Document Number: PEXMVHBRIDGESWUG
Rev. 1.0, 3/2016

Medium-voltage H-Bridge driver Processor Expert software component

Contents

1 Overview

This documentation describes how to install and use Processor Expert in conjunction with the component. The medium-voltage H-Bridge component is a software driver capable of controlling both DC brushed and stepper motors. The component provides an API layer between the hardware and the user application. It makes application development less time consuming by offering an easy-to-use interface for setting motor control options.

The medium-voltage H-Bridge component supports the following analog parts:

- NXP MC33926: Single H-Bridge, brushed DC motor driver, 5.0 V to 28 V, 5.0 A, 20 kHz
- NXP MC33931: Single H-Bridge, brushed DC motor driver, 5.0 V to 28 V, 5.0 A, 11 kHz
- NXP MC33932: Dual H-Bridge, stepper/brushed DC motor driver, 5.0 V to 28 V, 5.0 A, 11 kHz
- NXP MC34931(S): Single H-Bridge, brushed DC motor driver, 5.0 V to 36 V, 5.0 A, 11 kHz/20 kHz
- NXP MC34932(S): Dual H-Bridge, stepper/brushed DC motor driver, 5.0 V to 36 V, 5.0 A, 11 kHz/20 kHz

NXP offers following board solutions based on these chips:

- TWR-MC-MVHB1EVB (for MC33926 and MC33932)
- FRDM-33931-EVB
- FRDM-34931-EVB
- FRDM-34931S-EVB

For detailed descriptions, see the related hardware user guides and datasheets.

2 Medium-voltage H-Bridge compatibility

This chapter describes the MCU peripherals and resources required to control analog devices. Supported analog devices are enumerated and briefly introduced. Supported MCUs are listed along with feasible settings for analog boards.

2.1 Peripheral requirements

Peripherals and resource requirements critical to the MCU's ability to handle a given part are as follows:

- **TPM/FTM** timer (PWM, two to four channels) is required for direct input control (IN1 to IN4).
- **GPIOs** are required for the device enable pins (D1, EN/D2) and external interrupt (SF - Status Flag).
- **AD convertor** (1 channel) is required for feedback measurement.

2.2 Supported devices

This section describes which models of analog devices are supported by this component and the main differences and capabilities of named devices.

The **MC33926**, **MC33931** and **MC33932** are monolithic H-Bridge Power ICs designed primarily for automotive electronic throttle control, but are applicable to any low-voltage DC servo motor control application within the current and voltage limits stated in the specification.

The **MC34931(S)** and **MC34932(S)** are monolithic H-Bridge Power ICs designed primarily for DC brushed and servo motor driver applications within the current and voltage limits stated in the specification.

Common Freedom board features:

- Compatibility with Freedom series evaluation boards such as the FRDM-KL25Z
- Built in fuses for both part and load protection
- Screw terminals to provide easy connection to power and loads
- Test points for probing signals
- Built in voltage regulator to supply logic level circuitry
- LED indicators for key board functions (such as the status of the Logic power supply)

Common device features:

- 3.0 V and 5.0 V TTL/CMOS logic compatible inputs
- Each H-bridge is able to control inductive loads with currents up to 5.0 A (peak)
- Automatic thermal back-off, peak current limiting (6.5 A ±1.5 A) ensures continuous high availability operation
- Drain-to-Source on resistance $R_{DS(on)}$ typ. 120 mΩ
- Real-time load current monitoring (analog feedback current mirror)
- Sleep mode with low-current draw

Table 1. Device features

Device model	No. of H-Bridges	Operating voltage	Maximum PWM frequency
MC33926	Single	8.0 V to 28 V	20 kHz
MC33931	Single	8.0 V to 28 V	11 kHz
MC33932	Dual	8.0 V to 28 V	11 kHz
MC34931	Single	5.0 V to 36 V	11 kHz
MC34931S	Single	5.0 V to 36 V	20 kHz
MC34932	Dual	5.0 V to 36 V	11 kHz
MC34932S	Dual	5.0 V to 36 V	20 kHz

2.3 Supported MCUs

Target MCUs are a subset of the MCUs supported by Processor Expert for Kinetis using a logical device drivers (LDD) layer. MCUs supported by the example projects with the MVHBridge Processor Expert component are in Table 2.

Table 2. Supported MCUs

Supported MCUs	CW examples	KDS examples
TWR-K20D72M	YES	YES
TWR-K64F120M	YES	NO
TWR-K70F120M	YES	NO
TWR-KL25Z48M	YES	NO
FRDM-KL25Z	YES	NO

Compatibility with other MCUs and MCU boards depends on their peripherals and pin connections.

2.4 Feasible Freedom/Tower boards settings

2.4.1 Freedom Evaluation boards

The FRDM-33931S-EVB/FRDM-34931S-EVB/FRDM-34931-EVB consist of an H-Bridge, a parallel interface, power conditioning circuitry and a set of two input select jumpers. All +5.0 V V_{DD} power required by the board is obtained via the parallel interface.

Caution:

To avoid damaging the board, observe the following restrictions:

- The motor supply voltage (V_{PWR}) must be at least 5.0 V, but must not exceed 40 V.
- The peak operating current of the load must not exceed 5.0 A.

The safe configuration routine is as follows:

1. Connect the FRDM-34931S-EVB or FRDM-34931-EVB to the FRDM-KL25Z.
2. With the power switched off, attach the DC power supply to the VPWR and GND screw connector terminals on the evaluation board.
3. Attach one set of coils of the brushed motor to the OUT 1 and OUT 2 screw connector terminals on the evaluation board.

Figure 1 illustrates the hardware configuration.

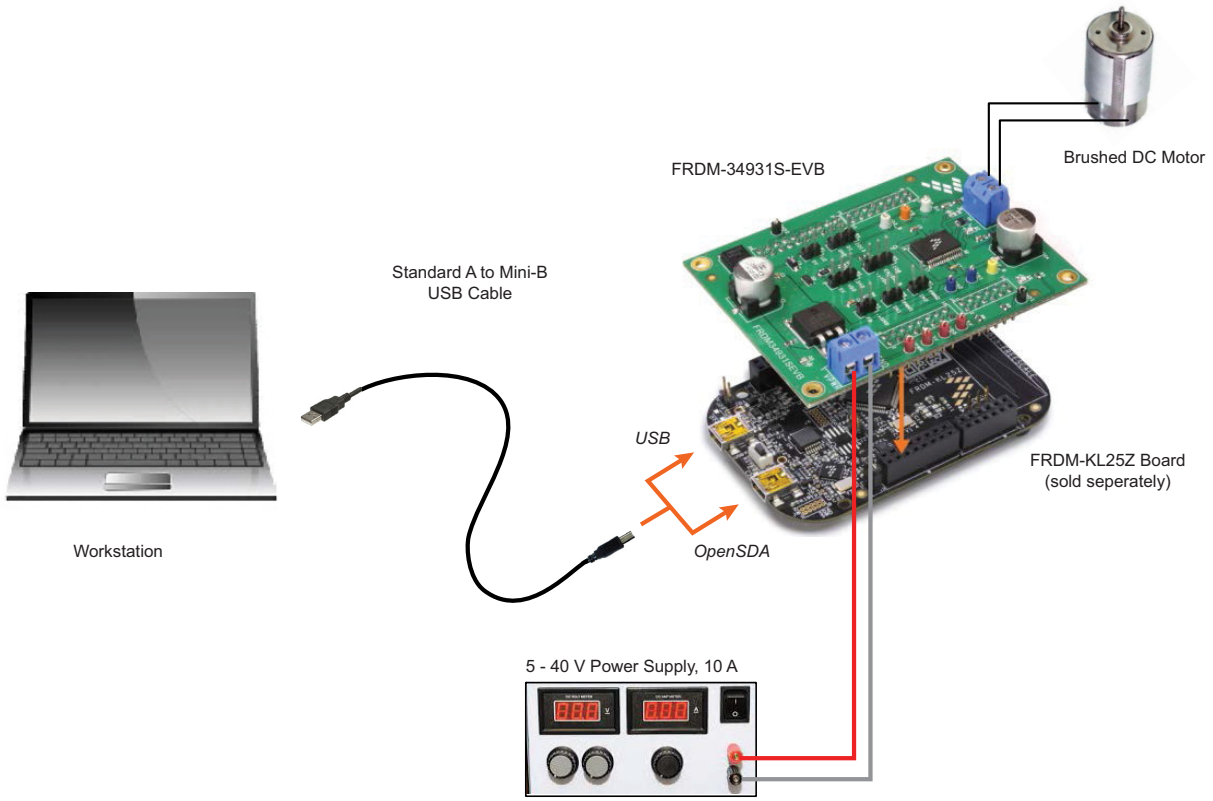

Figure 1. Freedom board configuration

2.4.2 Tower peripheral module

When configured as a tower platform module, the TWR-MC-MVHB1EVB must be used in conjunction with another tower MCU evaluation board. The following procedure describes how to set up the hardware when the TWR-MC-MVHB1EVB is configured to drive three DC brushed motors.

