NXP Semiconductors
User's guide

Document Number: PEXLVHBRIDGESWUG
Rev. 1.0, 2/2016

Low-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 low-voltage H-Bridge component is a software driver capable of controlling both DC brushed motors and stepper motor. The component provides an API layer between the hardware and the user application. It makes make application development less time consuming by offering an easy-to-use interface for setting motor control options.

Two sets of methods are associated with the low-voltage H-Bridge component, one for DC brushed motors control and one for stepper motor control.

A single H-Bridge device is sufficient for controlling brushed DC motor. The low-voltage H-Bridge component provides two ways of controlling the motor. The first approach is uses a timer device (TimerUnit_LDD component) that initiates on/off switching and controls motor speed. The second approach uses the GPIO pins (BitIO_LDD components) to enable basic motor state control and to switch the motor on and off.

Stepper motors require a dual H-Bridge model to generate the four output signals needed to control the motor. The low-voltage H-Bridge component allows the motor to be controlled in either full-stepping mode or in micro-stepping control mode (which enables more precise control of the motor).

The Low-Voltage H-Bridge component supports the following analog parts:

- NXP MPC17C724 0.4 A Dual H-Bridge Motor Driver
- NXP MPC17510 1.2 A 15 V H-Bridge Motor Driver
- NXP MPC17511 1.0 A 6.8 V H-Bridge Motor Driver
- NXP MPC17529 0.7 A Dual H-Bridge Motor Driver with 3.0V/5.0V Compatible Logic I/O
- NXP MPC17531A 0.7 A Dual H-Bridge Motor Driver with 3.0 V Compatible Logic I/O
- NXP MPC17533 0.7 A 6.8 V Dual H-Bridge Motor Driver
- NXP MC34933 1.4 A Dual H-Bridge Driver Compatible with 3.0 V Logic

NXP offers following board solutions based on these chips:

- FRDM-17C724-EVB
- FRDM-17510EJ-EVB
- FRDM-17511EP-EVB
- FRDM-17511EV-EVB
- FRDM-17529EV-EVB
- FRDM-17531EP-EVB
- FRDM-17531EV-EVB
- FRDM-17533EV-EVB
- FRDM-34933EP-EVB

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

2 Low-Voltage H-Bridge Compatibility

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, 2 to 4 channels) is required for direct input control (IN1 to IN4)
- **GPIOs** are required for the device enable pins (OE, EN, GIN, PSAVE) and for output signal generation (in GPIO mode)

2.2 Supported Devices

The LVHBridge component supports following devices: MPC17C724, MPC17510, MPC17511, MPC17529, MPC17531A, MPC17533, MC34933. All these ICs incorporate internal control logic, a charge pump, gate drive, and high current, low $R_{DS(on)}$ MOSFET output circuitry.

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

- PWM frequencies up to 200 kHz
- Undervoltage shutdown
- Cross-conduction suppression

Table 1. Device differences

Device Number	Number of H-Bridges	Power Supply (V)		Gate driver signal to external MOSS-FET switch	Output current [A]	
		Logic Circuit Power Supply	Motor Driver Power Supply		Continuous	Peak
MPC17510	1	5.0 V	15 V	yes	1.2 A	3.8 A
MPC17511	1	5.0 V	5.0 V	yes	1.0 A	3.0 A
MPC17529	2	5.0 V	5.0 V	no	0.7 A	1.4 A
MPC17531A	2	3.0 V	5.0 V	no	0.7 A	1.4 A
MPC17533	2	5.0 V	5.0 V	no	0.7 A	1.4 A
MC34933	2	3.0 V	5.0 V	no	1.0 A	1.4 A
MPC17C724	2	3.0 V	3.0 V	no	0.4 A	0.8 A

2.3 Supported MCUs

The LVHBridge component is fully compatible with the following Freedom boards:

- K20D50M
- KL25Z
- KL26Z
- KL27Z
- KL46Z

3 LVHBridge Component

This section summarizes the capabilities of the LVHBridge component properties and methods. This information is applicable to both the Kinetis Design Studio (KDS) Integrated Development Environment (IDE) an the CodeWarrior (CW) IDE.

The LVHBridge component is located under the Components folder in the IDE **Components** panel. This folder contains two sub-folders: Referenced_Components (containing components to configure the timer unit Logical Device Driver (LDD) and SPI communication properties) and LVH1:LVHBridge (containing the Low-Voltage H-Bridge component).

The component features built-in Help documentation, which can be accessed by right-clicking on the component name in the **Components** panel and selecting the **Help on Component** menu item. The Help window then provides information on all of the component's properties, methods and events.

3.1 Component Settings

Selecting the LVHBridge component in the IDE's **Components** panel gains access to properties in the **Component Inspector** window. These properties determine the component's general settings and its behavior after initialization. Application code can later change some of these properties using the provided API methods.

3.1.1 General Settings

Component settings in the Component Inspector view are organized into a tree structure. The H-Bridge Model is the root element of the tree.

ActiveMode defines the H-Bridge device operational mode (normal or power-conserving sleep mode), which is controlled by the enabling pin. Selection of the enabling pin is in the **Enable Pins** group. For more information, see H-Bridge model's data sheet. The mode can be changed later using the C code method **SetMode**.

The **Motor Control** group involves timer settings, H-Bridge device and motor control settings. The **Timer Settings** group contains the **Primary Timer Component** property (the name of a linked **TimerUnit_LDD** component) and the name of the hardware timer being used (defined in the **Primary Timer Device** property). **Secondary Timer** encompasses the properties of an additional timer.

Note that the **Secondary Timer Component** property must use a different **TimerUnit_LDD** component than the **Primary Timer Component** property. The purpose of the primary and secondary timers is to allow the input control pins of an H-Bridge device to be connected to different timers (this applies for some freedom H-Bridge boards and freedom MCUs). But these timers must be synchronized to control a stepper motor. So the primary timer is designed to be the source for the global time base and the secondary timer is synchronized with the primary timer. See MCU's data sheet to find out which timer provides the global time base (GTB) and set the **Primary Timer Device** property accordingly. An example of a timer selection using the FRDM-KL25Z MCU is shown in Figure 1. For single timer, set the **Secondary Timer Component** to **Disabled**.

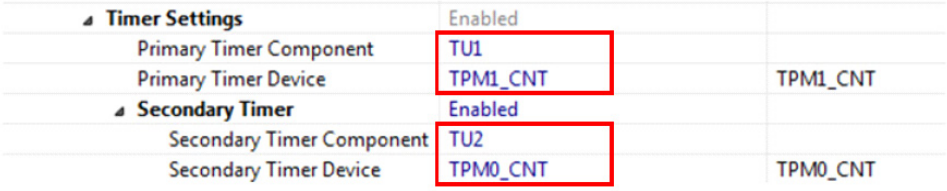

Figure 1. Selection of a FRDM-KL25Z MCU Primary and a Secondary Timer device

The H-Bridge 1 MCU Interface and H-Bridge 2 MCU Interface sets H-Bridge control functions. The H-Bridge 2 MCU Interface is shown only for dual H-Bridge models (for example MC34933). The **Input Control Pins** selects the H-Bridge input control pins that utilize the timer's channels or GPIO pins.

H-Bridge 1 MCU Interface		
DC brush		
Control Mode	Speed Control	
PWM Frequency	10 kHz	10.001 kHz
Direction Control	Bidirectional	
Init. Direction	Forward	
Input Control Pins	Two PWM Pins	
Pin for IN1A	PTD4/LLWU_P14/SPI1_PCS0/UART...	
Pin for IN1B	PTA12/TPM1_CH0	
H-Bridge 2 MCU Interface	Enabled	
DC brush		
Control Mode	State Control	
Init. Direction	Forward	
Input Control Pins	Two GPIO Pins	
Pin for IN2A	TSI0_CH5/PTA4/I2C1_SDA/TPM0_...	
Pin for IN2B	PTA5/USB_CLKIN/TPM0_CH2	
Auto Initialization	yes	

Figure 2. LVHBridge component – general settings

3.1.2 Setting up a Project to Control a DC Brushed Motor

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

*Component Inspector - LVH1 | Components Library | Basic | Advanced

Properties | Methods | Events

Name	Value	Details
Component Name	LVH1	
H-Bridge Model	MPC17511	
ActiveMode	yes	
Enable Pins	Enabled	
Pin for EN	CMP0_IN1/PTC7/SPI0_MISO/SPI0_...	
Pin for GIN	ADC0_SE14/TSI0_CH13/PTC0/EXT...	
Motor Control	Brushed	
Timer Settings	Enabled	
Primary Timer Component	TU1	
Primary Timer Device	TPM0_CNT	TPM0_CNT
Secondary Timer	Enabled	
Secondary Timer Component	TU2	
Secondary Timer Device	TPM1_CNT	TPM1_CNT
H-Bridge 1 MCU Interface		
DC brush		
Control Mode	Speed Control	
PWM Frequency	10 kHz	10.001 kHz
Direction Control	Bidirectional	
Init. Direction	Forward	
Input Control Pins	Two PWM Pins	
Pin for IN1	PTD4/LLWU_P14/SPI1_PCS0/UART...	
Pin for IN2	PTA12/TPM1_CH0	
Auto Initialization	yes	

Figure 3. Setup of the component to control a brush motor

2. Set the **Control Mode** property. There are two ways to control the DC brushed motor:
   a) **Speed Control** - motor speed is controlled by this setting. The **TimerUnit_LDD** component is used to generate the PWM signal. The **PWM Frequency** property is visible in this mode only. If the **Speed Control** mode is set on both interfaces (i.e. Interface 1 and Interface 2), the **PWM Frequency** property on Interface 2 sets automatically to the same value as Interface 1 (because Interface 2 uses the same timer).

b) **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 it does not use timer channels. If **State Control** is set on both interfaces or only a single H-Bridge model (one interface) with **State Control**, the **TimerUnit_LDD** component is not required anymore by the LVHBridge 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 to **Forward** restricts the motor's movement to the forward direction only. Setting the property to **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 **Speed Control** mode is set in the **Control Mode** property.

