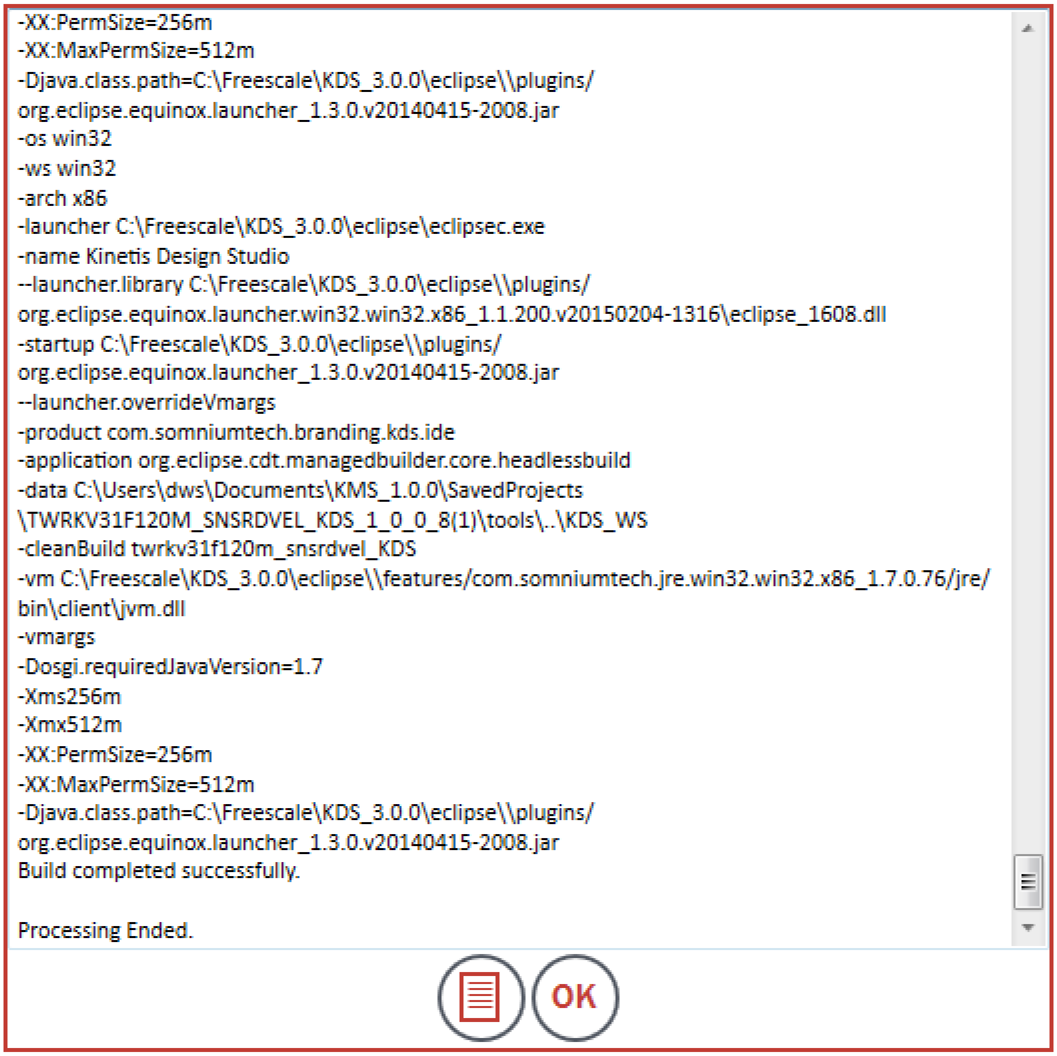


Figure 194. Example of compilation

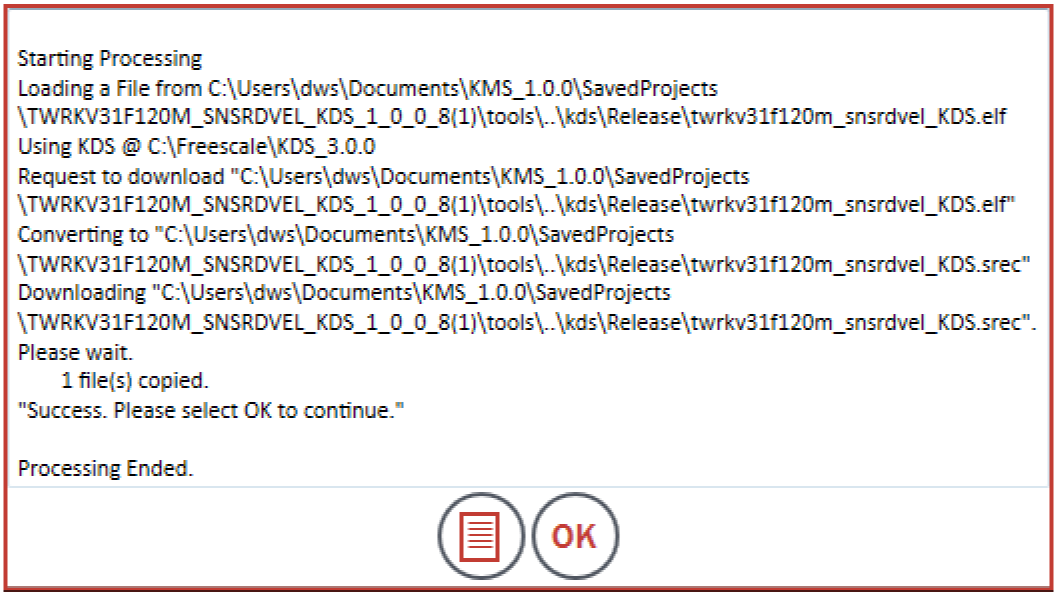


Figure 195. Example of downloading

Once the code has successfully been downloaded to the MCU, click OK and KMS reconnects communication between MCU and the KMS GUI. The user can then navigate to the Run Motion Sequence step on the Speed Control page, and click the Run Plan button(Figure 196) to start the plan. Behavior of the motor can then be monitored by opening the Software Oscilloscope and watching the motor transition through the specified States.