1. Assemble the tower platform by sliding the TWR-MC-MVHB1EVB elevator connectors into the top slots on the tower elevator modules. Insert the tower MCU evaluation board in the tower elevator modules in a set of slots below the TWR-MC-MVHB1EVB.
2. Connect the USB cable between the PC and the USB port on the tower MCU evaluation board.
3. Connect the MC33932 channel A load to connector J1 (OUT2 & OUT1) and the MC33932 channel B load to connector J2 (OUT4 and OUT3) on the TWR-MC-MVHB1EVB board. These two loads could be either two DC motors or one stepper motor.
4. Connect the MC33932 8.0 V to 28 V DC power supply to connector J3 (PWR-IN) on the evaluation board.
5. Connect the MC33926 load to connector J4 (OUT2 & OUT1) on the TWR-MC-MVHB1EVB board.
6. Connect the MC33926 8.0 V to 28 V DC power supply to connector J5 (PWR-IN) on the evaluation board.

Figure 2 illustrates the procedure.

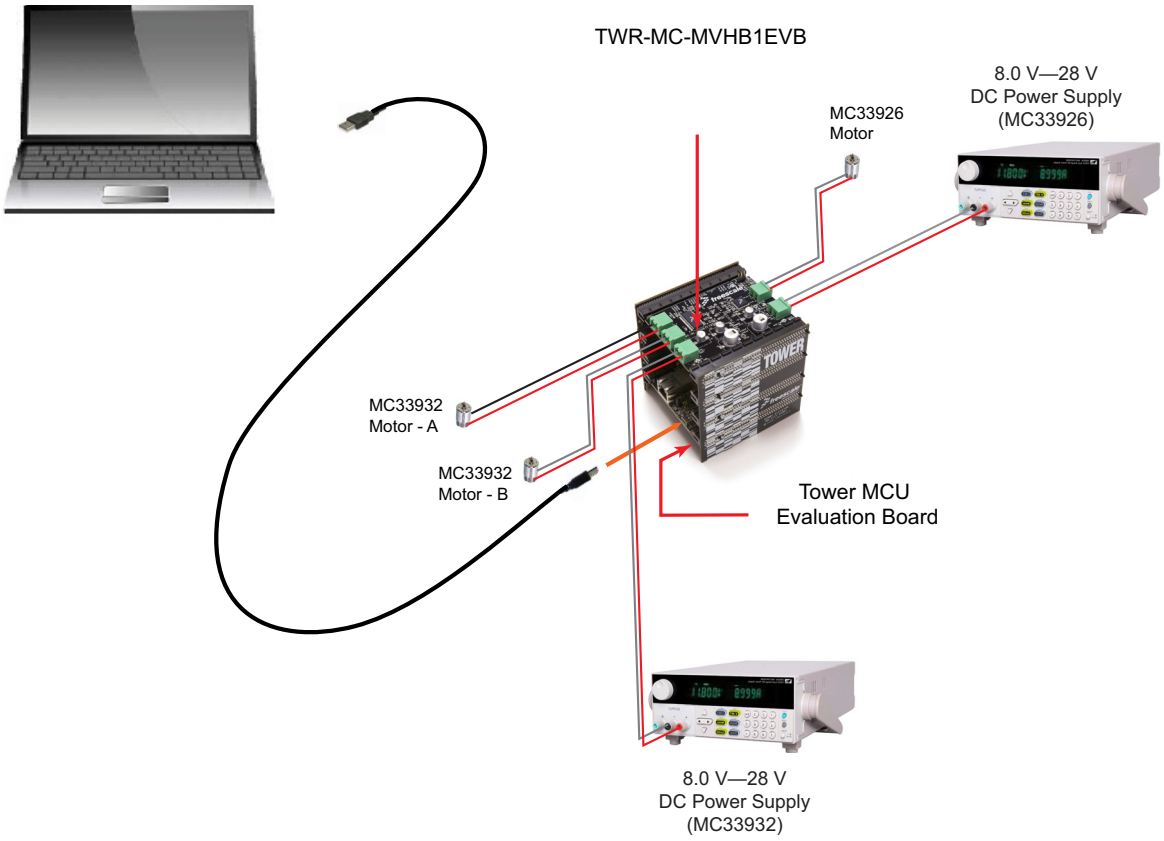

Figure 2. Tower system configuration

3 MVHBridge component

This chapter explains functionality, settings and use of the MVHBridge component (properties and methods). The user has access to this component through a component tree of an active project. Figure 3 shows MVHBridge exclusively inherited and referenced components. Functionality of the MVHBridge depends on these components for control, etc.

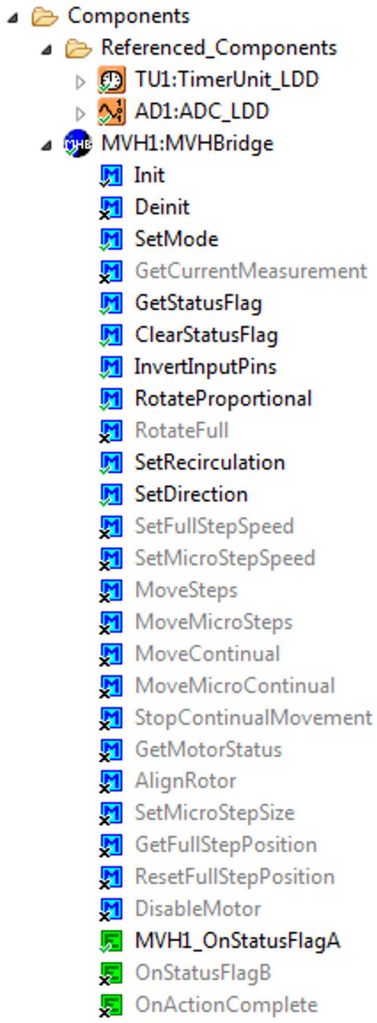

Figure 3. MVHBridge processor expert component

There is also available generated documentation of this component, accessible through **Help on Component**, when you right-click on MVHBridge in the component tree. All properties and methods of the component are described. Examples show how to work with API methods in the **Typical Usage** section.

3.1 Component settings

Selecting the MVHBridge component in the component tree gains access to properties in the **Component Inspector**, see Figure 4. These are used for general settings of the component and determine behavior after initialization (which can be changed later by provided API). Note that not all of the settings can be changed later in code.
Explanation of component properties follows.

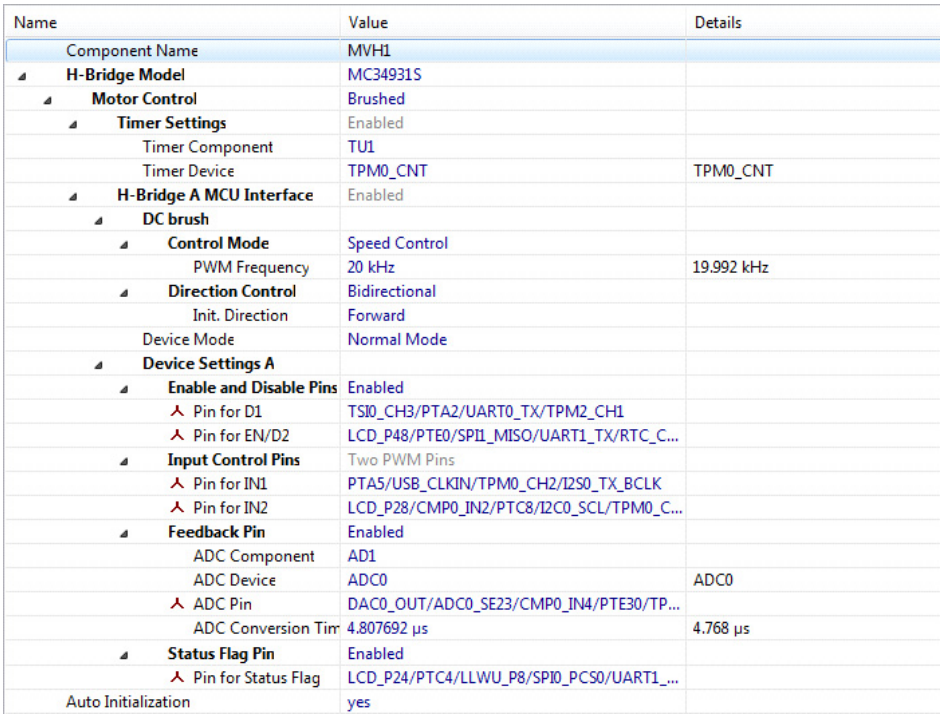

Figure 4. MVHBridge component properties

The **H-Bridge Model** is the top node in the tree structure. A drop-down menu in the **Value** column selects the H-Bridge model the project uses. The **Motor Control** group is directly below the **H-Bridge Model** node. The group contains two child nodes: **Timer Setting** and **H-Bridge A MCU Interface**. An MCU with dual H-Bridges would have an **H-Bridge B MCU Interface** group with settings similar to H-Bridge A. The settings in each of these groups are detailed by the following.

Timer Settings, when enabled, define timer settings for the project. The group contains the following settings:

- Timer Component defines the name of the linked **TimerUnit_LDD Component**.
- Timer Device defines the name of the hardware timer being used.