3.1.3 Setting up a Project to Control a Stepper Motor

Select the dual H-Bridge model to configure and set **Stepper** in the **Motor Control** property. Note that the dual H-Bridge model is required, because a two phase bipolar stepper motor has four inputs.

*Component Inspector - LVH1 | Components Library | Basic | Advanced

Properties | Methods | Events

Name	Value	Details
Component Name	LVH1	
H-Bridge Model	MPC17C724	
ActiveMode	yes	
Enable Pins	Enabled	
Pin for PSAVE	CMP0_IN1/PTC7/SPI0_MISO/SPI0_...	
Motor Control	Stepper	
Timer Settings	Enabled	
Primary Timer Component	TU1	
Primary Timer Device	TPM1_CNT	TPM1_CNT
Secondary Timer	Enabled	
Secondary Timer Component	TU2	
Secondary Timer Device	TPM0_CNT	TPM0_CNT
Stepper Motor		
Output Control	PWM	
Motor Control Mode	Full-step and Micro-step	
Full-step Configuration		
Speed	100 D	
Acceleration	400 D	
Micro-step Configuration		
PWM Frequency	20 kHz	20.011 kHz
Micro-steps per Step	8 Micro-steps	
Speed	400 D	
Acceleration	400 D	
H-Bridge 1 MCU Interface		
Input Control Pins	Two PWM Pins	
Pin for IN1A	PTD4/LLWU_P14/SPI1_PCS0/UART...	
Pin for IN1B	PTA12/TPM1_CH0	
H-Bridge 2 MCU Interface	Enabled	
Input Control Pins	Two PWM Pins	
Pin for IN2A	TSI0_CH5/PTA4/I2C1_SDA/TPM0_...	
Pin for IN2B	PTA5/USB_CLKIN/TPM0_CH2	
Auto Initialization	yes	

Figure 4. Component settings to control a stepper motor

In the **Stepper Motor** group, set the properties that apply to the selected environment.

- The **Output Control** property defines the control method.
  a) With **PWM** selected the component utilizes four channels of a timer to control the stepper motor. Signal is generated in hardware and micro-step mode is also available.
  b) In **GPIO** mode, GPIO pins are used instead of timer channels and only full-step mode is available (no micro-step mode).

- **Manual Timer setting** property is only visible when the visibility of the component properties is switched to **Advanced**. It is designed to change the Counter frequency of the linked **TimerUnit_LDD** component. By default the Counter frequency is set automatically by LVHBridge component. In some cases the frequency value does not have to be set appropriately (or instance, if the user wants to set a different value or an error has occurred). For more information see Section 3.1.3.1, "Stepper Motor Speed".
- **Motor Control Mode** allows to select the step mode. Selecting full-step and micro-step mode allows to switch between full-stepping and micro-stepping in C code.

a) 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 5. The acceleration setting is 400, as shown in Figure 4.

- Desired motor speed is set to 100 full-steps per second. This value is defined by the **Speed** property in the Processor Expert GUI and can be changed in C code.
- Acceleration and deceleration is set to 400 full-steps per second^2. This value is defined by the **Acceleration** property. Note that the motor reaches the speed in 0.25 second (desired_speed / acceleration = 100 / 400 = 0.25).

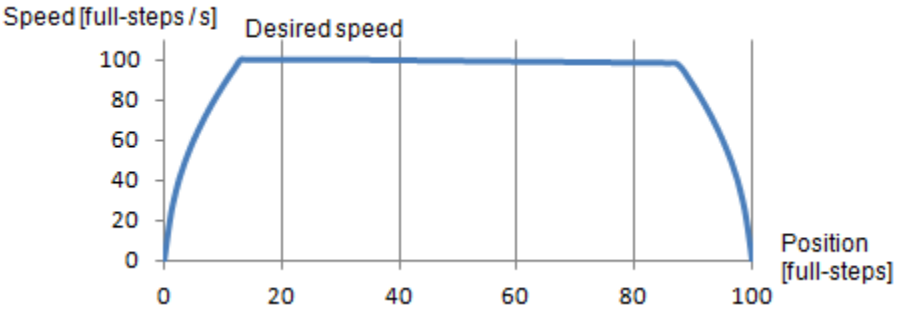

Figure 5. Acceleration and deceleration ramp

b) **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.

3.1.3.1 Stepper Motor Speed

The LVHBridge component defines the stepper motor's minimum and maximum speed. These limit values are used by the component methods. Minimum speed in full-step and micro-step modes is one step per second. Maximum speed is 5000 steps per second. There is a specific case in which minimum full-stepping speed is affected by timer input frequency. This applies only when one FTM timer is used to control the stepper motor. In this case, the **Primary Timer Device** property must use FTM timer values (FTM0_CNT, or FTM1_CNT, etc.). The **Secondary Timer** property must be set to **Disabled**. The Stepper Motor **Output Control** property must be set to PWM. Figure 6 illustrates this configuration.

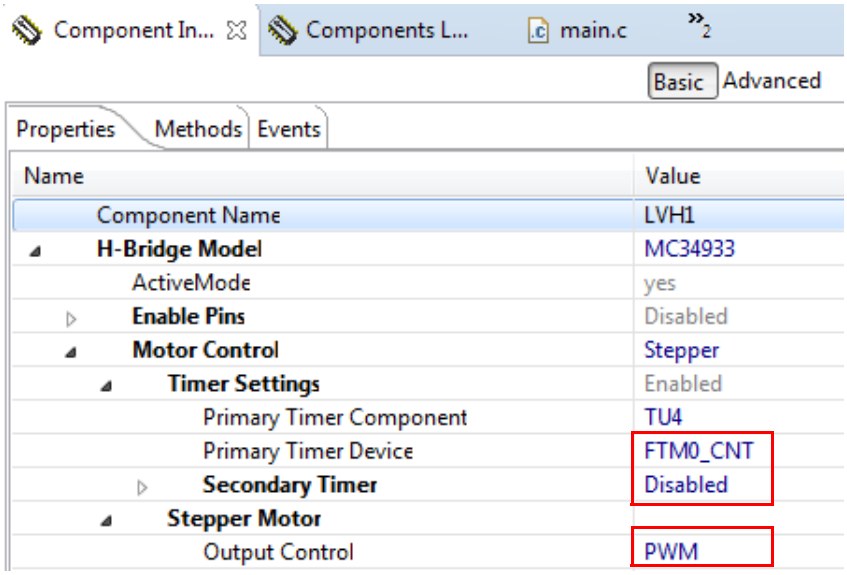

Figure 6. Stepper Mode configuration affecting minimum Full-stepping speed

Possible values for the timer input frequency (**Counter frequency** property in **TimerUnit_LDD**) are in Table 2. Input frequency values depend on LVHBridge component settings. Note that two frequency values are needed in "Full-step and Micro-step mode" in one case (LVHBridge component switches in runtime between these two values).

Table 2. Minimum and maximum timer input frequency per stepper control mode

Mode Description	LVHBridge component properties				Primary Timer Input Frequency			Secondary Timer Input Frequency
	Timer Device	Secondary Timer	Output Control	Motor Control Mode	Values	Min	Max	
Full-step mode	TPM	Don't care	PWM	Full-step	1	131 kHz	1 MHz	Any value (user selection)
Full-step and Micro-step mode	TPM	Don't care	PWM	Full-step and Micro-step	1	1.2 MHz	10 MHz	Any value (user selection)
Full-step mode (SW control)	FTM or TPM	Disabled	GPIO	Full-step	1	131 kHz	1 MHz	Secondary timer is not enabled
Full-step mode	FTM	Disabled	PWM	Full-step	1	131 kHz	1 MHz	Secondary timer is not enabled
Full-step mode	FTM	Enabled	PWM	Full-step	1	131 kHz	1 MHz	The same values as for primary timer
Full-step and Micro-step mode	FTM	Disabled	PWM	Full-step and Micro-step	2	1st value for Full-step: 131 kHz 2nd value for Micro-step:1.2 MHz	1st value for Full-step: 1 MHz 2nd value for Micro-step:10 MHz	Secondary timer is not enabled
Full-step and Micro-step mode	FTM	Enabled	PWM	Full-step	1	1.2 MHz	10 MHz	The same values as for primary timer

3.1.3.2 Computation of Minimum Full-stepping Speed

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

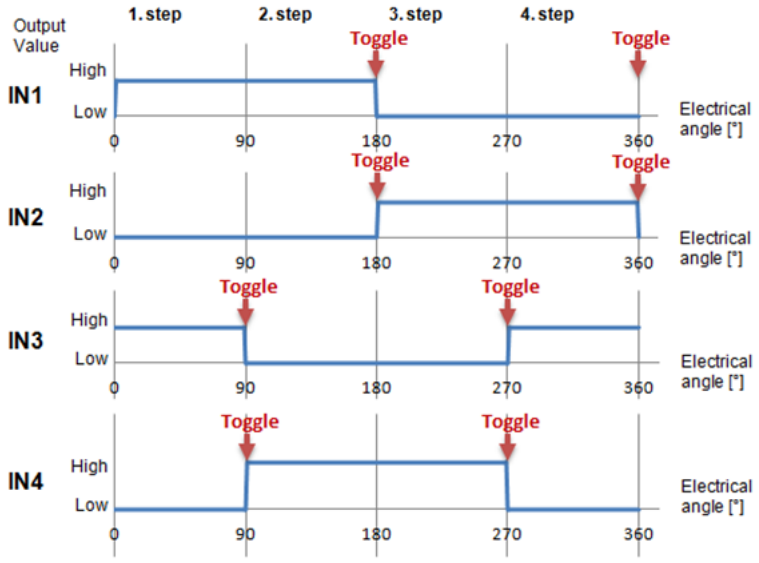

Figure 7. Generating the Full-step control signal

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). The minimum values for speed in the low-voltage H-Bridge can be found in the header file (see constant <component_name>_MIN_FULLSTEP_ SPEED). The formula for calculation of this value is as follows:

$$Speed_{min} = \frac{2 \ X \ Counter\_frequency}{65536} + 1$$

where:

Counter_frequency =input frequency of the timer device

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 187,500 Hz, the minimum speed is:

$$Speed_{min} = \frac{2 \ X \ Counter\_frequency}{65536} + 1 = \frac{2 \ X \ 187500}{65536} + 1 = 6.72$$

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

3.1.3.3 Setting the Minimum Full-stepping Speed

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

1. Launch Processor and select the LVHBridge component.
2. In the Processor Expert menu bar, set component visibility to **Advanced**.
3. In the Properties tab, find the **Motor Control** -> **Stepper Motor** -> **Manual timer** setting property and set the value to **Enabled**. Make sure that component visibility is set to **Advanced** (see Figure 8).
4. Set the **TimerUnit_LDD** frequency.
   a) In the Components view, double click on the **TimerUnit_LDD** component.
   b) Press the button in the **Counter frequency** field.
   c) Set the frequency value (187.5 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).

d) Set the **Allowed Error** value at 10% (see Figure 10).