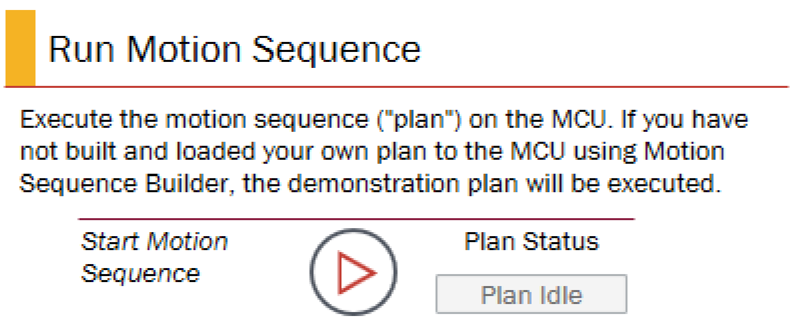


Figure 196. Run motion sequence

Refer to the Kinetis Motor Suite Lab Guide for hands-on instructions regarding how to build a simple motion sequence, validate operation, then add code manually in the IDE to more closely approximate an application design.

10 Faults

10.1 Definitions

Three types of firmware faults may be shown in the KMS user interface:

10.1.1 Motor faults - non clearable

Nonclearable faults typically relate to CPU faults including

- memory
- register
- clock

The MCU must be reset to proceed after a nonclearable fault.

10.1.2 Motor faults - clearable

Clearable faults typically refer to motor control protections including:

- overcurrent
- overvoltage
- loss of speed regulation

Clearable faults may be acknowledged and dismissed from the GUI. Work from the GUI can proceed after clearing faults.

10.1.3 Inertia identification error

Inertia identification errors indicate that the process of identifying system inertia has failed to produce an appropriate value. Inertia identification should be attempted again before proceeding (speed control depends on an inertia value) but other GUI actions are not prohibited if an inertia identification fault is asserted.

10.2 Display

On either type of motor fault (clearable or nonclearable), the Motor Status Indicator turns red and hovering over the motor displays a description of the asserted faults (Figure 197).

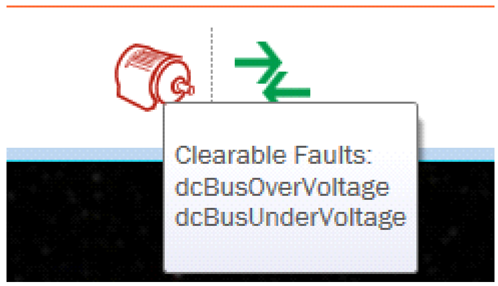


Figure 197. Example of fault assertion

When a clearable fault occurs, a dropdown box indicating type of fault appears, as does a Clear Faults button, which enables a return to normal operation.

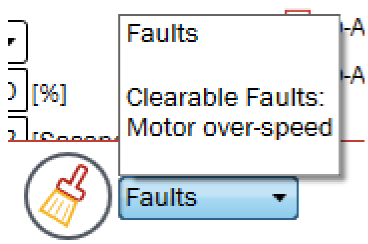


Figure 198. Fault dropdown and clear button

Table 59. Automatic parameter measurement parameters

Parameter	Description	Firmware variable
Clear faults	Clears asserted clearable faults and places motor in idle (stopped) operating mode.	• user.command.resetFault

When an inertia identification error occurs, an error code appears in the System Inertia Measurement step of the Identify page.

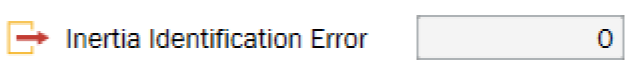


Figure 199. Location of inertia identification error notification

Common inertia identification errors can often be resolved by adjusting Inertia Identification Speed and Ramp Time (Figure 200).

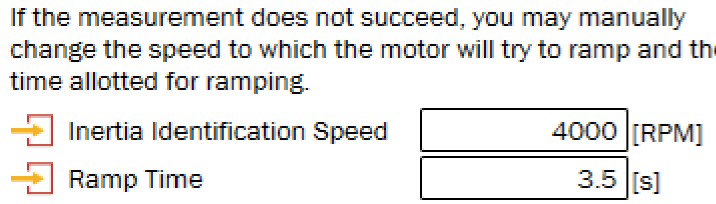


Figure 200. Editable fields for Inertia Identification Speed and Ramp Time

Common inertia identification error codes and suggested solutions are outlined below:

Table 60. Common inertia identification adjustments

Error Code	2003	2004		2006
Meaning	Bad estimation value	Process timeout		Motor stops during test
Motor Behavior		Motor spins	Motor starts slowly	
Solution	Decrease Ramp Time	Decrease Inertia Identification Speed	Decrease Ramp Time	Decrease Ramp Time
Commonly occurs in these applications	Automotive pumps	Washing machines	Compressors	High friction/ cogging force

*Ramp Time has valid range in seconds [sample time, 25.0].