H-Bridge A MCU Interface defines H-Bridge interface settings. The group contains three child nodes: **DC Brush**, **Device Mode** and **Device Settings A**.

DC Brush selects the motor control mode and the motor direction:

Control Mode selects whether the settings control the motor speed (**Speed Control**) or the motor is controlled by GPIO pin signals (**State Control**).

- **PWM Frequency** sets the pulse width modulation frequency.

Direction Control determines in which direction the motor is allowed to rotate. Forward means the motor can rotate only in the clockwise direction. Reverse allows movement only in the counterclockwise direction. Bidirectional allows the motor to rotate in either direction.

- **Init Direction** determines which direction (forward or reverse) the motor moves at startup.

Device Mode defines the H-Bridge operational mode for the selected device. The mode specifics depend on the device, but Normal, Sleep and Standby are typical. For more information, see the data sheet for the device. Device Mode is controlled by the **Enabling and Disabling Pins property**. The mode can be changed in C code using the **SetMode** method.

Device Settings A associates each of the output pins with a corresponding input pin name.

Enable and Disable Pins defines settings that control the **Device Mode** property. The number and the names of pins in this group depend on the H-Bridge model selected. In all cases, assign the appropriate value to each pin name in the group.

Slew Rate enables the selection of fast or slow slew rate.

- **Pin for Slew** defines the component setting associated with the device's SLEW pin.

Input Invert sets the device's INV pin, subsequently causing IN1 and IN2 to be set to LOW = TRUE (see the device's data sheet for additional information on the Input Invert (INV) pin).

- **Pin for INV** defines the component setting associated with the device's INV pin.

Input Control Pins settings define H-Bridge outputs. These pins are controlled by timer channels or by GPIO pins, according to other settings in the component.

Feedback Pin settings define current measurements on the feedback pin. H-Bridge feedback provides ground-referenced 0.24% of the high-side output current.

- **ADC Component** sets the name of the linked ADC_LDD component.
- **ADC Device** defines the device used for current measurement.
- **ADC Pin** defines the pin used for ADC current sensing.
- **ADC Conversion Time** specifies the time interval in micro-seconds allowed for a single analog to digital conversion.

Status Flag Pin tracks the H-Bridge status flag. Method **GetStatusFlag** provides current device status. Method **ClearStatusFlag** clears the status flag. Use Event **OnStatusFlagA** or **OnStatusFlagB** (depending on the H-Bridge interface) to handle errors indicated by the status flag.

Auto Initialization selects whether component initialization should be automatically called from CPU component initialization function PE_low_level_init() or the user is responsible for calling initialization method.

3.2 Setting up a project to control a DC Brushed motor

1. Select the H-Bridge model desired to configure and set the **Motor Control** property to **Brushed**.

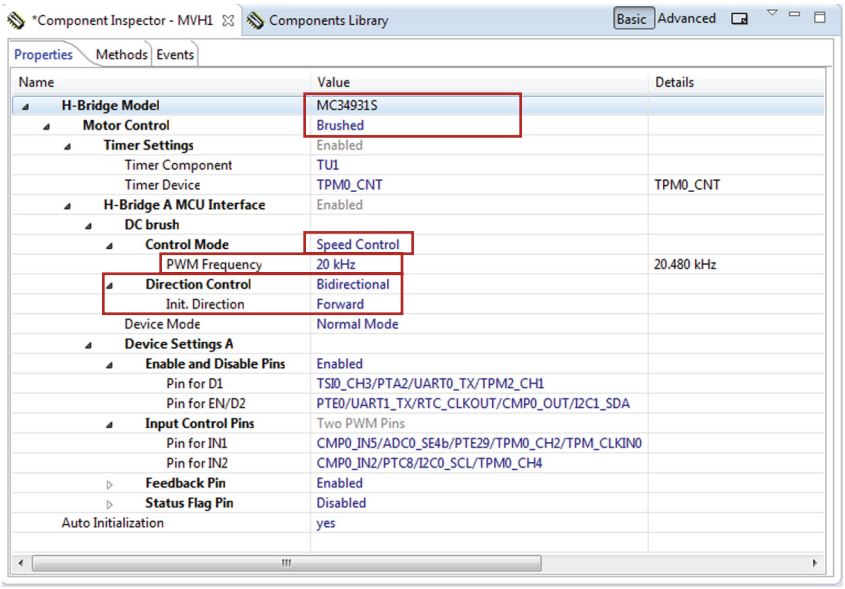

Figure 5. Brushed motor control setup

2. Set the **Control Mode** property. There are two ways to control the DC brushed motor:
   - **Speed Control** - motor speed is controlled by the settings. The **TimerUnit_LDD** component generates the PWM signal. The **PWM Frequency** property is visible in this mode only. If setting the **Speed Control** mode on both interfaces (i.e. Interface A and Interface B), the **PWM Frequency** property on Interface B is set automatically to the same value as Interface A (because Interface B uses the same timer.)
   - **State Control** - motor is controlled by GPIO pins (**BitIO_LDD** components). This means the motor can be switched on or off without speed adjustments. The advantage of this mode is that no timer channels are needed. When setting the **State Control** on both interfaces or only on a single H-Bridge model (one interface) with **State Control**, the **TimerUnit_LDD** component is not required by the MVHBridge component and can be removed from the project.
3. Set the **PWM Frequency**.

4. Set the **Direction Control** property. The **Direction Control** property determines what direction the motor is allowed to move in. Setting the property **Forward** restricts the motor's movement to the forward direction only. Setting the property **Reverse** restricts movement to the reverse direction only. A **Bidirectional** setting allows the motor to move in either direction. The Bidirectional mode requires two timer channels. Forward or Reverse requires only one timer channel and one GPIO port. This setting is available only when the **Speed Control** mode is set in the **Control Mode** property.

3.3 Setting up a Project to control a stepper motor

The following application notes apply to stepper motor control:

- The MVHBridge component was tested with a core clock frequency ranging from 20 MHz (minimal value) to 120 MHz.
- Do not change the settings of the timer device (**TimerUnit_LDD**) linked by the MVHBridge component. The component sets the timer device automatically.
- The acceleration and deceleration ramp of the stepper motor is computed in real-time using integer arithmetic. This solution is based on the article “Generate stepper-motor speed profiles in real time” (Austin, David. 2005).
- The stepper motor holds its position (coils are powered) after motor movement is completed. Use the **DisableMotor** method to set H-Bridge outputs to LOW (coils not powered).
- The FTM or TPM timer device is needed by stepper control logic.
- The **AlignRotor** method affects the position of the motor. This method executes four full-steps and is only available when full-step mode is enabled.

To set up the project for stepper motor:

1. Select the dual H-Bridge model to configure and set **Stepper** in the **Motor Control** property. Use a dual H-Bridge model because a two phase bipolar stepper motor has four inputs.
2. In the **Stepper Motor** group, set the properties which apply to your environment.

Output Control property defines the control method for the H-Bridge outputs.

- With **PWM** selected, the component uses four channels of a timer to control the stepper motor. Signals are generated in hardware and micro-step mode is available.
- With **GPIO** selected, GPIO pins instead of timer channels control the motor and only full-step mode is available. Micro-step mode is not available. This mode consumes more MCU clock cycles, because all signals are generated in interrupts using four GPIO pins.

Manual Timer setting (only visible when **Advanced** is selected) is designed to change the **Counter frequency** property of the linked **TimerUnit_LDD** component. By default, **Counter frequency** is set automatically by the MVHBridge component. In some cases, the frequency value does not have to be set appropriately (for example, when you want to set a different value or when an error has occurred).

Motor Control mode selects the step mode. The allowable values are **Full-step**, **Micro-step and Full-step and Micro-step**. Selecting Full-step and Micro-step switches between full-stepping and micro-stepping in C code.

- The **Full-step Configuration** group contains speed and acceleration settings. Code for the acceleration and deceleration ramp is generated when the Acceleration property is set to a value greater than zero. Note that acceleration is always the same as deceleration. An example of an acceleration ramp is depicted in Figure 6.

  Initial Speed is set to 100 full-steps per second. This value can be changed in C code.

  Acceleration Ramp defines the ramp speed for both acceleration and deceleration. In this example, the value is set to 400 full-steps per second2. Note that the motor reaches the speed in 0.25 seconds (desired_speed/acceleration = 100 / 400 = 0.25).

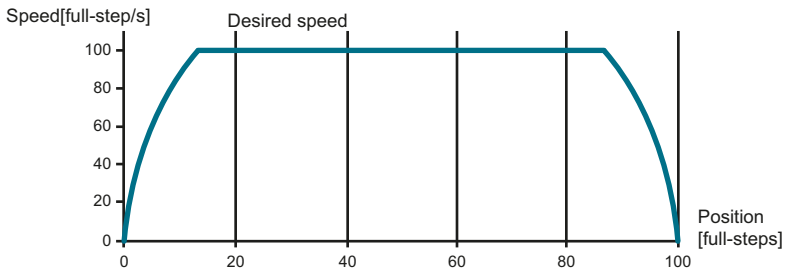

Figure 6. Acceleration and deceleration ramp

- **Micro-step Configuration** group settings are similar to those of the Full-step Configuration. PWM Frequency is the frequency of the micro-step PWM signal. Micro-step per Step is the number of micro-steps per one full-step.