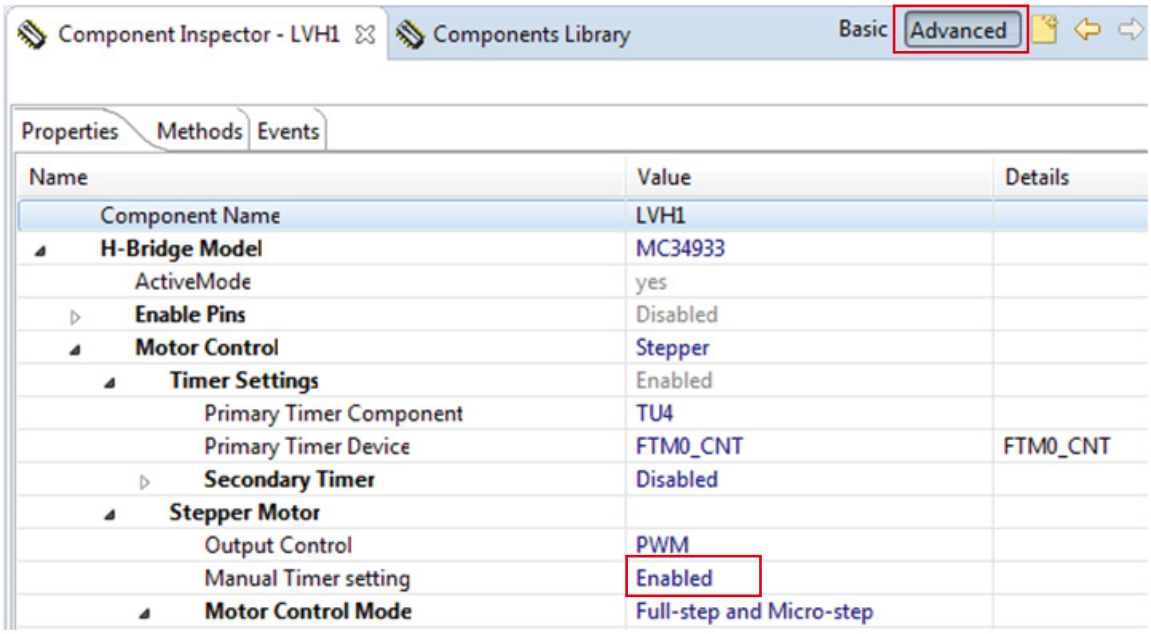

Figure 8. Enabling the Manual frequency setting

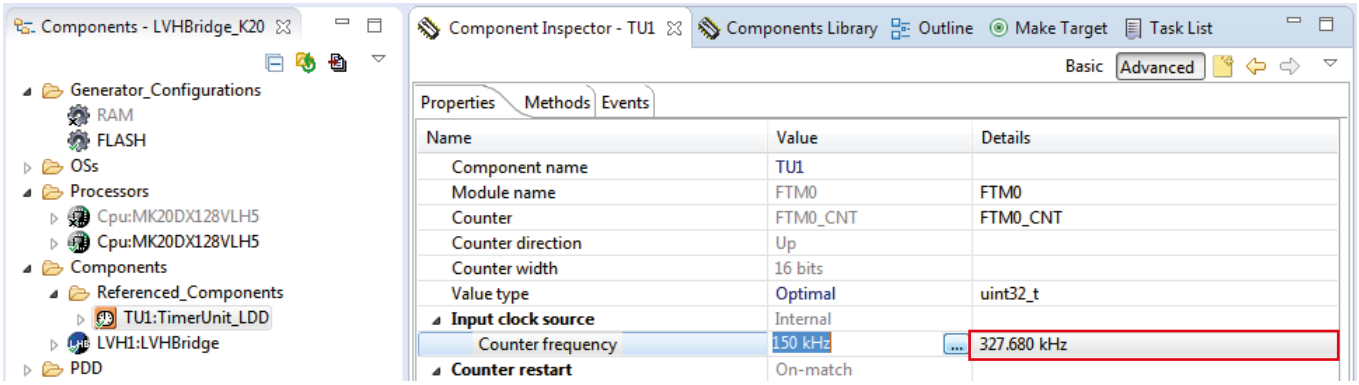

Figure 9. Component TimerUnit_LDD timing dialog

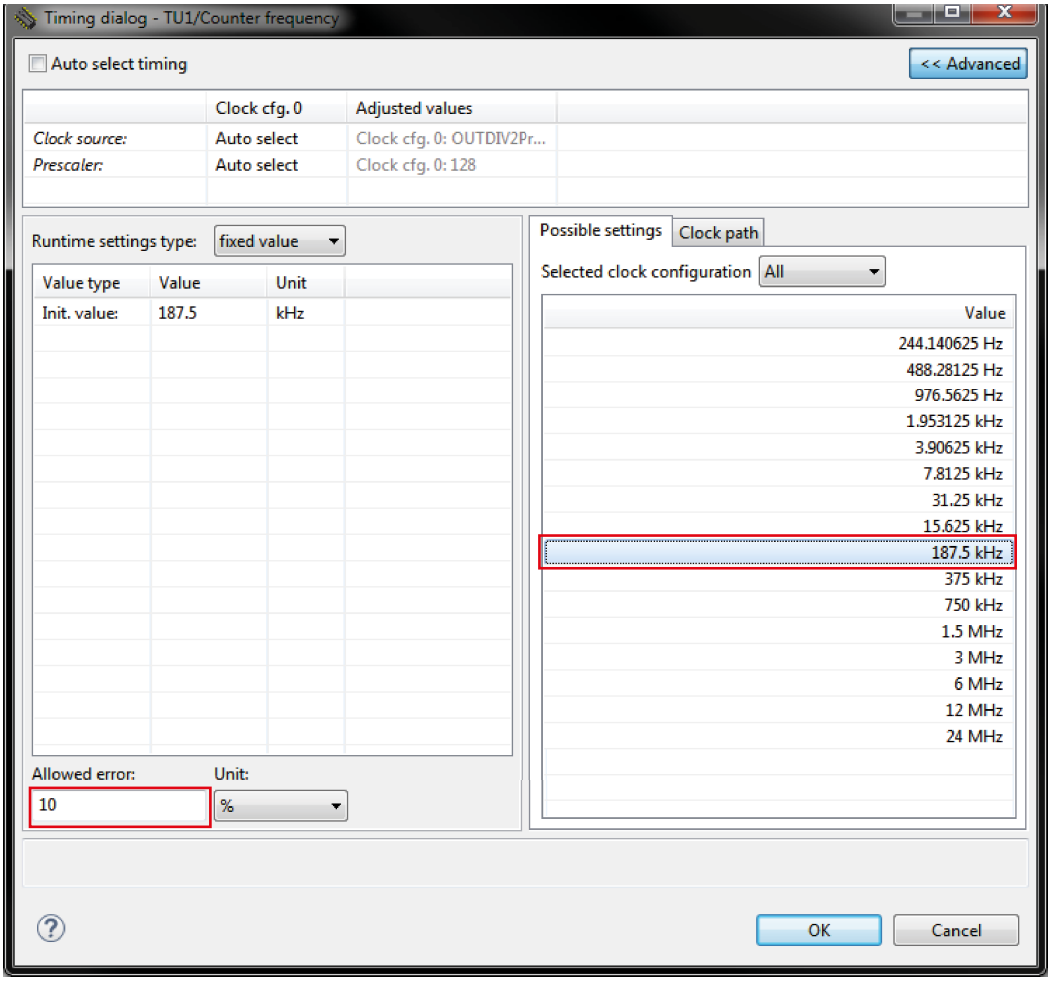

Figure 10. Component TimerUnit_LDD timing dialog—select input frequency

3.2 Component API

LVHBridge component provides API, which can be used for device real-time dynamic configuration. API methods and events are listed in Table 3.

Some methods/events are marked with ticks while others are marked with crosses. Only methods marked with ticks are generated. This behavior can be changed in Processor Expert Inspector under Methods tab. Note that methods displayed in grey color are always generated because they are needed for proper functionality. This forced behavior depends on various combinations of settings of component properties.

Table 3. LVHBridge component API descriptions

Method/Event	Description
Init	Initializes the device. Allocates memory for the device data structure, sets H-Bridge device mode, etc. This method can be called only once. Components linked by H-Bridge (TimerUnitLDD) are not initialized here.
SetMode	Sets the H-Bridge device mode using enable pin.
SetGateDriver	Controls the Gate Driver Input (GIN) pin. This method is available only for MPC17510 and MPC17511.
RotateProportional	Starts rotation of the brush motor. The method allows control of motor speed.
RotateFull	Starts rotation of the brush motor. This method is intended for state motor control (on/off) and is available only when the property **Control Mode** is set to **State Control**.
SetTriState	Sets the output of the specified H-Bridge to tri-state (high impedance) using input control pins.