10.3 Fault lists

Refer to the KMS API Reference Manual for the list of possible motor & system faults.

11 Resources utilized

KMS embedded firmware requires certain MCU resources to enable motor & motion control. Processor, memory, peripheral, and pin usage are enumerated in this section.

Performance values are intended to be illustrative only and not a guarantee of performance on the user's system. They depend on compiler, optimization settings, and application specifics.

11.1 Processor

The tables in this section describe CPU usage in terms of cycles and percentage by:

- control type
- development environment
- interrupt service routine

Table 61. CPU usage - sensorless velocity

IDE	IAR				KDS			
ISR	CPU cycles		% CPU at 120 MHz		CPU cycles		% CPU at 120 MHz	
	Typical	Maximum	Typical	Maximum	Typical	Maximum	Typical	Maximum
Fast	3723	4141	31.00	34.51	3870	4290	32.25	35.75
Slow	4783	5438	3.99	4.53	4850	5500	4.04	4.58

Table 62. CPU usage - sensored velocity

IDE	IAR				KDS			
ISR	CPU cycles		% CPU at 120 MHz		CPU cycles		% CPU at 120 MHz	
	Typical	Maximum	Typical	Maximum	Typical	Maximum	Typical	Maximum
Fast	2930	2950	24.42	24.58	3022	3024	25.18	25.20
Slow	4422	5608	3.69	4.67	4506	5655	3.76	4.71

Table 63. CPU usage - sensored position

IDE	IAR				KDS			
ISR	CPU cycles		% CPU at 120 MHz		CPU cycles		% CPU at 120 MHz	
	Typical	Maximum	Typical	Maximum	Typical	Maximum	Typical	Maximum
Fast	2947	2969	24.56	24.74	3020	3025	25.17	25.21
Slow	5104	8188	4.25	6.82	5344	9161	4.45	7.63

11.2 Memory

Memory required by KMS embedded firmware accounts for both proprietary, preprogrammed code marked as execute-only and compiled reference project code. Usage of memory by each is described below.

11.2.1 Execute-only code

The execute-only portion of KMS firmware is located in the upper-most 8192 bytes on any KMS-enabled KV3 MCU. Based on the memory size of the selected MCU, this code block resides at different addresses, described in Table 64.

Table 64. Execute-only locations for different MCUs

MCU part number	Execute-only memory address
MKV31F512xxx12P	0x7E000
MKV31F256xxx12P	0x3E000
MKV31F128xxx12P	0x1E000
MKV30F128xxx10P	0x1E000
MKV30F64xxx10P	0x0E000

NOTE

The execute-only address is captured in the linker file with the symbol LST_SECURE_START. This symbol is used by the core motor control library so that it can link the unique IP code provided in the execute only region. Do not remove this symbol from the linker file: doing so prevents proper linking of the KMS reference project.

11.2.2 Reference project code

The tables in this section describe program (Flash) and data (RAM) memory sizes in bytes for each element of KMS reference project code by:

- control type
- development environment

Table 65. Memory usage (bytes) - sensorless velocity

IDE	IAR		KDS	
Component	**Flash size**	**RAM size**	**Flash size**	**RAM size**
Motor control	23580	0	27691	0
COM protocol*	8289	1046	9116	405
SPI interface**	684	1	1524	1
KSDK libraries	7846	140	14805	120
Total	**53441**	**6198**	**61048**	**6084**

Table 66. Memory usage (bytes) - sensored velocity

IDE	IAR		KDS	
Component	**Flash size**	**RAM size**	**Flash size**	**RAM size**
Motor control	23209	0	27836	0
COM protocol*	8289	1046	9116	405
SPI interface**	684	1	1524	1
KSDK libraries	7920	140	15107	120
Total	**53209**	**6066**	**61096**	**5956**

Table 67. Memory usage (bytes) - sensored position

IDE	IAR		KDS	
Component	**Flash size**	**RAM size**	**Flash size**	**RAM size**
Motor control	28729	0	34738	0
COM protocol*	8289	1046	9116	405
SPI interface**	684	1	1524	1
KSDK libraries	7920	140	15091	120
Total	**58559**	**6598**	**68424**	**6488**

*Does not consider COM Protocol data buffer.

**Only used if development platform has SPI interface.

11.3 Peripherals and pins

The KMS reference project is set up to use the resources on the enumerated development modules as described in Table 68. This configuration may be modified by the user in customization of the reference project.