3.3.1 Stepper motor speed

This defines the stepper motor's minimum and maximum speed. These limit values are used by various component methods. Minimum and maximum speeds depend on the MVHBridge component settings. Counter input frequency does not influence micro-stepping spend, but could affect full-stepping speed.

Table 3. Minimum and maximum stepper motor speed

Mode description	MVHBridge component properties			Stepper motor speed (steps per second)	
	Timer device	Output control	Motor control mode	Minimum speed (steps/sec.)	Maximum speed (steps/sec.)
Full-step mode	FTM	PWM	Full-step	Depends on timer input frequency	11,000
Micro-step mode	FTM or TPM	PWM	Micro-step	1	5,000
Full-step mode (using TPM timer)	TPM	PWM	Full-step	1	11,000
Full-step mode (SW control)	FTM or TPM	GPIO	Full-step	1	11,000

Possible values for the timer input frequency (Counter frequency property in TimerUnit_LDD) are in Table 4. Input frequency values depend on MVHBridge component settings. Note that, in one case, two frequency values are needed in Full-step and Micro-step mode when FTM timer is used (the MVHBridge component switches in runtime between these two values).

Table 4. Minimum and maximum stepper motor speed

Mode description	MVHBridge component properties			Stepper motor speed (steps per second)		
	Timer device	Output control	Motor control mode	Number of values	Minimum	Maximum
Full-step mode	FTM or TPM	PWM	Full-step	1	131 kHz	1.0 MHz
Micro-step mode	FTM or TPM	PWM	Micro-step	1	1.2 MHz	10 MHz
Full-step and micro-step mode	FTM	PWM	Full-step and micro-step	2	1st value for Full-step: 131 kHz	1st value for Full-step: 1.0 MHz
					2nd value for Micro-step: 1.2 MHz	2nd value for Micro-step: 10 MHz
Full-step and micro-step mode (using TPM timer)	TPM	PWM	Full-step and micro-step	1	1.2 MHz	10 MHz
Full-step mode (SW control)	FTM or TPM	GPIO	Full-step	1	131 kHz	1.0 MHz

3.3.2 Computation of minimum full-stepping speed

The minimum full-stepping speed depends on the timer input frequency only when the Timer Device is set to FTM (FTM0_CNT, or FTM1_CNT, etc.) and the Output Control is set to PWM. The Full-step signal is generated by a timer while channels toggle on compare (Figure 7).

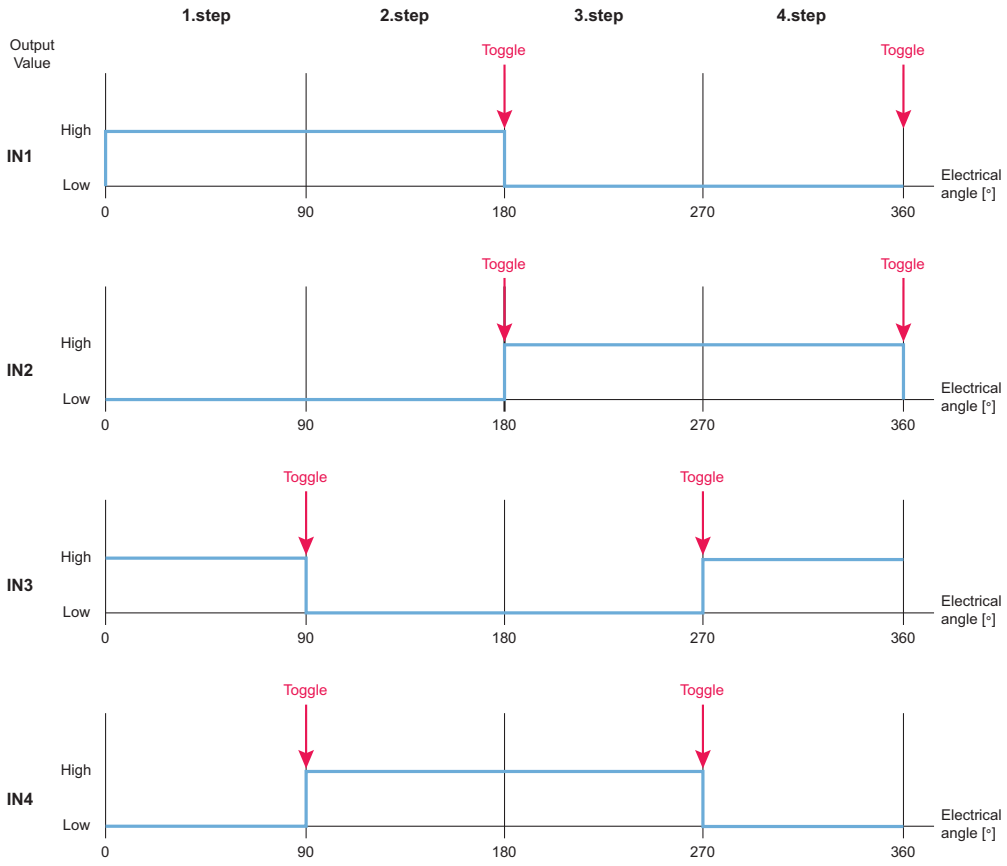

Figure 7. Generating the full-step control signal

The Full-step minimum speed is derived from the input frequency of the timer device (the **Counter frequency** property of the **TimerUnit_LDD** component being used). Find minimum values for speed in the MVHBridge header file (see constant <component_name>_MIN_FULLSTEP_ SPEED). The formula for calculation of this value is as follows:

$$\text{Speed}_{\text{min}} = \frac{2 \times \text{Counterfrequency}}{65536} + 1$$

- Counter frequency = input frequency of the timer device in Hz
- 65536 = maximum value of TimerUnit_LDD counter (16-bit counter).
- Adding 1 ensures that the 16-bit counter does not overflow (which is the point of the formula).

For example if the Counter frequency is set to 187500 Hz, the minimum speed is:

$$\text{Speed}_{\text{min}} = \frac{2 \times \text{Counterfrequency}}{65536} + 1 = \frac{2 \times 18750}{65536} + 1 = 6.72$$

The MCU rounds the value down, so the result is 6 full-steps per second.

3.3.3 Setting the minimum full-stepping speed

This section describes how to change the input frequency of the **TimerUnit_LDD** component.

1. Launch Processor Expert and select the MVHBridge component.
2. In the Processor Expert menu bar, set component visibility to **Advanced**.
3. In the Properties tab, find the **Manual Timer** setting property and set the value to **Enabled**. If this property is not there, make sure the component visibility is set to **Advanced**.

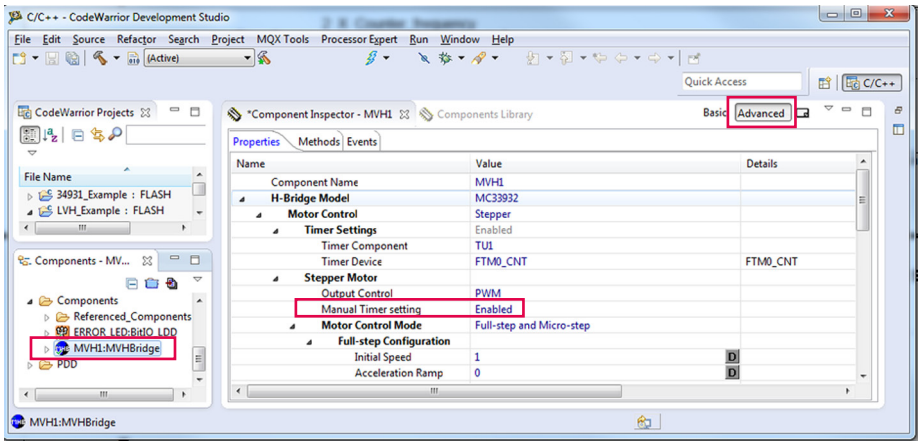

Figure 8. Enabling the manual frequency setting

4. In the Components panel, expand the **Referenced_Components** folder and double-click on the component **TU1:TimerUnit_LDD**.
5. Click the **Counter frequency** value field. An ellipsis button appears at the right edge of the cell.
6. Click on the button.

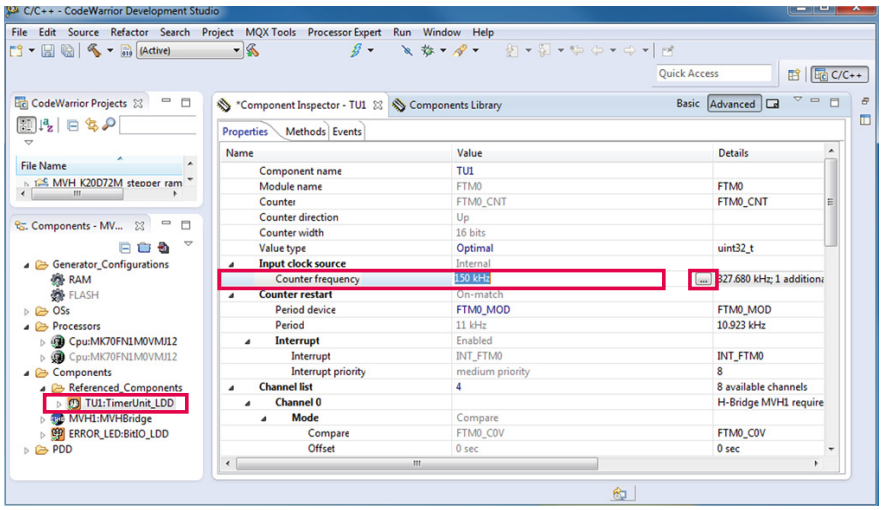

Figure 9. Component TimerUnit_LDD timing dialog

7. Set the frequency value (163.84 kHz in illustration). The list of available frequencies depends on the CPU component settings (with an external crystal as the clock source and a core clock of 48 MHz).
8. Set the **Allowed Error** value at 10%.
9. Click **OK** to complete the process.