Table 3. LVHBridge component API descriptions (continued)

Method/Event	Description
SetDirection	Sets the direction of the brush motor. In forward direction the first IN pin is set to high (or PWM) and the second IN pin to low. In reverse direction the first IN is set to low and the second IN to high (or PWM). Change of direction is applied when rotation starts (not when the motor is running).
SetFullStepSpeed	Sets motor speed in full-step mode. Unit is the number of full-steps per second. Motor speed cannot be changed while the motor is running.
SetMicroStepSpeed	Sets motor speed in micro-step mode. Unit is the number of micro-steps per second. The size of the micro-step depends on settings in Processor Expert (number of micro-steps per full-step). Motor speed cannot be changed while the motor is running.
MoveSteps	Moves the motor by a specified number of full-steps. When the rotor is not at physical full-step position then the Move-Steps method sets the nearest full-step position without correction. The number of steps returned by the method [GetFull-StepPosition] are updated before they are executed. Thus, if a user calls the method [MoveSteps] with the parameter Steps equal to 100, a certain number of these steps are counted before they are physically executed (for example 64 steps when the **OutputControl** property is set to PWM and the FTM device is used). Wait for completion of this action before running the motor again (using method [GetMotorStatus] or event [OnActionComplete]).
MoveMicroSteps	Moves the motor by specified number of micro-steps. When the rotor is not at a physical micro-step position, the method sets the nearest micro-step without correction. For example, assume the size is initialized to 32 micro-steps per one full-step and the motor has executed three micro-steps. If the user then changes micro-step size to 2 micro-steps per one full-step and starts motor movement, the previous three micro-steps are discarded. Wait for completion of this action before running the motor again (using method [GetMotorStatus] or event [OnActionComplete]).
MoveContinual	Moves the motor continually in full-step mode. Stop the motor by calling the [StopContinualMovement] method. When the rotor is not at a physical full-step position, the method sets the nearest full-step without correction. This method is not available when the acceleration ramp is used.
MoveMicroContinual	Moves the motor continually in micro-step mode. Stop motor by calling the [StopContinualMovement] method. When the rotor is not at a physical micro-step position, the method sets the nearest micro-step without correction. For example, assume the size is initialized to 32 micro-steps per one full-step and the motor has executed three micro-steps. If the user then changes micro-step size to 2 micro-steps per one full-step and starts motor movement, the previous three micro-steps are discarded. This method is not available when the acceleration ramp is used.
StopContinualMovement	Stops the continual movement of the stepper motor. The method does not stop the motor immediately and the motor may execute several steps before stopping. In micro-step mode the motor does not have to stop at a full-step position and can stop anywhere. This method is not available when acceleration ramp is used or method [MoveContinual] or [MoveMicro-Continual] is not enabled. Wait for completion of this action before running the motor again (using method [GetMotorSta-tus] or event [OnActionComplete]).
GetMotorStatus	Returns the status of the stepper motor control. Possible values are defined in TMotorStatus enumeration in header file.
AlignRotor	Aligns the rotor to a full-step position. This method executes 4 full-steps forward (one electrical revolution) at minimum speed (see **component_name**_MIN_FULLSTEP_SPEED constant). These steps are not counted towards the number of full-steps. Wait for completion of this action before running the motor again (using method [GetMotorStatus] or event [OnActionComplete]).
SetMicroStepSize	Serves to change size of micro-step. Note that the size of the micro-step is initialized to the value set in the Processor Expert property **Micro-steps per Step**. The motor must not be running during this method. The method is available only when micro-stepping is enabled.
GetFullStepPosition	Returns the current full-step position. The position resets to zero when an initialization of the H-Bridge component occurs. It can be reset by method [ResetFullStepPosition].
GetMicroStepPosition	Returns the current micro-step position. The size of the micro-step depends on the property **Micro-steps per Step** in the Processor Expert component GUI. The position resets to zero when an initialization of the H-Bridge component occurs. It can be reset by method [ResetFullStepPosition].
ResetFullStepPosition	Sets the full-steps counter to zero.
DisableMotor	Sets IN pins output value to LOW. The method can be used to stop the stepper motor. The output value of the pins are not changed immediately because the counter registers are updated after the counter overflows (at the beginning of the next counter period). Note that the default behavior of the motor control is to hold position when a movement is completed.
OnActionComplete	This event is called when the motor reaches the desired number of steps or the motor is stopped by method [StopContinu-alMovement]. The handler is available only for stepper motors.

3.3 Utilized Components

The LVHBridge component uses several other components inherited from or shared with Processor Expert components. Re-usage of existing Processor Expert components speeds the development process and provides verified solutions for common tasks.

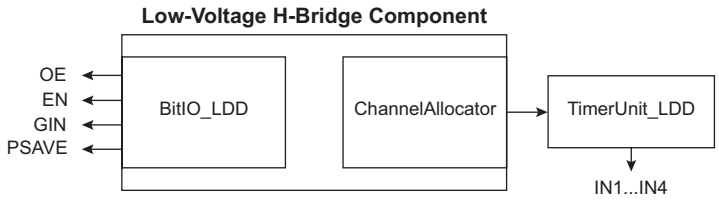

Figure 11. Components used by the LVHBridge component

Components description

TimerUnit_LDD	The component controls timer device
BitIO_LDD	Output signal generation (GPIO mode), output gate driver signal, gate driver input pin, enable pin control
Channel Allocator	Management of timer channels (2 allocators for up to 2 timers)

3.4 Implementation Notes

The LVHBridge component is designed to control a two phase bipolar stepper motor. Because a stepper motor uses electrical commutation to rotate, it requires a dual H-Bridge device. The basic control method is full-stepping which fully powers each coil in sequence. Increased precision is achieved by using the PWM to control coil current (open loop control). This method is called micro-stepping (available in the LVHBridge component.)

In both micro-step and full-step mode, the control motor speed, direction, acceleration and deceleration and the position of the stepper motor can be controlled.

The following application notes apply to stepper motor control:

- The LVHBridge component was tested with a core clock frequency ranging from 20 MHz (minimum value) to 120 MHz.
- Do not change the settings of the timer device (**TimerUnit_LDD**) linked by the LVHBridge 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 method **DisableMotor** to set H-Bridge outputs to LOW (coils are not powered).
- Forward motor direction indicates that steps are executed in the order depicted in Figure 12. IN1 through IN4 are the input pins of the H-Bridge device which control H-Bridge outputs. These pins input to the stepper motor. Connect the stepper motor to output pins OUT1-OUT4 and select control input pins on MCU in the component settings.
- The FTM or TPM timer device is needed by the stepper control logic.
- The **AlignRotor** method affects the position of the motor. This method executes four full-steps. It is available only when full-step mode is enabled.

3.4.1 Full-step Control Mode

The component uses normal drive mode where two coils are powered at the same time.

As mentioned in Section 3.1.3, "Setting up a Project to Control a Stepper Motor", a full-stepping signal can be generated either by using four channels of a timer or by using four GPIO pins. The signal generated by the MCU (inputs of H-Bridge device) using four timer channels is shown in Figure 12. The voltage levels applied to the coils of the stepper motor are depicted in Figure 13. Note that the voltage is applied to both coils at the same time.

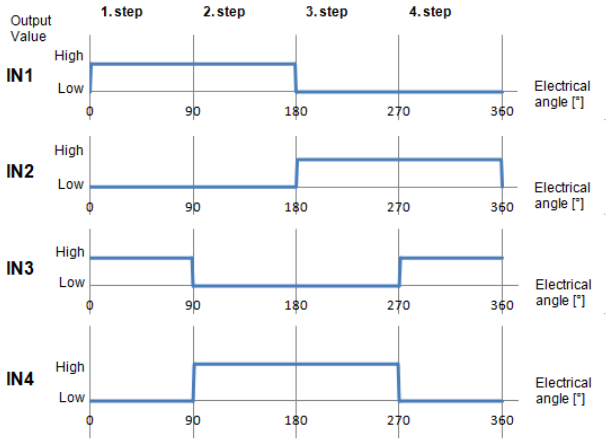

Figure 12. Signals of logic input pins generated by the MCU in Full-step mode

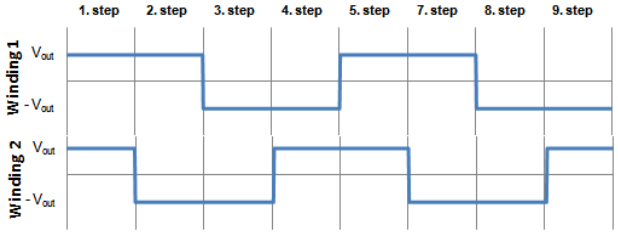

Figure 13. Output of the H-Bridge device in Full-step mode

3.4.2 Micro-step Control Mode

Micro-step control mode provides smoother motor movement and increased precision. The current varies in motor windings A and B depending on the micro-step position. A PWM signal is used to reach the desired current value (see the following equations). This method is called sine cosine micro-stepping.

$I_A = I_{MAX} \text{ X } \sin(\theta)$

$I_B = I_{MAX} \text{ X } \cos(\theta)$

where:

- I_A = the current in winding A
- I_B = the current in winding B,
- I_{MAX} = the maximum allowable current
- θ = the electrical angle

In micro-step mode, a full-step is divided into smaller steps (micro-steps). The LVHBridge component offers 2, 4, 8, 16 and 32 micro-steps per full-step. The micro-step size is defined by the property **Micro-steps per Step** and can be changed later in C code.

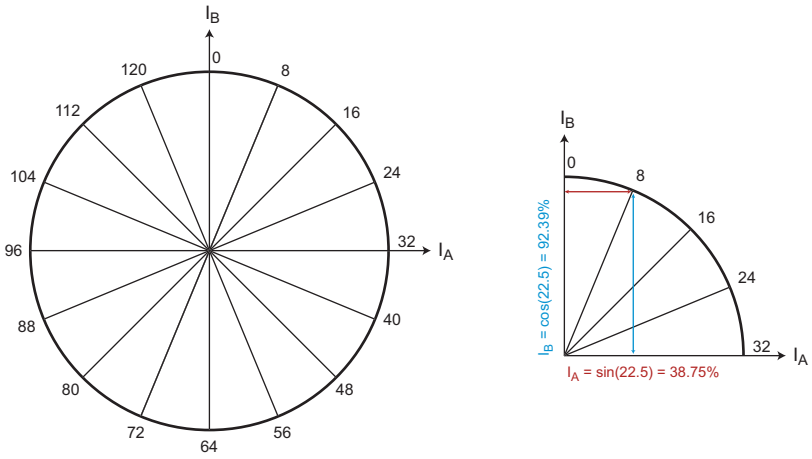

Figure 14. Micro-stepping phase diagram

The micro-stepping signal is generated using four timer channels (see Figure 15). Output from logic analyzer in Figure 16 shows the change of PWM duty with respect to the micro-step position. Current values applied to the stepper motor coils are depicted in Figure 17.