Table 68. Development module configuration

Development module	Peripheral	Pins	Function
FRDM-KV31F	FAC	• N/A	Secures execute-only code
	ADC0	• PTB0: ADC0_SE8 - Phase A current • PTB3: ADC0_SE13 - DC Bus voltage	Phase A shunt current and DC Bus voltage sensing
	ADC1	• ADC1_DP0 - Phase B current • ADC1_DP3 - Phase C current	Phase B and Phase C shunt current sensing
	PDB0	• N/A	Trigger ADC reading based on FlexTimer0 Initialization Trigger
	FlexTimer0	• PTC1: FTM0_CH0 - Phase A, High Side • PTC2: FTM0_CH1 - Phase A, Low Side • PTC5: FTM0_CH2 - Phase B, High Side • PTC4: FTM0_CH3 - Phase B, Low Side • PTD4: FTM0_CH4 - Phase C, High Side • PTD5: FTM0_CH5 - Phase C, Low Side	PWM output to motor drive
	UART0	• PTB16: UART0_RX - Receive • PTB17: UART0_TX - Transmit	PC communication
	SPI0	• PTE16: SPI0_PCS0 - Chip Select • PTE17: SPI0_SCK - Clock • PTE18: SPI0_SOUT - Data Out • PTE19: SPI0_SIN - Data In	Motor drive IC configuration
	FlexTimer1	• PTA12: FTM1_CH0 - Encoder Phase A • PTA13: FTM1_CH1 - Encoder Phase B	Quadrature encoder decode (see note below)

Table 68. Development module configuration

<table>
<tr><th>Development module</th><th>Peripheral</th><th>Pins</th><th>Function</th></tr>
<tr><td rowspan="8">TWR-KV31F120M</td><td>FAC</td><td>• N/A</td><td>Secures execute-only code</td></tr>
<tr><td>ADC0</td><td>• PTB0: ADC0_SE8 - Phase A current
• PTB3: ADC0_SE13 - DC Bus voltage</td><td>Phase A shunt current and DC Bus voltage sensing</td></tr>
<tr><td>ADC1</td><td>• ADC1_DP0 - Phase B current
• ADC1_DP3 - Phase C current</td><td>Phase B and Phase C shunt current sensing</td></tr>
<tr><td>PDB0</td><td>• N/A</td><td>Trigger ADC reading based on FlexTimer0 Initialization Trigger</td></tr>
<tr><td>FlexTimer0</td><td>• PTC1: FTM0_CH0 - Phase A, High Side
• PTC2: FTM0_CH1 - Phase A, Low Side
• PTC5: FTM0_CH2 - Phase B, High Side
• PTC4: FTM0_CH3 - Phase B, Low Side
• PTD4: FTM0_CH4 - Phase C, High Side
• PTD5: FTM0_CH5 - Phase C, Low Side</td><td>PWM output to motor drive</td></tr>
<tr><td>UART0</td><td>• PTB16: UART0_RX - Receive
• PTB17: UART0_TX - Transmit</td><td>PC communication</td></tr>
<tr><td>SPI0</td><td>• PTE16: SPI0_PCS0 - Chip Select
• PTE17: SPI0_SCK - Clock
• PTE18: SPI0_SOUT - Data Out
• PTE19: SPI0_SIN - Data In</td><td>Motor drive IC configuration</td></tr>
<tr><td>FlexTimer1</td><td>• PTA12: FTM1_CH0 - Encoder Phase A
• PTA13: FTM1_CH1 - Encoder Phase B</td><td>Quadrature encoder decode (see note below)</td></tr>
<tr><td rowspan="7">HVP-KV31F120M</td><td>FAC</td><td>• N/A</td><td>Secures execute-only code</td></tr>
<tr><td>ADC0</td><td>• PTB0: ADC0_SE8 - Phase A current
• PTB2: ADC0_SE12 - DC Bus voltage</td><td>Phase A shunt current and DC Bus voltage sensing</td></tr>
<tr><td>ADC1</td><td>• ADC1_DP0 - Phase B current
• ADC1_DP3 - Phase C current
• PTC11: ADC1_SE7B - IPM Temperature</td><td>Phase B and Phase C shunt current and IPM Temperature sensing</td></tr>
<tr><td>PDB0</td><td>• N/A</td><td>Trigger ADC reading based on FlexTimer0 Initialization Trigger</td></tr>
<tr><td>FlexTimer0</td><td>• PTC1: FTM0_CH0 - Phase A, High Side
• PTC2: FTM0_CH1 - Phase A, Low Side
• PTC5: FTM0_CH2 - Phase B, High Side
• PTC4: FTM0_CH3 - Phase B, Low Side
• PTD4: FTM0_CH4 - Phase C, High Side
• PTD5: FTM0_CH5 - Phase C, Low Side
• PTA19: FTM0_FLT0 - Fault Input</td><td>PWM output to motor drive</td></tr>
<tr><td>UART0</td><td>• PTB16: UART0_RX - Receive
• PTB17: UART0_TX - Transmit</td><td>PC communication</td></tr>
<tr><td>FlexTimer1</td><td>• PTA12: FTM1_CH0 - Encoder Phase A
• PTA13: FTM1_CH1 - Encoder Phase B</td><td>Quadrature encoder decode (see note below)</td></tr>
</table>

NOTE

FlexTimer1 is used only in reference projects requiring sensored control (sensored velocity or sensored position).

12 Frequently asked questions

12.1 Why are Automatic Parameter Measurement and System Inertia Measurement separate steps?

— Automatic Parameter Measurement is seeking out just the motor's electrical characteristics, so should be performed without anything to the shaft, while System Inertia Measurement deals with the mechanical, so should be coupled to the application's mechanical setup. The separation of measurement is intended to reinforce this distinction.

12.2 What happens if my motor fails Automatic Parameter Measurement?

— Apart from invalid input of Basic Motor Information, this is most likely to occur when using sensored control and having a faulty encoder setup. Refer to the Kinetis Motor Suite Lab Guide for instructions on adding an incremental encoder to your system to supported sensored position/velocity feedback.