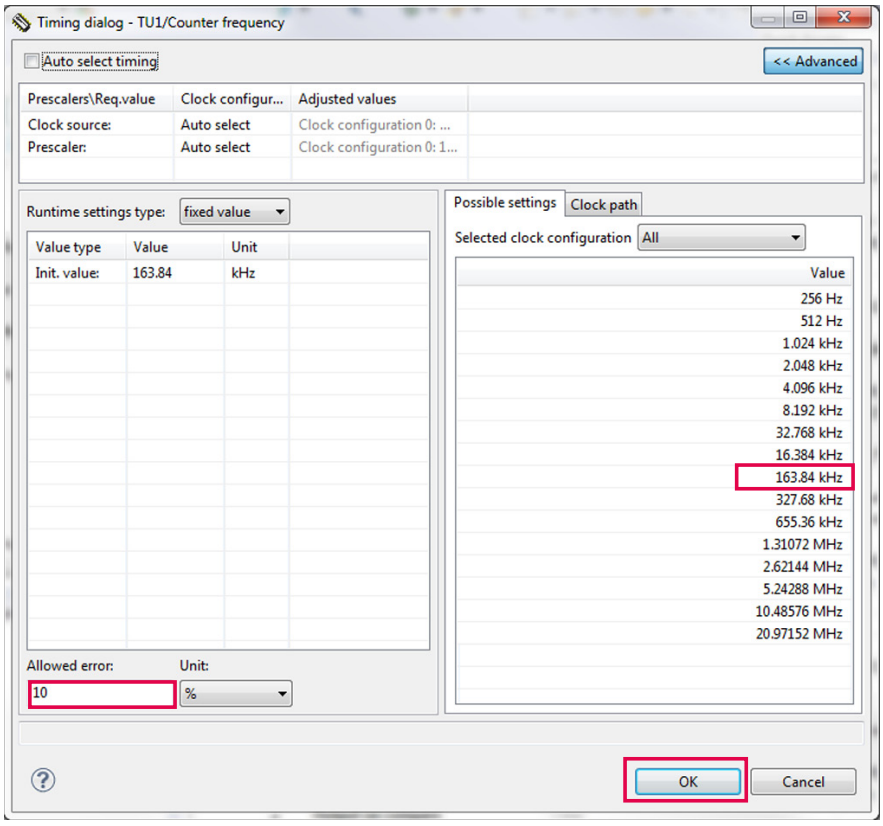

Figure 10. Component TimerUnit_LDD timing dialog-select input frequency

3.4 Component API

MVHBridge component provides API, which can be used for dynamic real-time configuration of the device in user code. Available methods and events are listed under the component selection, see Figure 3.

Some of those methods/events are marked with ticks and other ones with crosses, it distinguishes which methods/events are supposed to be generated. This setting can be changed in the Processor Expert Inspector. Note that methods with grey text and tick are always generated, because they are needed for proper functionality. This forced behavior depends on various combinations of settings of component properties.

Table 5 summarizes the available API methods, events and their descriptions.

Table 5. MVHBridge component API

Method	Description
Init	This initializes the device.
Deinit	This method deinitializes the device.
SetMode	This method sets H-Bridge device mode using enable and disable pins.
GetCurrentMeasurement	This method is intended for measuring of H-Bridge output current.
GetStatusFlag	This method returns fault status of H-Bridge device.
ClearStatusFlag	This method clears fault flag of H-Bridge device by toggling of D1 pin.
InvertInputPins	This method inverts output value of IN pins. The method is available only when Input Invert setting is enabled and MC33926 H-Bridge model is used.
RotateProportional	This method starts rotation of brush motor. The method allows control of motor speed.
RotateFull	This method starts rotation of brush motor. The method is intended for state motor control (on/off).
SetRecirculation	This method sets recirculation of brush motor.
SetDirection	This method sets direction of brush motor.
SetFullStepSpeed	Set speed of full-step mode. Unit is number of full-steps per second.
SetMicroStepSpeed	Set speed of micro-step mode. Unit is number of micro-steps per second.
MoveSteps	This method moves the motor by specified number of full-steps.
MoveMicroSteps	Moves motor by specified number of micro-steps.
MoveContinual	Moves motor continually in full-step mode.
MoveMicroContinual	This method moves motor continually in micro-step mode.
StopContinualMovement	This method is intended to stop continual movement of stepper motor.
GetMotorStatus	This method returns status of stepper motor control.
AlignRotor	Align the rotor to full-step position. The method executes 4 full-steps forward (one electrical revolution) at minimum speed.
SetMicroStepSize	This method serves to change size of micro-step.
GetFullStepPosition	This method returns current full-step position.
ResetFullStepPosition	This method sets counter of full-steps to zero.
DisableMotor	The method sets IN pins output value to LOW. The method can be used to stop the stepper motor.
OnStatusFlagA	This event is called when status flag signal goes to LOW on Bridge A.
OnStatusFlagB	This event is called when status flag signal goes to LOW on Bridge B.
OnActionComplete	This event is called when the motor reaches desired number of steps or the motor is stopped by method StopContinualMovement. The handler is available only for stepper motor.

3.5 Utilized components

Description of inherited and shared components, which are used by the MVHBridge component, see Figure 11. Functionality of MVHBridge depends on these components in terms of control, etc.

Re-use of existing Processor Experts components speeds up, facilitates development process and provides verified implementation solutions for regular tasks.

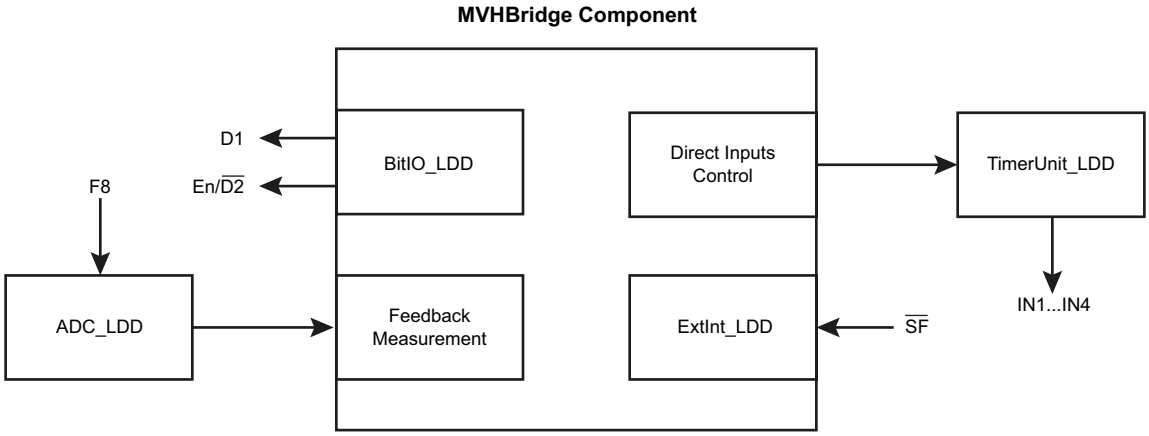

Figure 11. Components used by the MVHBridge component

- Referenced components:
  - TU1:TimerUnit_LDD (input control pins signal generation IN1...4)
  - AD1:ADC_LDD (feedback measurement FB)
- MVH1:MVHBridge
  - BitIO_LDD (hidden, enable pins D1, En/D2)
  - ExtInt_LDD (hidden, status flag pin SF)

4 Installing the Processor Expert software

This chapter describes installation of CodeWarrior and use of Processor Expert for application development. Processor Expert software is available as part of the CodeWarrior Development Studio for Microcontrollers, Kinetis Design Studio, or as an Eclipse-based plug-in for installation into an independent Eclipse environment (Microcontroller Driver Suite). For more information about Processor Expert refer to Processor Expert Software and Embedded Components.

4.1 Installing CodeWarrior

This procedure explains how to obtain and install the latest version of CodeWarrior (version 10.6 in this guide). The procedure for Kinetis Design Studio installation is very similar.

Note:

The component and some examples in the component package are intended for CodeWarrior 10.6 and Kinetis Design Studio 3.0. If CodeWarrior 10.6 or above (or Kinetis Design Studio 3.0 or above) is already on your system, the steps in this section can be skipped.