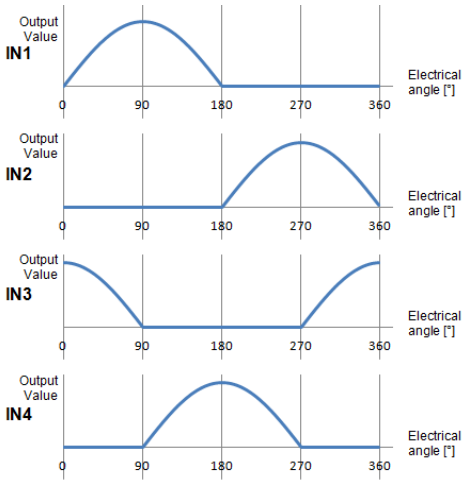

Figure 15. Logic input pin signals generated by the MCU in Micro-step mode

Figure 16. Logic Analyzer output

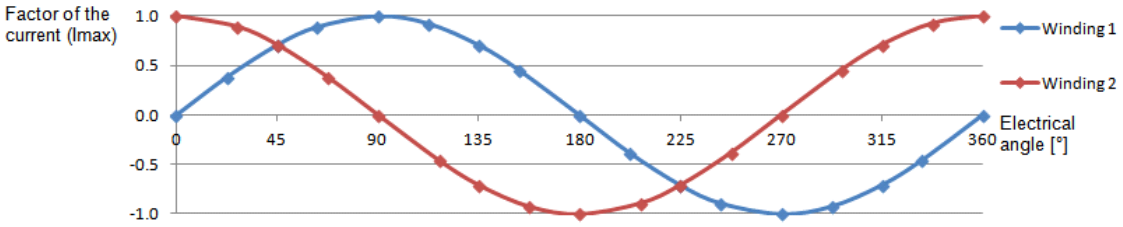

Figure 17. H-Bridge device output in Micro-step mode

3.5 Frequently Asked Questions

Q: Why do I occasionally unexpected behavior in my stepper or DC brushed motor?

A: Check the value of the signals on the enable and disable pins (D1, EN/D2, D3, EN/D4). These signals affect the H-Bridge device mode. To provide a wider range of MCU compatibility, some pins are wired to more than one MCU board pin using 0 Ω resistors. Check schematic and remove resistors as needed to disconnect unused pins.

Q: How do I set up the LVHBridge component when two or more components with conflicting values are configured to control brushed motors? (See Figure 18)

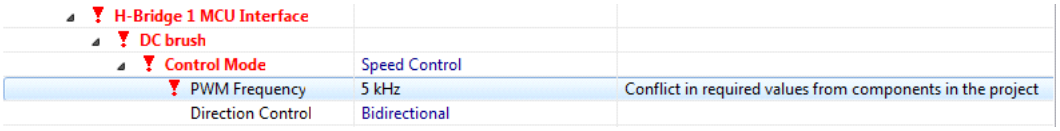

Figure 18. Conflict in the required values for components in the project

A: You can use more than one LVHBridge components in the same project. These components can share the same timer device in brushed motor control mode, but the **PWM Frequency** and **Timer Device** properties must conform in all of the components.

Q: I sometimes get the following unexpected error while generating Processor Expert code: "Generator: FAILURE: Unexpected status of script: Drivers\Kinetis\TimerUnit_LDD.drv, please contact Freescale support". What causes this? (Figure 19)

A: Occasionally, when the LVHBridge component is enabled in the project, the **TimerUnit_LDD** component channels have not been allocated. If this occurs, changing certain MVHBridge properties forces allocation of the channels. For configuring a stepper motor (**Motor Control** property set to **Stepper**), try changing the **Output Control** property to **GPIO** and then back to **PWM**. For configuring a brushed motor (**Motor Control** property set to **Brushed**), change the **Control Mode** property to **State Control** and then back to **Speed Control** on interface 1 or interface 2.

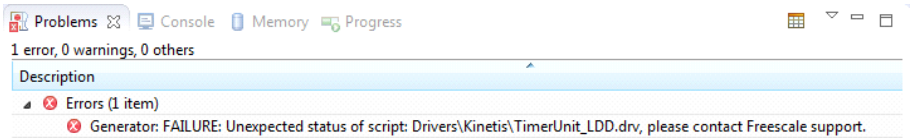

Figure 19. Unexpected error related to the LVHBridge TimerUnit_LDD component

Q: I have set up several CPU clock configurations (via the Clock configurations property of the CPU component). Sometimes during runtime, when I switch between these configuration (using the CPU **SetClockConfiguration** method), the speed of the stepper motor appears to be inaccurate. Why does this occur?

A: Switching to a different configuration results in the use of a different input frequency by a timer device.

Q: What does the error message “The component has no method to enable its event (**OnCounterRestart**) raised in an LVHBridge **TimerUnit_LDD** component mean?

A: This occurs only when an LVHBridge component is added to a project and set the **Motor Control** property to **Stepper**. The errors disappear if any property of the LVHBridge component is changed.

4 Installing the Software

Processor Expert software is available as a 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 this link:
www.nxp.com/products/software-and-tools/software-development-tools/processor-expert-and-embedded-components:BEAN_STORE_MAIN?fsrch=1&sr=1&pageNum=1

4.1 Installing Kinetis Design Studio

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

NOTE

The component and examples in the component package are intended for Kinetis Design Studio 3.0.0 and CodeWarrior 10.6. Skip this section, if Kinetis Design Studio 3.0.0 and CodeWarrior 10.6 are already installed on your system.

1. Obtain the latest Kinetis Design Studio installer file from the NXP KDS website here:
   www.nxp.com/products/software-and-tools/run-time-software/kinetis-software-and-tools/ides-for-kinetis-mcus/kinetis-design-studio-integrated-development-environment-ide:KDS_IDE?tid=PEH
2. Run the executable file and follow the instructions.

4.2 Downloading the LVHBridge Component and Example Projects

The examples used in this section are based on a pre-configured CodeWarrior project. First download the project and its associated components:

1. Go to the NXP website http://www.nxp.com/LVHBRIDGE-PEXPERT
2. Download the zip file with components and example projects.
3. Unzip the downloaded file and check that the folder contains the files listed in Table 4.

Table 4. LVHBridge example project and components

Folder Name	Folder Contents
CodeWarrior_Examples	**Example project folder for CodeWarrior.**
LVH_KL25Z_brush_MC34933	Example project for DC brush motor control using FRDM-34933EP-EVB H-Bridge board and FRDM-KL25Z MCU board
LVH_KL25Z_brush_MPC17510	Example project for DC brush motor control using FRDM-17510EJ-EVB H-Bridge board and FRDM-KL25Z MCU board
LVH_KL25Z_stepper	Example project intended to control stepper motor using FRDM-34933EP-EVB H-Bridge board and FRDM-KL25Z MCU board
LVH_KL25Z_stepper_ramp	Example project intended to control stepper motor using FRDM-34933EP-EVB H-Bridge board and FRDM-KL25Z MCU board. Acceleration ramp is enabled
Component	Processor Expert component folder
KDS_Examples	**Example project folder for Kinetis Design Studio 3.0.0 or newer.**
LVH_K20D50M_brush_MC34933	Example project for DC brush motor control using FRDM-34933EP-EVB H-Bridge board and FRDM-K20D50M MCU board
LVH_K20D50M_brush_MPC17510	Example project for DC brush motor control using FRDM-17510EJ-EVB H-Bridge board and FRDM-K20D50M MCU board
LVH_K20D50M_stepper_bitIO	Example project intended to control stepper motor using FRDM-34933EP-EVB H-Bridge board and FRDM- K20D50M MCU board
LVH_K20D50M_stepper_ramp_bitIO	Example project intended to control stepper motor using FRDM-34933EP-EVB H-Bridge board and FRDM- K20D50M MCU board. Acceleration ramp is enabled
LVH_KL25Z_brush_MC34933	Example project for DC brush motor control using FRDM-34933EP-EVB H-Bridge board and FRDM-KL25Z MCU board
LVH_KL25Z_brush_MPC17510	Example project for DC brush motor control using FRDM-17510EJ-EVB H-Bridge board and FRDM-KL25Z MCU board
LVH_KL25Z_brush_FreeMASTER	Example project intended to control DC brush motor using FreeMASTER tool. Latest Freemaster installation package: http://www.nxp.com/freemaster
LVH_KL25Z_step_FreeMASTER	Example project intended to control stepper motor using FreeMASTER tool
LVH_KL25Z_stepper	Example project intended to control stepper motor using FRDM-34933EP-EVB H-Bridge board and FRDM-KL25Z MCU board
LVH_KL25Z_stepper_ramp	Example project intended to control stepper motor using MC34933 H-Bridge freedom board and FRDM-KL25Z MCU board. Acceleration ramp is enabled
LVH_KL26Z_stepper	Example project intended to control stepper motor using FRDM-34933EP-EVB H-Bridge board and FRDM-KL26Z MCU board
LVH_KL26Z_stepper_iar	Example project intended to control stepper motor using FRDM-34933EP-EVB H-Bridge board and FRDM-KL26Z MCU board. IAR compiler is used instead of GNU C compiler

4.2.1 Importing the LVHBridge Component into the Kinetis Design Studio

1. Launch Kinetis Design Studio. When the Kinetis Design Studio IDE opens, go to the menu bar and click **Processor Expert** -> **Import Component(s)**.
2. In the pop-up window, locate the component file (.PEupd) in the example project folder LVHBridge _PEx_SW\Component. Select **LVHBridge_b1508.PEupd** and **ChannelAllocator_b1508.PEupd** files then click **Open** (see Figure 20).
3. Select the repository where the components are to be imported and confirm by clicking the OK button. (This step applies only to the Kinetis Design Studio IDE. In CodeWarrior, components are automatically imported into the predefined repository.)
4. If the import is successful, the LVHBridge component appears in Components Library -> SW -> User Component (see Figure 21). Note that the component **ChannelAllocator** is not visible, because it is not designed to be users accessible.