12.3 What if I can't complete System Inertia Measurement?

— System Inertia Measurement from Motor Tuner attempts to identify your system inertia with a default configuration, then adjusts this configuration in the event of failure to complete (Figure 201). KMS notifies and requires the user to explicitly retry after each incomplete measurement attempt because if the motor is in application (e.g., garage door), repeated automatic attempts to accelerate and decelerate may not be desirable.

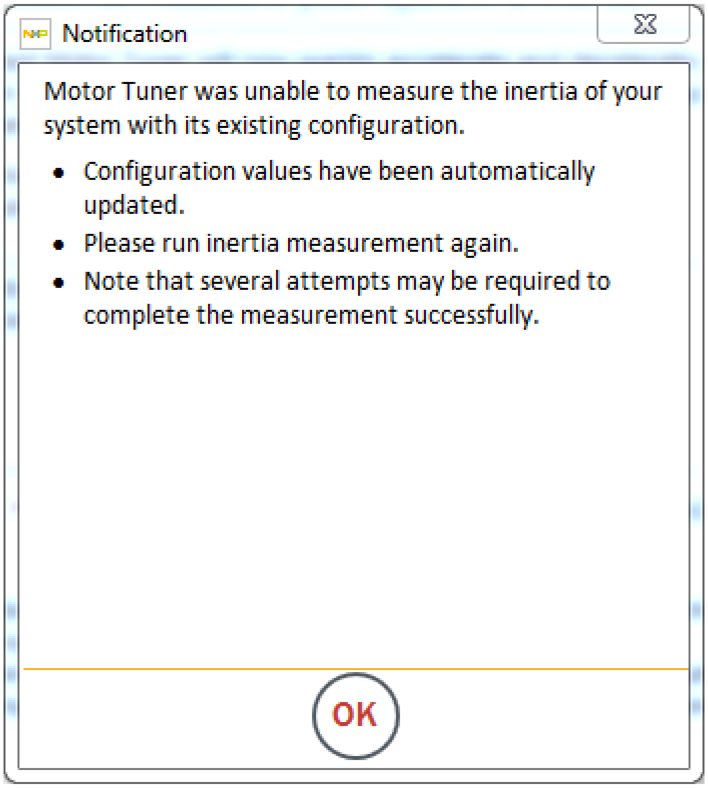


Figure 201. Incomplete inertia measurement

— Only after Motor Tuner has exhausted its adaptive configuration options does it declare an inability to complete the measurement (Figure 202).

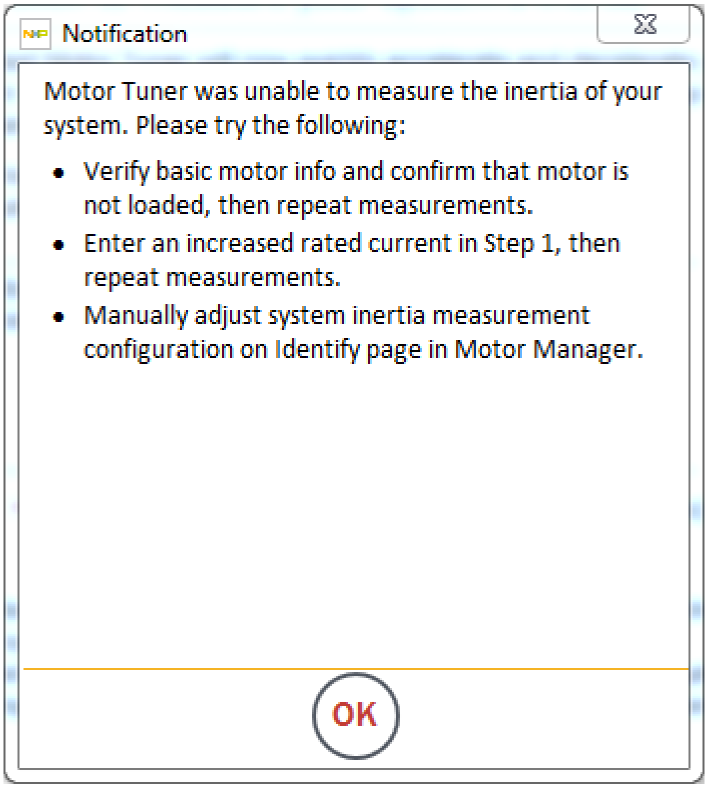


Figure 202. Failed inertia measurement

In the case of the former, incomplete measurement, simply click OK and try again. It is not uncommon for multiple attempts to be required due to the wide array of possible inertial systems.

In the case of failed measurement, utilize the error recommendations in Table 60 as a starting point for adjusting the inertia measurement configuration parameters manually, then iterating. You may also observe the adjustments that KMS automatically makes in response to see trend of useful adjustments and parameters. You may also plot the measurement in an attempt to change configuration parameters to match the signal of a valid measurement (Figure 203).