1. Obtain the latest CodeWarrior installer file from the NXP CodeWarrior website.
2. Run the executable file, follow the on-screen instructions and proceed through the installation.

4.2 Downloading the component and example projects

This section describes how to get MVHBridge component and example projects from website. There are also steps on how to run these examples in CodeWarrior 10.6 or newer.

First download the project and its associated components:

1. Download example projects and MVHBridge component zip from the www.nxp.com\MVHBRIDGE-PEXPERT.
2. Unzip the downloaded file and ensure the folder contains the files listed in Table 6.

Table 6. Content of the downloaded zip file

Folder Name	Folder Contents
CodeWarrior_Examples	Example project folder for CodeWarrior
MVH_K20D72M_brushed	Example project for DC brush motor control
MVH_K20D72M_brushed_FreeMaster	Example project intended for control of brushed motor using FreeMaster tool
MVH_K20D72M_step_FreeMaster	Example project intended for control of stepper motor using FreeMaster tool
MVH_K20D72M_stepper	Example project for stepper motor control using full-stepping and micro-stepping mode
MVH_K20D72M_stepper_fullstep	Example project for stepper motor control demonstrating full-step mode
MVH_K20D72M_stepper_ramp	Example project for stepper motor control demonstrating acceleration and deceleration ramp
MVH_K64F120M_brushed_2component	Example project for DC brush motor control using two H-Bridges (i.e. MC33932 and MC33926)
MVH_K70F120M_brushed	Example project for TWR-K70F120M with DC brushed motor control
MVH_K70F120M_stepper	Example project for TWR-K70F120M with stepper motor control using full-stepping and micro-stepping mode
MVH_KL25Z48M_brushed_2component	Example project for DC brushed motor control using a dual H-Bridge device (e.g. MC33932 and 33926)
MVH_KL25Z48M_fullstep_ramp	Example project for stepper motor control demonstrates acceleration and deceleration ramp in full-step mode
Component	Processor Expert component folder
MVHBridge_Bxxx.PEupd	MVHBridge component update file
DriverSuite_Examples	Example project folder for Driver Suite
MVH_K20D72M_stepper	Example project for stepper motor control uses full-stepping and micro-stepping mode

Table 6. Content of the downloaded zip file (continued)

Folder Name	Folder Contents
KDS_Examples	Example project folder for Kinetis Design Studio
MVH_K20D72M_stepper	Example project for stepper motor control, which uses full-stepping and micro-stepping mode
MVH_K20D72M_stepper_ramp	Example project for stepper motor control demonstrating usage of acceleration and deceleration ramp
FRDM34931SEVB_Examples	Example project folder for CodeWarrior and H-Bridge board FRDM-34931SEVB
MVH_KL25Z_brushed	Example project for DC brush motor control

4.2.1 Importing the MVHBridge component into the Processor Expert library

Follow these instructions to install the component:

1. Launch CodeWarrior and click 'Processor Expert -> Import Component(s)' in the menu. In the pop-up window, locate Component file (.PEupd) in the example project folder: 'MVHBridge_PEx_SW/Component'. Select MVHBridge_bxxxx.PEupd file, then click 'open' (see Figure 12).

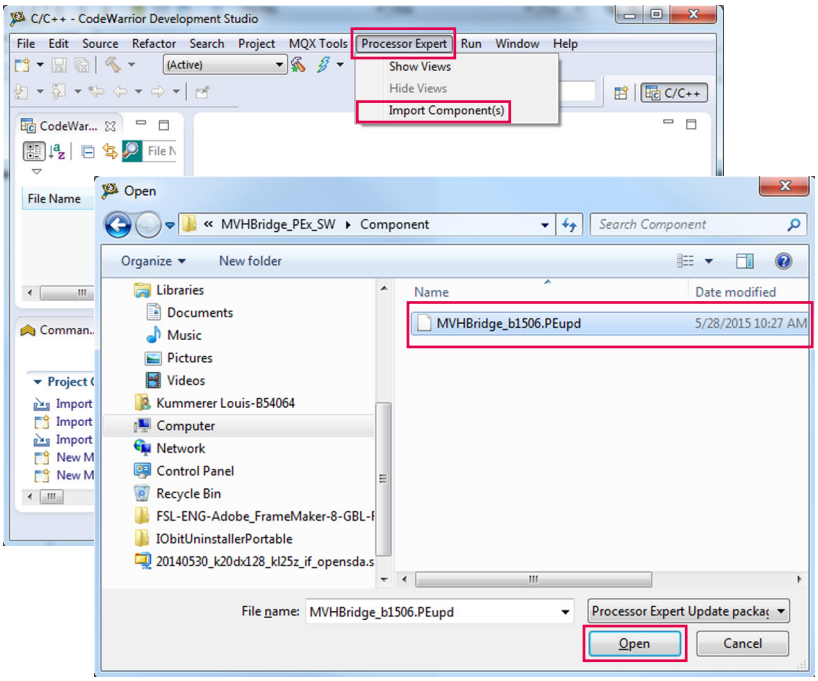

Figure 12. Importing the MVHBridge component

2. If import is successful, then the MVHBridge component is in the 'Components Library -> Your Repository -> SW -> User Components (see Figure 13).

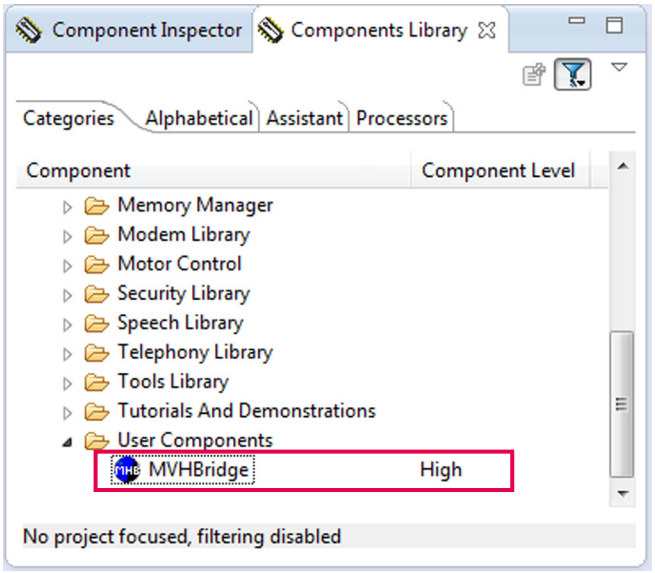

Figure 13. MVHBridge component location after CodeWarrior import

3. The MVHBridge component is now ready to use.

4.2.2 Importing an example project into CodeWarrior

The following steps show how to import an example from the downloaded zip file into CodeWarrior.

1. Click 'File -> Import…' in the in CW menu. In the pop-up window, select 'General -> Existing Projects into Workspace' and click 'Next'.

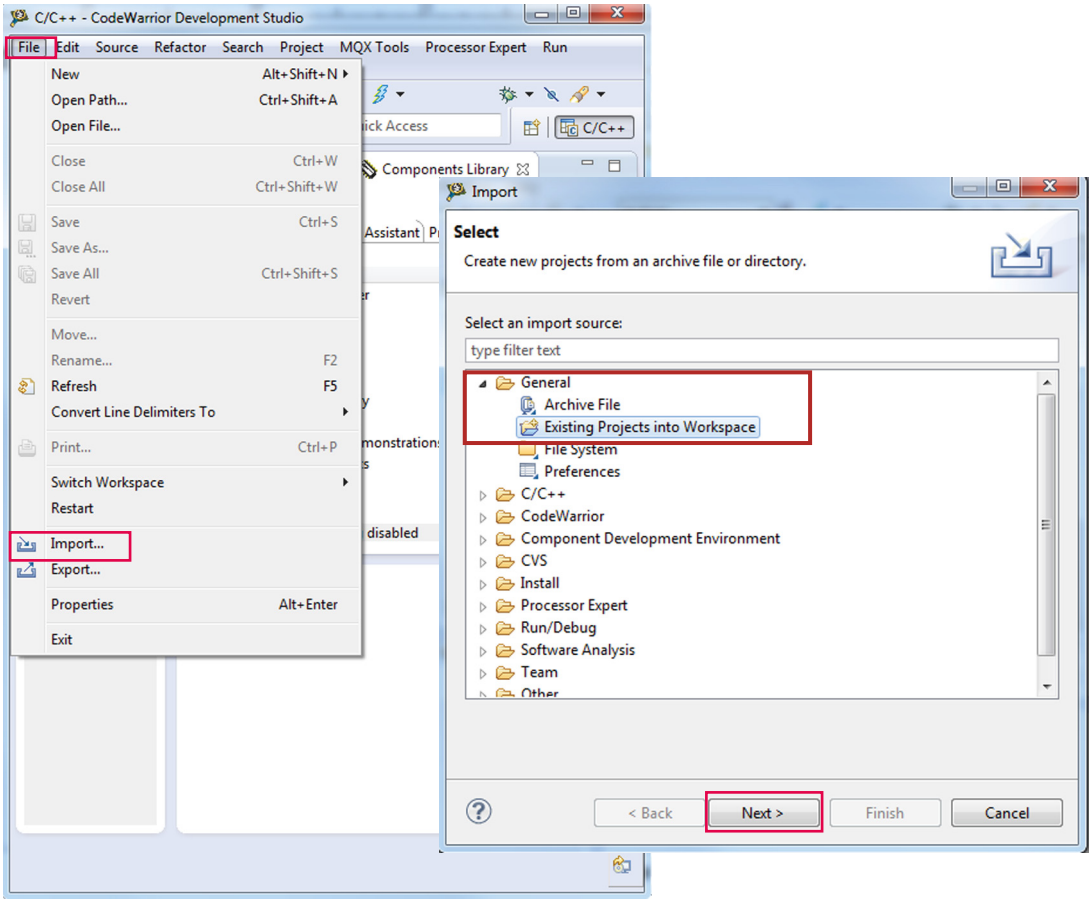

Figure 14. Importing an example project

2. Locate the example in the folder: 'MVHBridge_PEx_SW/CodeWarrior_Examples'. Then click 'Finish'.
3. The project is now in the CW workspace and can be built and run.