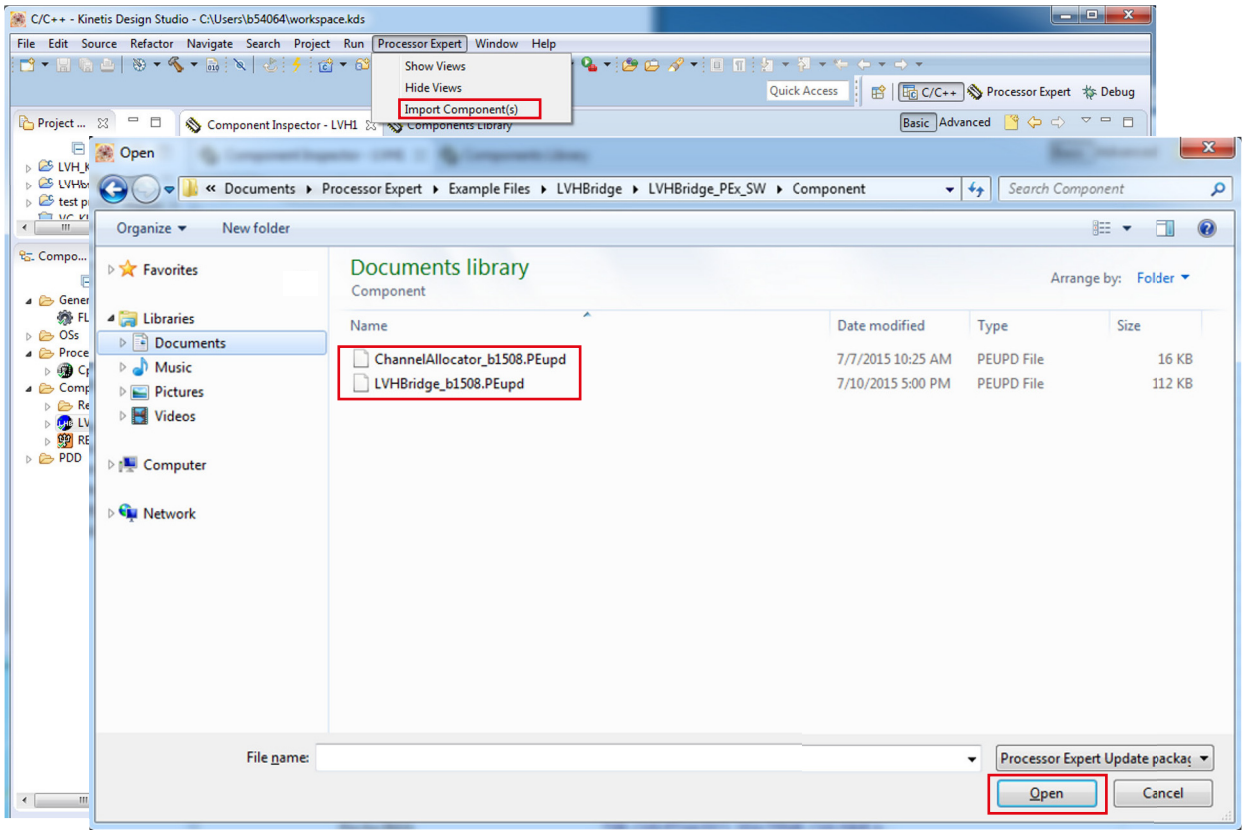

Figure 20. Importing the LVHBridge component

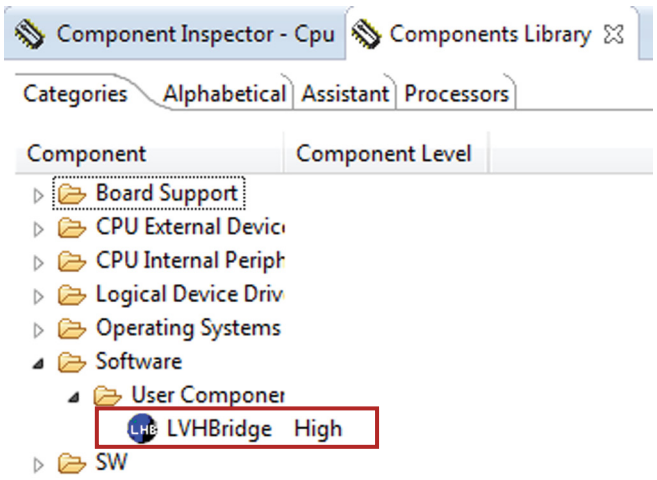

Figure 21. LVHBridge component location after CodeWarrior import

The LVHBridge component is ready to use.

4.2.2 Importing an Example Project into Kinetis Design Studio

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

1. In the Kinetis Design Studio menu bar, click **File** -> **Import**… In the pop-up window, select **General** -> **Existing Projects into Workspace** and click **Next.**
2. Locate the example in folder: LVHBridge_PEx_SW\CodeWarrior_Examples (see Figure 22, which shows LVH_KL25Z_brush_MC34933 as the imported project). Then click **Finish**.

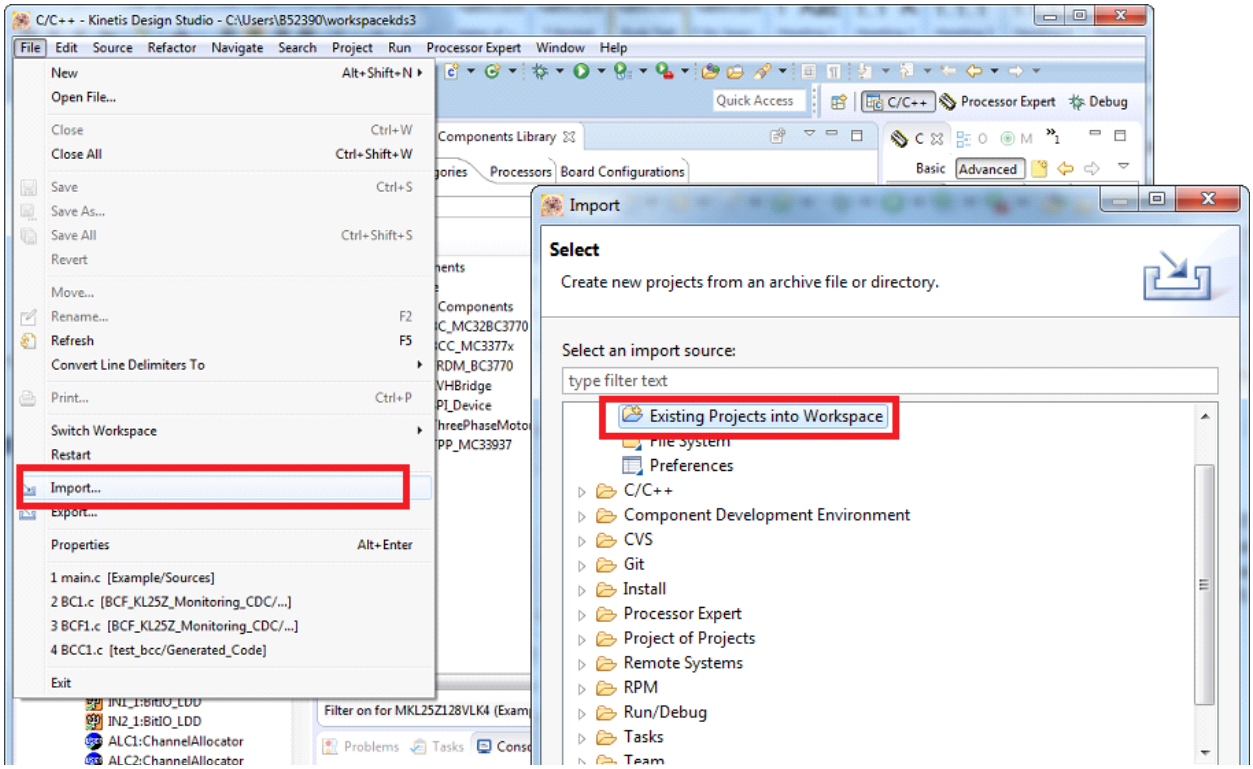

Figure 22. Example project import (a)