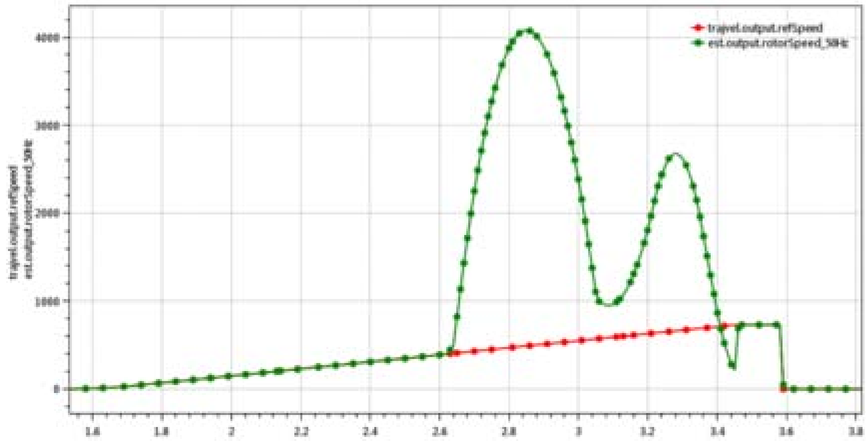


Figure 203. Valid inertia measurement

To see what your inertia measurement looks like, simply open the SpeedFeedback plot and begin sampling before starting inertia measurement.

12.4 Can I manually enter motor parameters?

— Yes, the Motor Manager Advanced Tuning page allows for this. This is not advised unless the user is a motor control expert.

12.5 What happens if my motor cannot run to rated speed?

— Your motor may require advanced tuning. Key parameters to try adjusting include:

- Percent of Rated Current used at startup and Speed Threshold (sensorless operation)
- Speed and Position Regulator Bandwidth
- Current Regulator Bandwidth.

12.6 What happens if my motor starts spinning in the wrong direction?

— The firmware flags a fault if the Target Speed and Actual Speed are mismatched for a certain period of time. This causes the motor to stop and it may need to be restarted again.

12.7 What can I do if I have mass erased my device?

— The device must be replaced for any future work with KMS. KMS requires the existence of the execute-only firmware block to run, both from the GUI as well as from within the IDE. Contact your support personnel for assistance.

12.8 How do I provide information to support personnel to help resolve my issue?

— The KMS support time capsule, accessed via the View menu or when the GUI is forced to shut down due to error, aggregates the files described in Table 69.

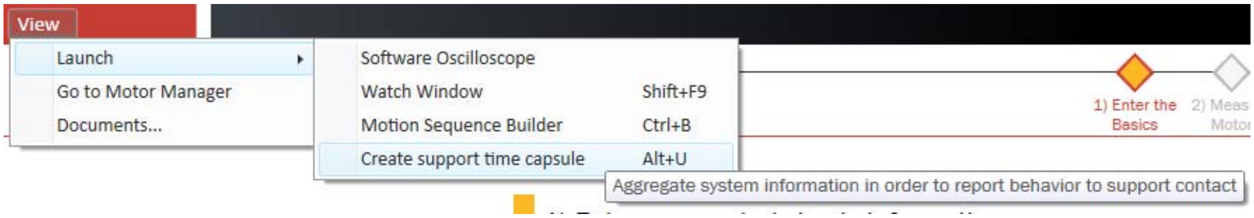


Figure 204. Create support time capsule from View menu

Table 69. Support time capsule files

File	Purpose
Current .kms file	Defines current state of GUI values. Used by support personnel to try to view GUI as user sees it.
Current trace log	Describes actions that have taken place in GUI and publishes any exception information.
system.h	Describes key motor characteristics so the full system may be considered, not just the GUI.
GeneralInfo.txt	Self-reported information from the user regarding contact information and steps to reproduce.

— After the support time capsule functionality is activated, a form appears to take the information that ultimately constitutes the GeneralInfo.txt file.

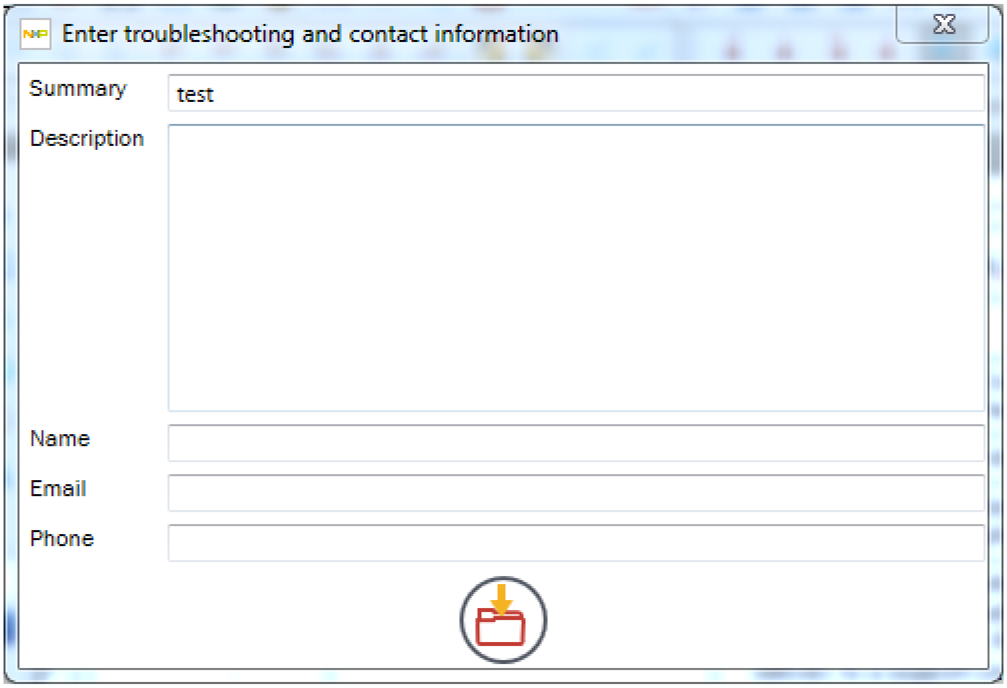


Figure 205. Support time capsule form

— Upon creating the time capsule by clicking the button at bottom, a notification informs the user of the location where the relevant files have been zipped and place.