4.3 Creating a New Project with Processor Expert and the MVHBridge Component

If the example project is not used, follow these instructions for the creation and setup of a new project with the MVHBridge component.

1. Create new Bareboard Project and name it (see Figure 15).

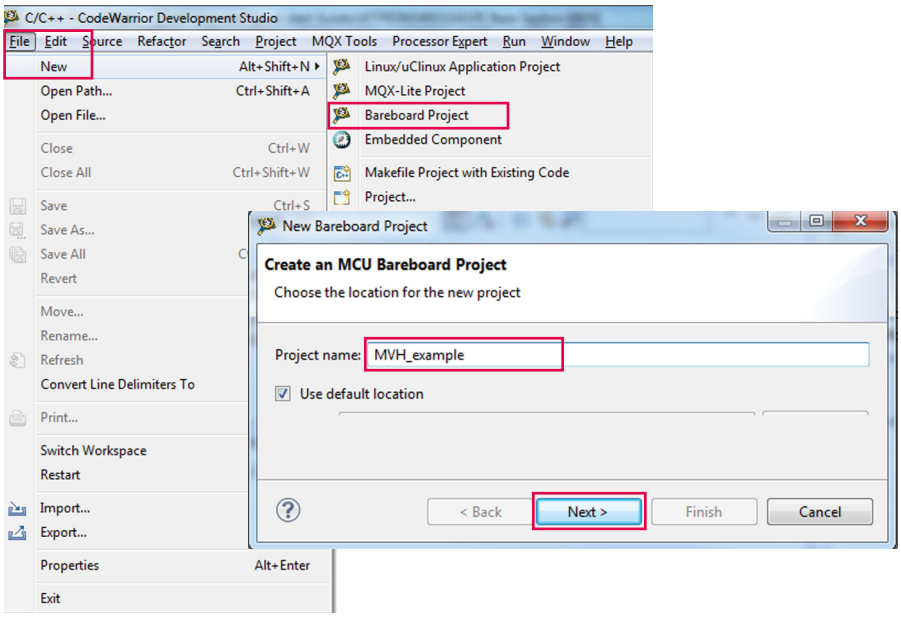

Figure 15. Creating a bareboard project

2. Choose the MCU class to be used in the MCU Freedom board (MK20DX256 in this example). Select **Processor Expert** and click **Finish** (see Figure 16).

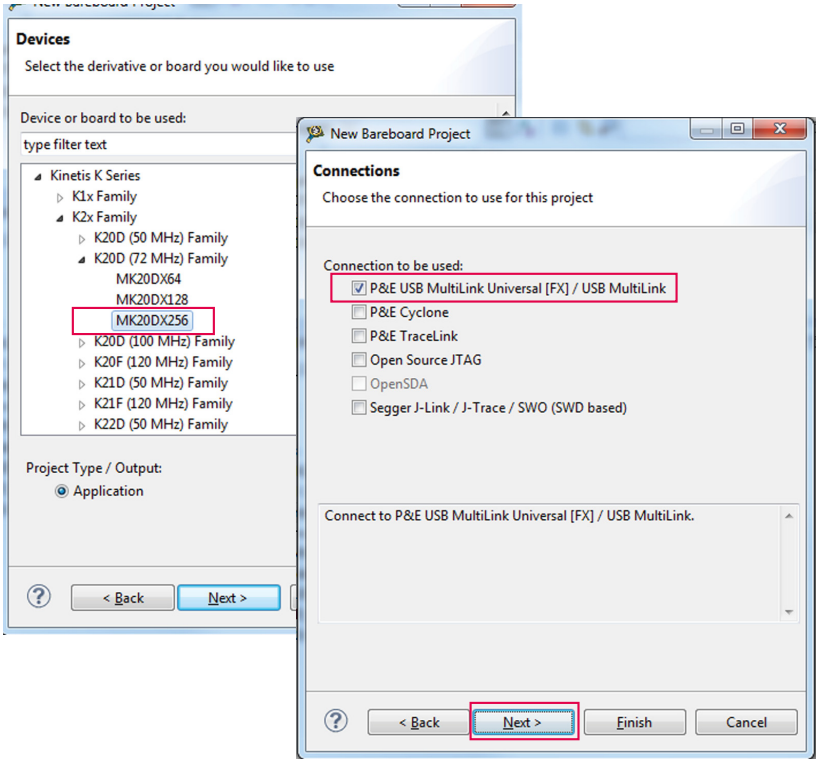

Figure 16. Choosing the Devices and Connections options

4.4 Adding the MVHBridge component into the project

1. Find MVHBridge in the Components Library and add it into the project (see Figure 17).

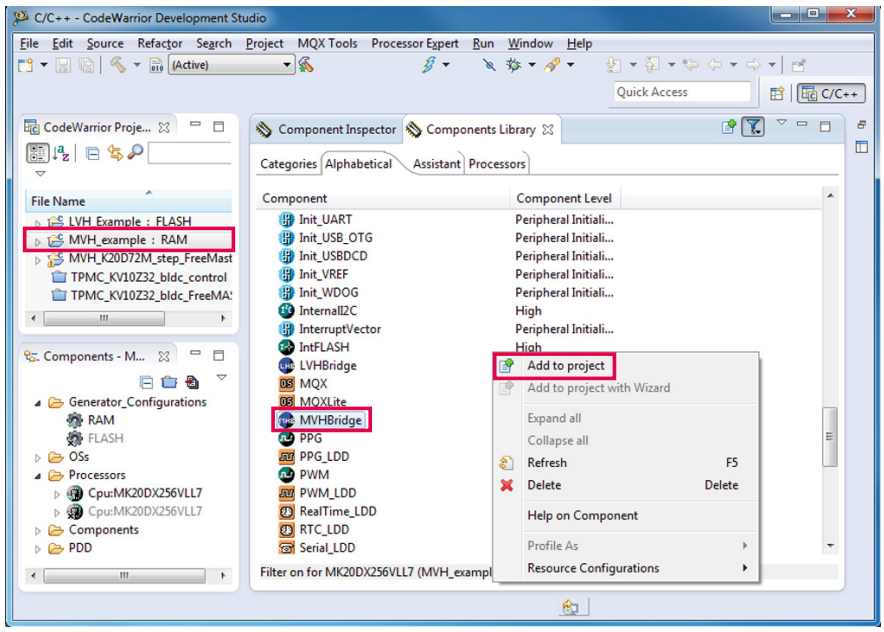

Figure 17. Add MVHBridge component to the project

2. Double click the MVHBridge component (see Figure 18) to show configurations in the Component Inspector view. Set all necessary values according to the application needs.

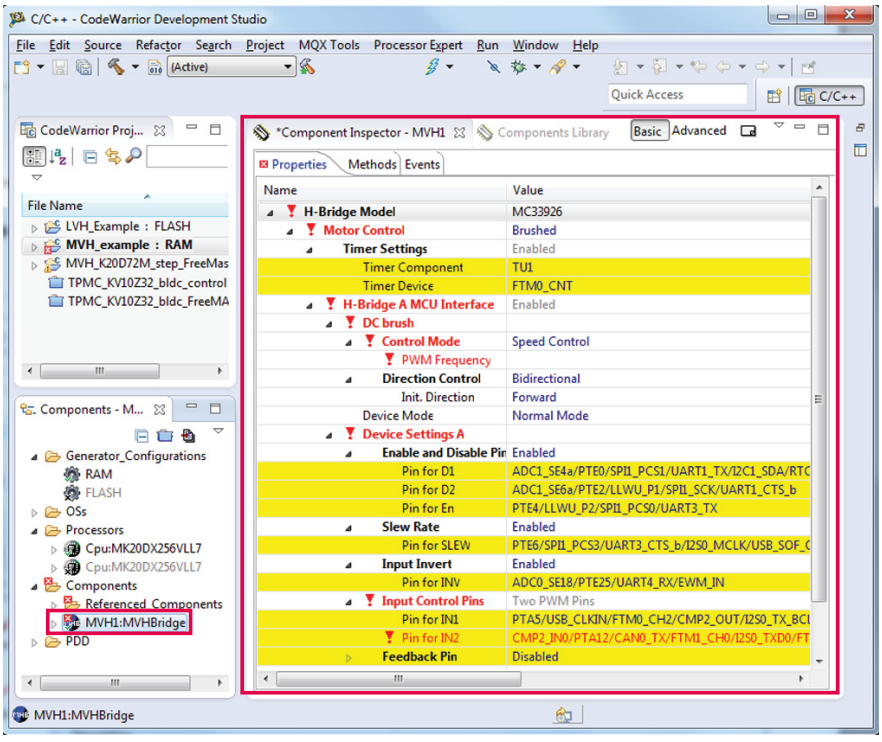

Figure 18. Component inspector view

4.5 Setting up the project

Once the new project is created and the MVHBridge component is added, it must be set up. Section 3.1, Component settings, page 8 describes the capabilities of the component and what is necessary to configure it properly.