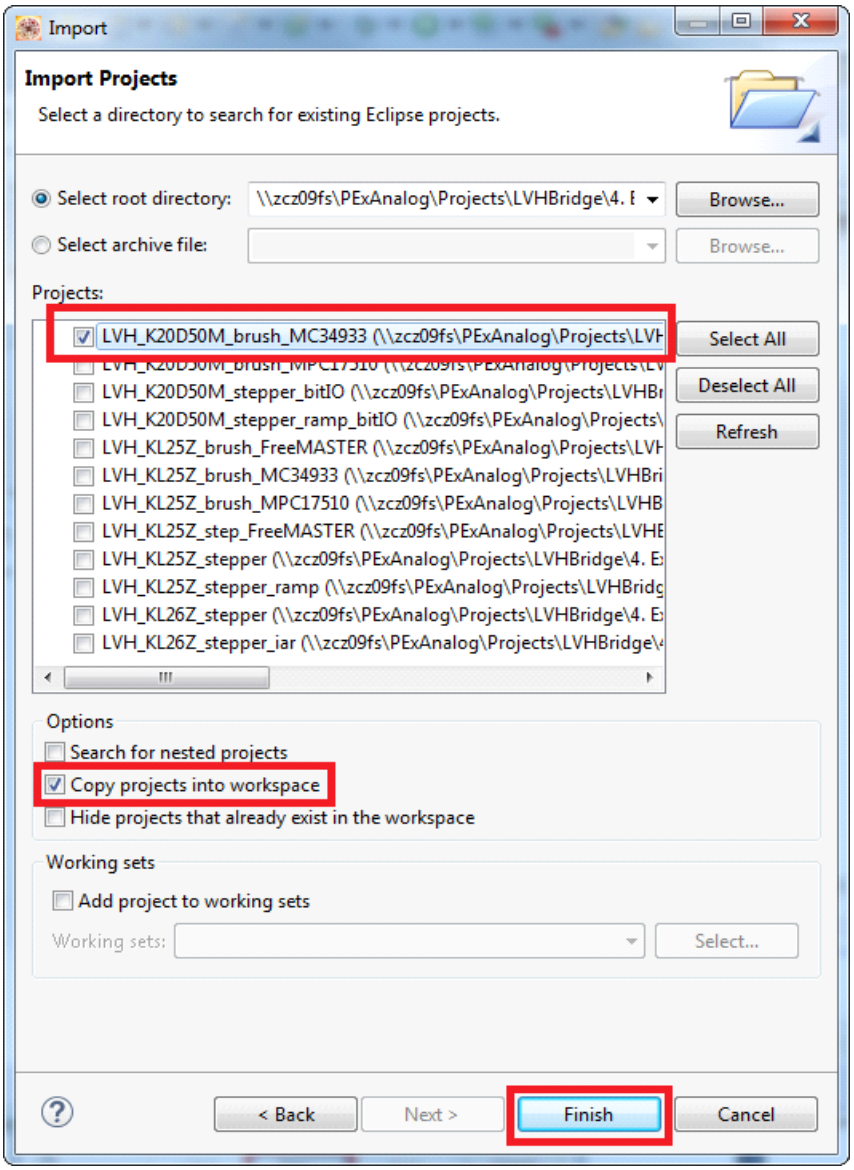

Figure 23. Example project import (b)

The project is now in the CodeWarrior workspace.

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

If an example project is not being used, the following instructions describe how to create and setup a new project that uses the LVHBridge component. If the LVHBridge component is not in the Processor Expert Library, follow the steps in Section 4.2.1, "Importing the LVHBridge Component into the Kinetis Design Studio".

1. In the Kinetis Design Studio menu bar, select **File** -> **New** -> **Kinetis Project**. When the New Kinetis Project dialog box opens, enter a project name into the text box and then click **Next** as shown in Figure 24.
2. In the **Devices** dialog box, select the MCU class used by the project (in this case MKL25Z128xxx4 was selected). Then click **Next**.
3. In the **Rapid Application Development** dialog box, make sure that the **Processor Expert** option is selected. Then click **Finish** as shown in Figure 25.
4. In the **Processor Expert Target Compiler** dialog box, select the compiler to use (GNU C Compiler in Figure 25) and then click **Finish**.

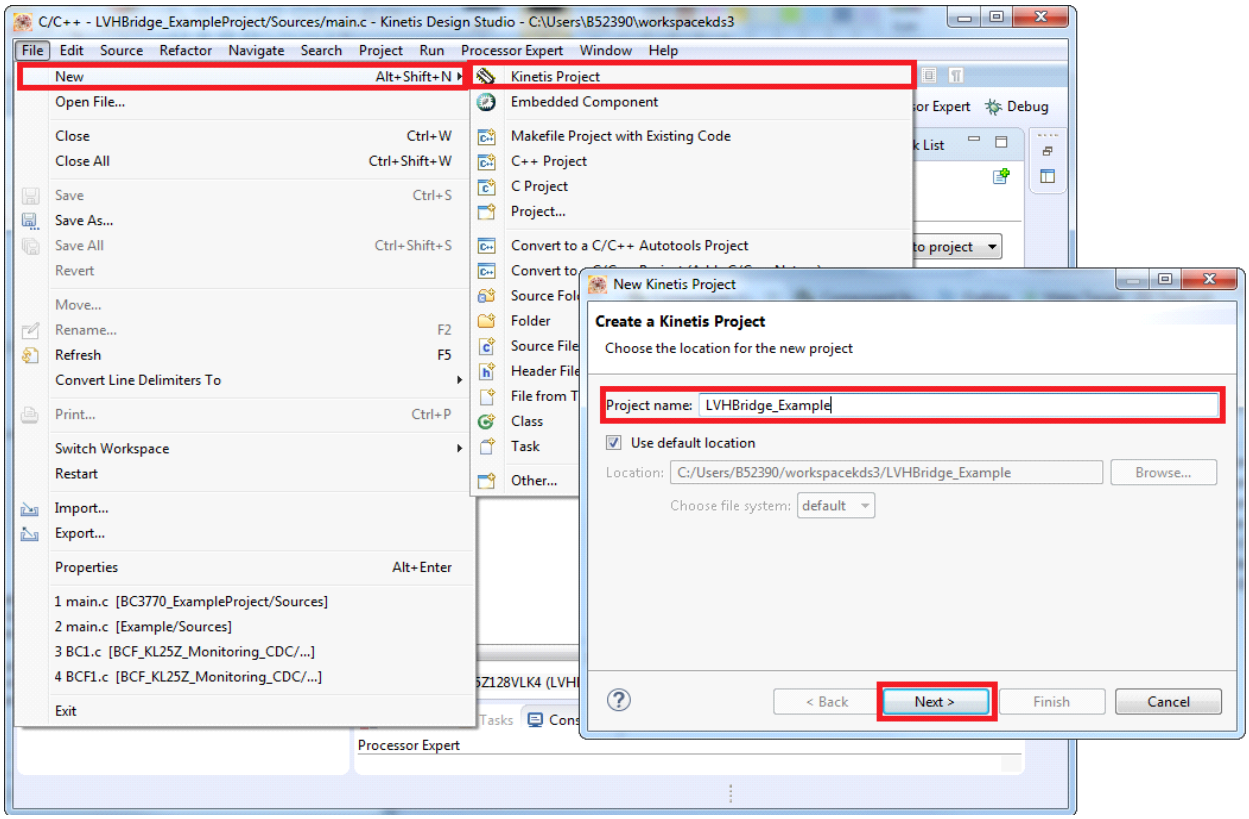

Figure 24. New project creation (a)

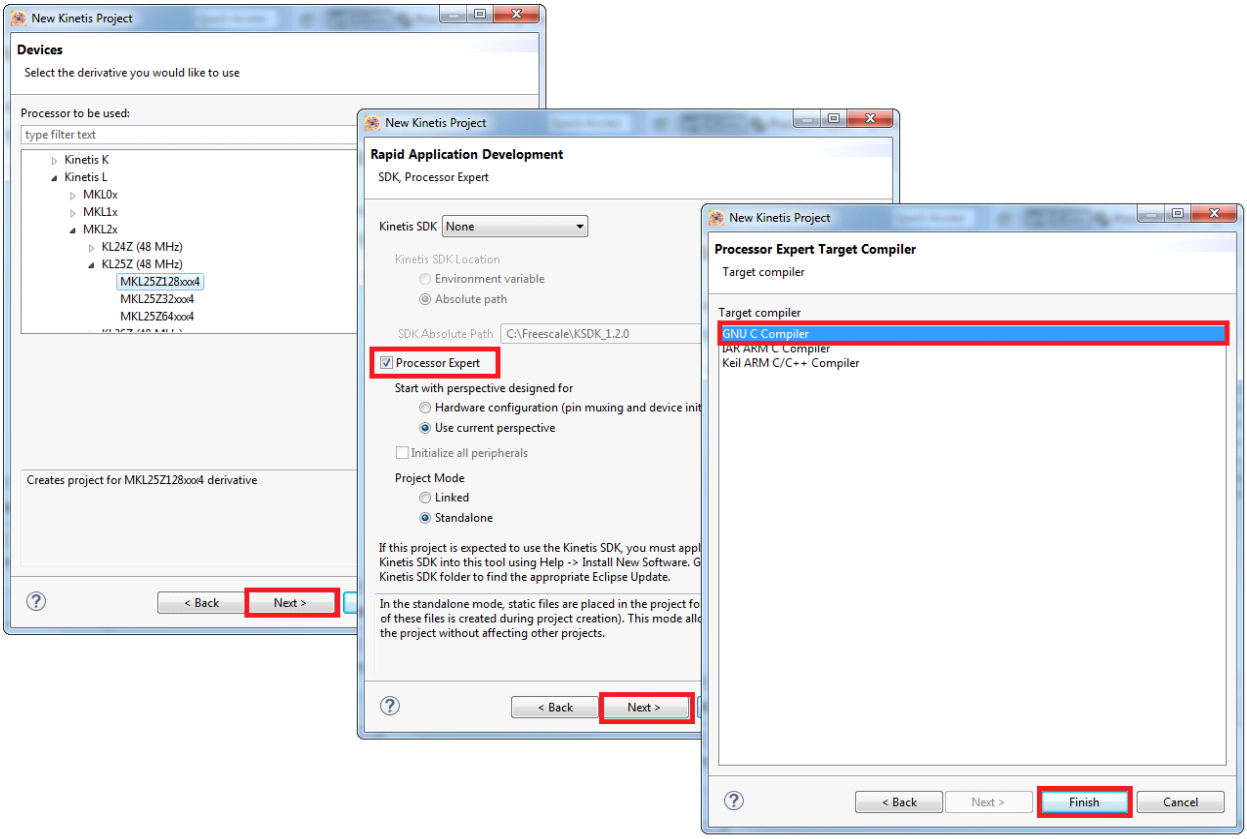

Figure 25. New project creation (b)

4.3.1 Adding the LVHBridge Component into the Project

1. Find LVHBridge in the **Components Library** and add it into the project by double clicking, dragging and dropping, or by right-clicking the mouse button and selecting the **Add to Project** option (see Figure 26).