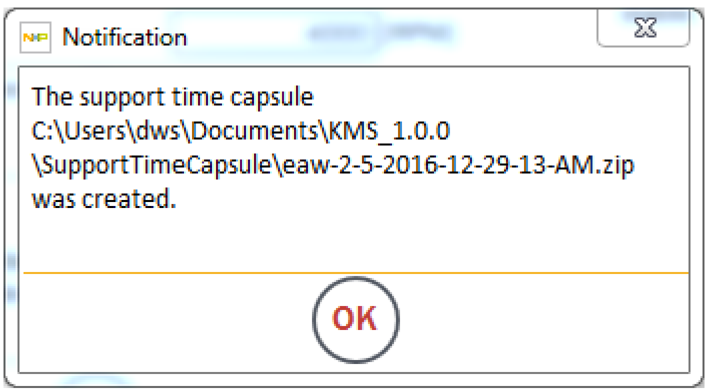


Figure 206. Time capsule notification

— Email this zip file to your support contact.

13 GUI elements glossary

Table 70. Selected KMS GUI elements

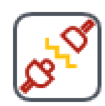
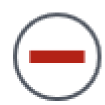
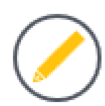
Name	Type	Location(s)	Overview	Image
Add	Button	• Software Oscilloscope • Watch Window	• Add a Plot • Add a Variable	
Communication Status	Indicator	Status Icon Bar (bottom middle)	Indicates whether board is communicating to PC: • Green = online • Red = offline	 
Connect/Disconnect	Button	Activation Bar (bottom right)	Connects or disconnects communication from board to PC. Right clicking on this button displays a menu of further communication options, including configuration.	 
Delete	Button	• Software Oscilloscope • Watch Window	• Delete a Plot • Delete a Variable	
Edit	Button	• Software Oscilloscope • Watch Window	• Edit a Plot • Edit a Variable	

Table 70. Selected KMS GUI elements

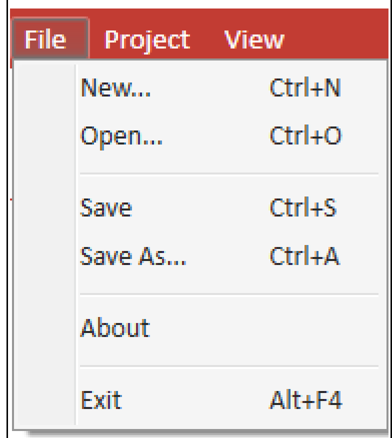
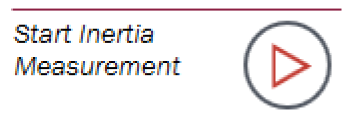
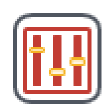
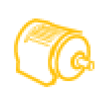
Name	Type	Location(s)	Overview	Image
File menu	Menu	Upper left hand corner of KMS	Allows users to manage KMS files	
Inertia measurement	Button	Identify page	A specific instance of the Run button. Begins the automatic inertia identification process, during which the motor is accelerated & decelerated. The values for inertia and friction update.	
Manage files	Menu	Motion Sequence Builder	Allows users to manage Motion Sequence Builder files	
Motor Manager (control panel)	Button	Activation Bar (bottom right)	Click to transition from Motor Tuner to Motor Manager	
Motor Manager pages	Menu	Top right hand corner of Motor Manager	Navigate to the desired page	
Motor Status	Indicator	Status Icon Bar (bottom middle)	Indicates the state of the motor based on firmware error codes: • Green = good • Yellow = unknown • Red = fault	 

Table 70. Selected KMS GUI elements

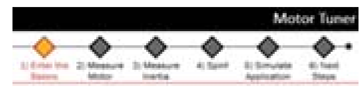
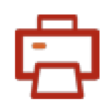
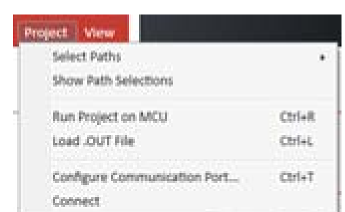
Name	Type	Location(s)	Overview	Image
Motor Tuner (wizard)	Button	Activation Bar (bottom right)	Click to transition from Motor Manager to Motor Tuner	
Motor Tuner Steps	Menu	Top right hand corner of Motor Tuner	Navigate to the desired step	
New	Menu Item	• Landing page • File menu	Creates a new Kinetis Motor Suite project.	
No	Button	Notification Window (pop-up)	Rejects the option proposed in a pop-up window.	
OK	Button	Notification Window	Accepts the information/option indicated in a pop-up window.	
Open	Menu Item	• Landing page • File menu	Opens an existing Kinetis motor suite project.	
Print Source File	Button	Motion Sequence Builder Source Code Page	Prints the selected source code file	
Project Menu	Menu	Upper left hand corner of KMS	Allows users to manage the key connection points to embedded firmware operation	