4.6 Generating driver source code

After completing the configuration of the components, it is ready to generate the driver code to be incorporated into the application. The process is as follows:

1. Click on the Generate Processor Expert Code icon in the upper right corner of the Components panel (Figure 19).

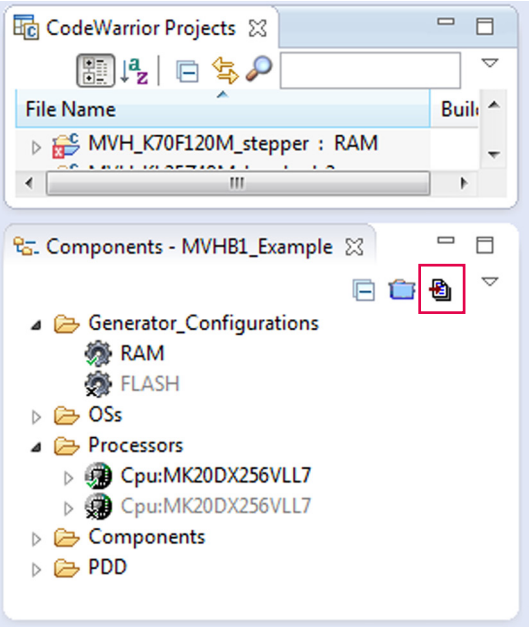

Figure 19. Generating the source code

The driver code for the valve controller is generated into the Generated_Code folder in the Project Explorer panel. The component only generates the driver code. It does not generate application code. Figure 20 shows the locations of the generated driver source and the application code.

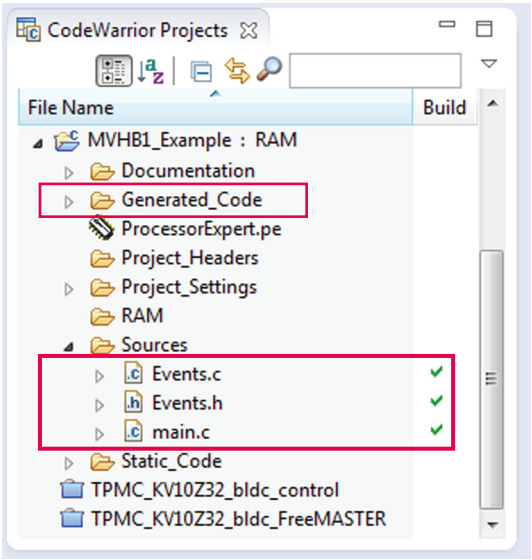

Figure 20. Source code locations

4.7 Writing application code

All application code must reside in the Sources folder in the project directory. The code in main.c and Events.c may be modified, but the original comments related to usage must be retained.

To add a component method into the application source code:

1. In the Components view for the project, click on Components folder and select a component. Find the method to add to the code.
2. Drag and drop the method directly into the source code panel.
3. Add the appropriate parameters to the method. Hovering the mouse over the method displays a list of the required parameters.

For example, open the MVHBridge component method list, drag and drop RotateProportional to main.c and add the necessary parameters (see Figure 21).

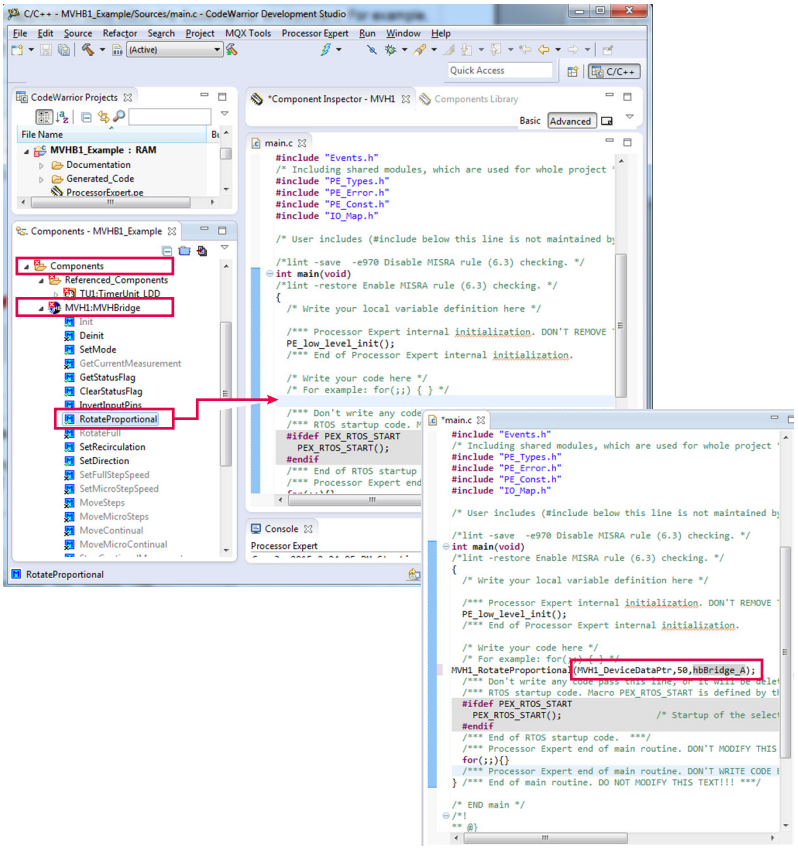

Figure 21. Using the interface

Hovering the mouse over any of the methods displays a description of the method, including a list of required parameters (Figure 22). The MVHBridge component encompasses a help, describing component properties, methods and typical use. To show the help, do the following:

1. In the **Components** view, right click the MVHBridge component and select **Help on Component**.
2. The web page with the help information displays.

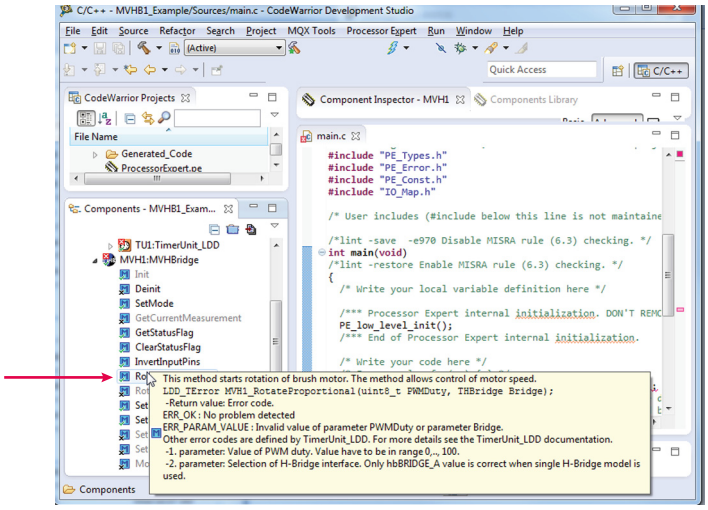

Figure 22. Additional information

4.8 Compiling, downloading and debugging

To compile/download and debug on the board, click compile and the debug button in the tool bar. CW downloads and launches the program on board (see Figure 23).

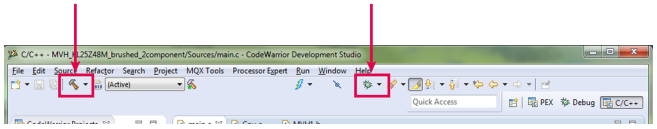

Figure 23. Compile and download application

5 References

The following are URLs for additional information related to NXP products and application solutions:

Table 7. References

NXP.com Support Pages	Description	URL
MC33926	Product Summary Page	www.nxp.com/MC33926
MC33931	Product Summary Page	www.nxp.com/MC33931
MC33932	Product Summary Page	www.nxp.com/MC33932
Kinetis Design Studio	Software	www.nxp.com/kds
CodeWarrior	Software	www.nxp.com/codewarrior
Processor Expert Code Model	Code Walkthrough Video	www.nxp.com/video/processor-expert-code-model-codewarrior-code-walkthrough:PROEXPCODMODCW_VID

5.1 Support

Visit www.nxp.com/support for a list of phone numbers within your region.

5.2 Warranty

Visit www.nxp.com/warranty to submit a request for tool warranty.

6 Revision History

Revision	Date	Description of Changes
1.0	3/2016	• Initial release

How to Reach Us:

Home Page:
NXP.com

Web Support:
http://www.nxp.com/support

Document Number: PEXMVHBRIDGESWUG
Rev. 1.0
3/2016