Table 70. Selected KMS GUI elements

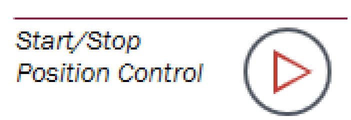
Name	Type	Location(s)	Overview	Image
Restart	Button	Select system configuration	Restarts the process of defining desired KMS reference project in terms of platform, control type, IDE, etc.	
Run	Button	Various	A generic button style that initiates actions ranging from starting Automatic Parameter Identification to commencing Data Sampling. Toggles to Stop after the relevant action has been initiated.	
Run on Target	Button	Motion Sequence Builder Source Code Page	Launches a background process that compiles and builds the motion sequence source code, and downloads the code to the MCU	
Save	• Button • Menu Item	• Pop-up Windows • Main Menu	• Saves the relevant configuration (e.g., Communication Settings) • Saves the Kinetis Motor Suite project	 
Save As CSV	Button	Software Oscilloscope	Saves plot information as a comma separated value file for assessment in Excel or other spreadsheet application.	
Save As PNG	Button	Software Oscilloscope	Saves plot information as an image.	
Start/Stop Position Control	Button	• Position Control • Dashboard	A specific instance of the Run button. Changes the internal operating state to start and stop position control.	

Table 70. Selected KMS GUI elements

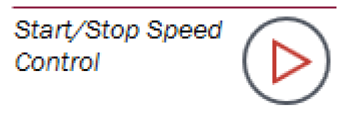
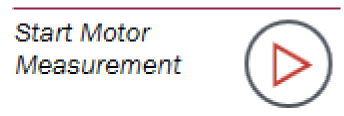
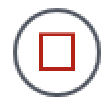
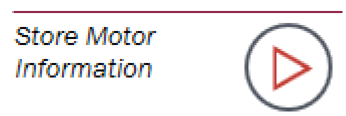
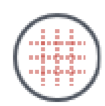
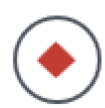
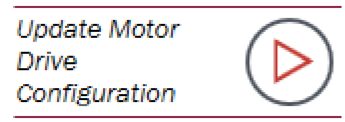
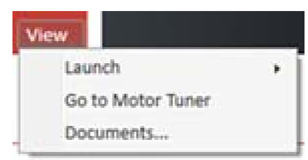
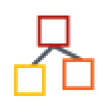
Name	Type	Location(s)	Overview	Image
Start/Stop Speed Control	Button	• Speed Control • Dashboard	A specific instance of the Run button. Changes the internal MCU Drive State Machine to start and stop speed control.	
Start Motor measurement	Button	Expert Identify page	A specific instance of the Run button. Changes the internal MCU Motor Self-Commissioning module to start automatic motor parameter identification.	
Stop	Button	Various	A generic button style that stops actions ranging from starting Automatic Parameter Identification to commencing Data Sampling in plot. Toggles to Start after the relevant action has been stopped.	
Store Motor Information	Button	Various	Generates a header file describing motor control configuration for assessment and manual adjustment.	
Toggle Grid	Button	Software Oscilloscope	Toggles on and off background gridlines in the Software Oscilloscope.	
Toggle Line	Button	Software Oscilloscope	Toggles on and off lines between data points in the Plot Window.	
Toggle Marker	Button	Software Oscilloscope	Toggles on and off markers for the data points in the Software Oscilloscope.	

Table 70. Selected KMS GUI elements

Name	Type	Location(s)	Overview	Image
Update Motor Drive Configuration	Button	• Protection and Hardware page • Advanced Tuning page	A specific instance of the Run button. Recalculates the motor drive tuning parameters. Loads new parameters into the RAM of the MCU.	
View .c File	Button	Motion Sequence Builder Source Code Page	View the motion sequence .c file generated by Motion Sequence Builder	
View .h File	Button	Motion Sequence Builder Source Code Page	View the motion sequence header file generated by Motion Sequence Builder	
View Menu	Menu	Upper left hand corner of KMS	Allows users to activate the Watch Window or Motion Sequence Builder	
View Software Oscilloscope	Button	Activation Bar (bottom right)	Activates a Software Oscilloscope Plot of desired motor characteristics as a function of time	
View Source Code	Button	Motion Sequence Builder	View the source code generated by Motion Sequence Builder	
View State Diagram	Button	Motion Sequence Builder	Launches image viewer to display diagram of configured motion sequence	
View Watch Window	• Button • Menu Item	• Activation Bar (bottom right) • View Menu	Activates Watch Window for access to system variables.	

14 References

Key documents referred to in this User's Guide are enumerated in Table 71.

Table 71. Key companion documents

Document	Content
KMS API Reference Manual	Description of embedded motor control software architecture and modules
KMS Lab Guide	Hands-on instructions intended to introduce the user to both the KMS GUI and KMS embedded motor control software
KMS Release Notes	Definition of KMS release contents, system requirements, and software updates
Adapting KMS for Custom Hardware	Step-by-step explanation to move from a development platform to the end-application hardware
Kinetis V series documentation	Typical hardware-focused documentation provided in support of Kinetis MCUs

15 Revision history

Table 72 provides a revision history for this document.

Table 72. Document revision history

Rev. number	Date	Substantive change(s)
0	02/2016	Initial release

How to Reach Us:

Home Page:
freescale.com/kinetismotorsuite

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Document Number: KMS100UG
Rev. 0
02/2016