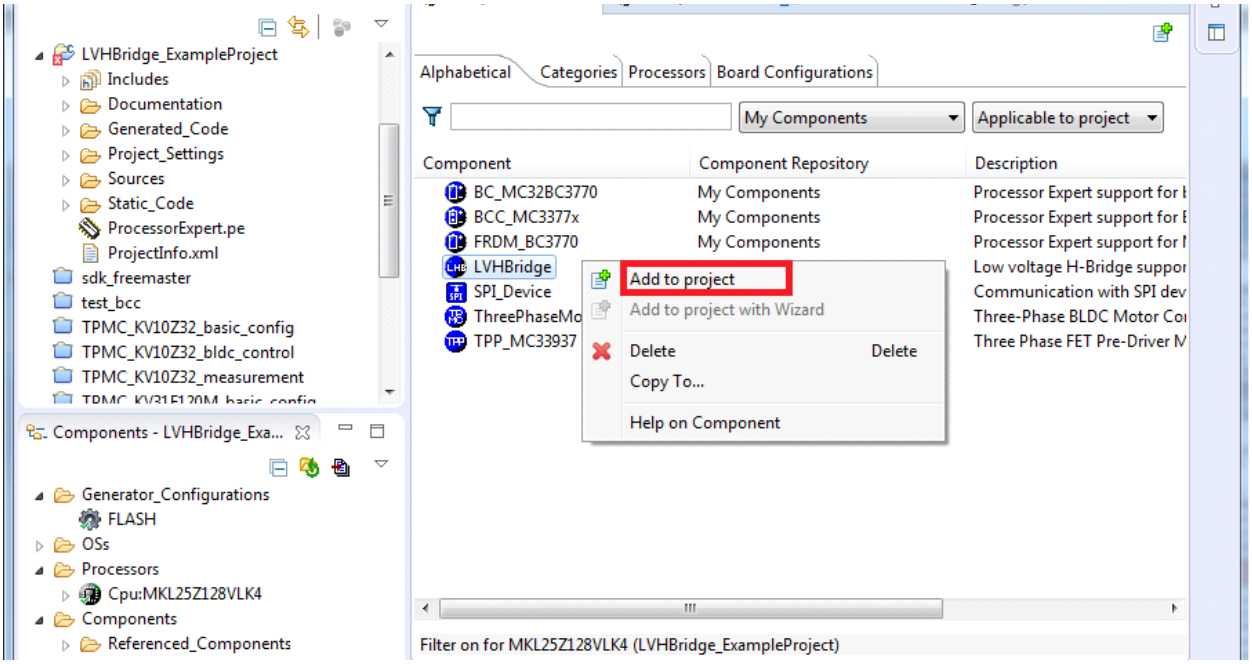

Figure 26. Adding the LVHBridge component to the project

2. Click the LVHBridge component in the **Components** window to show the configuration in the **Component Inspector** view (see Figure 27).

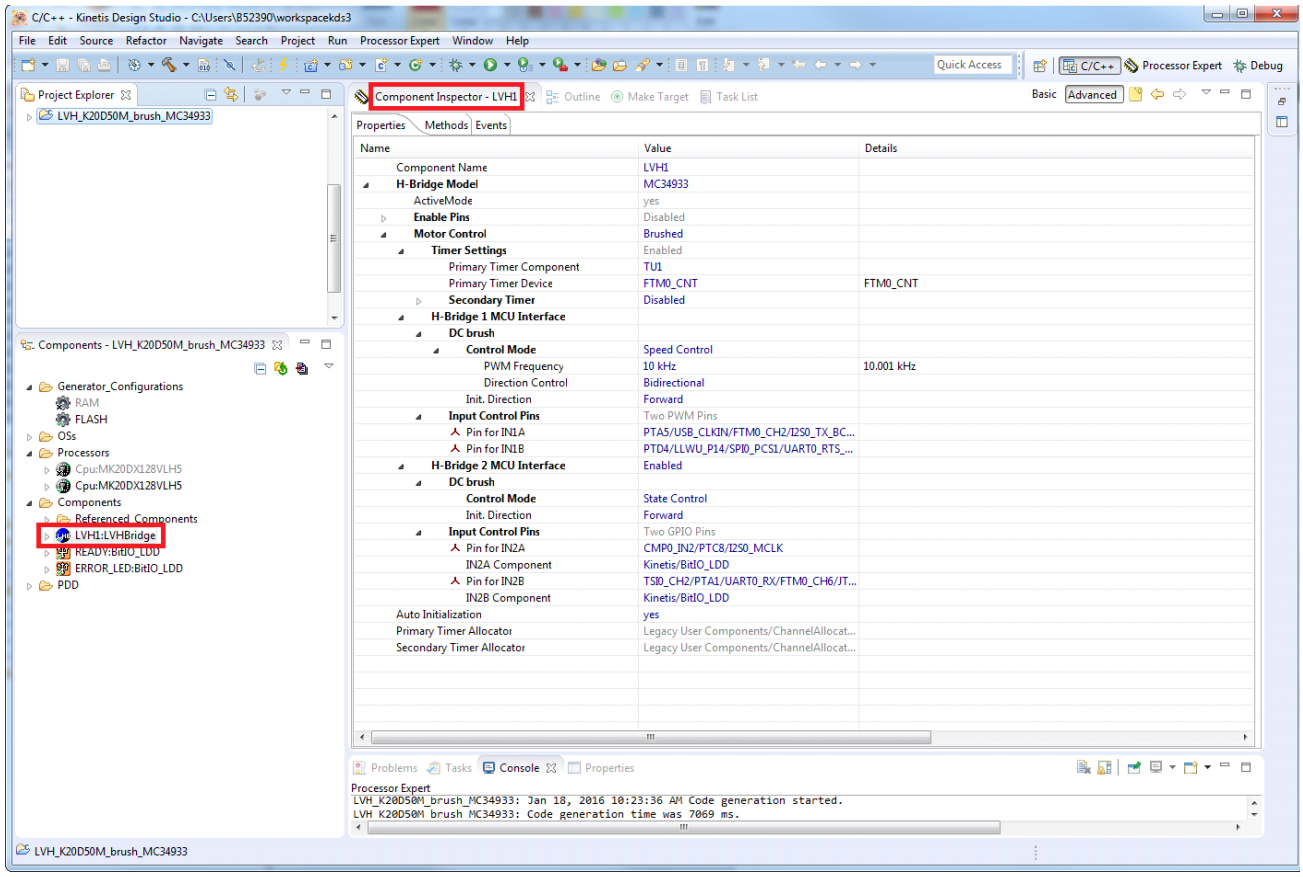

Figure 27. Component Inspector view

4.3.2 Generating Driver Source Code

After the components are configured, the driver code that will be incorporated into the application is ready to be generated. The process is as follows

1. Click on the **Generate Processor Expert Code** icon in the upper right corner of the **Components** panel (see Figure 28).

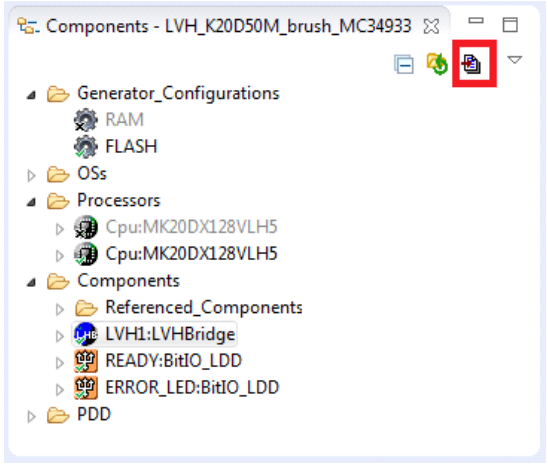

Figure 28. Generating the source code

2. The driver code for the LVHBridge device is generated into the **Generated_Code** folder in the **Project** panel. The component only generates the driver code. It does not generate application code. Figure 29 shows the locations of the generated driver source and the application code.

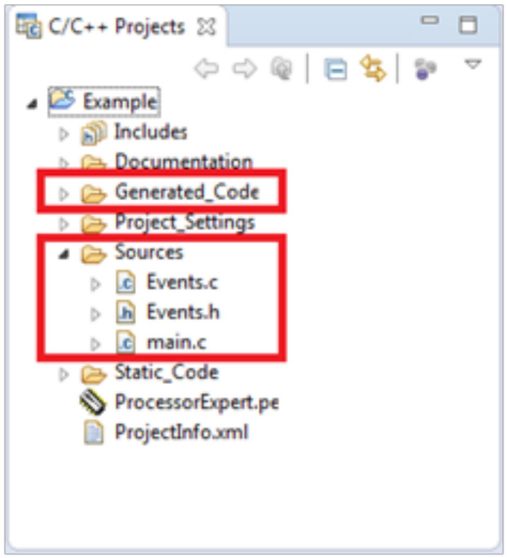

Figure 29. Source code locations

4.3.3 Writing the Application Code

The 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 directions must be retained.

To add a component method into the application source code:

1. In the **Components** panel for project, click on **Components**. 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, the user can open the MVHBridge component method list, drag and drop **RotateProportional** to **main.c** and add the necessary parameters (see Figure 30).

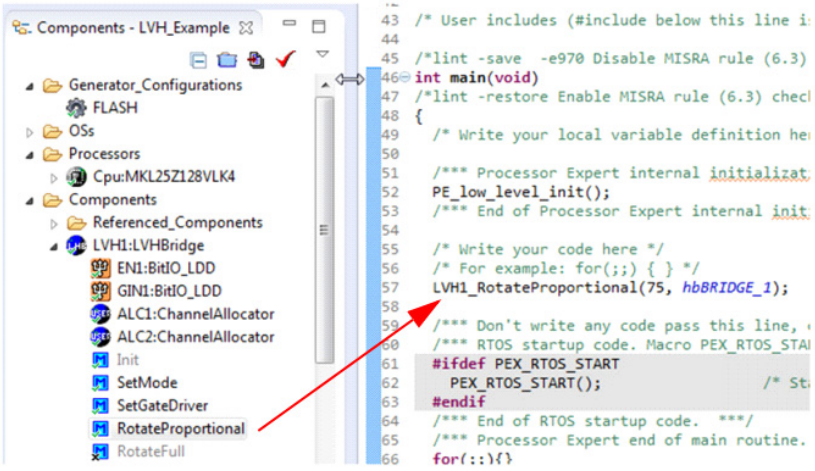

Figure 30. Adding component methods

4.3.4 Compiling, Downloading and Debugging

To compile a project, click the compile icon in the tool bar (see Figure 31).

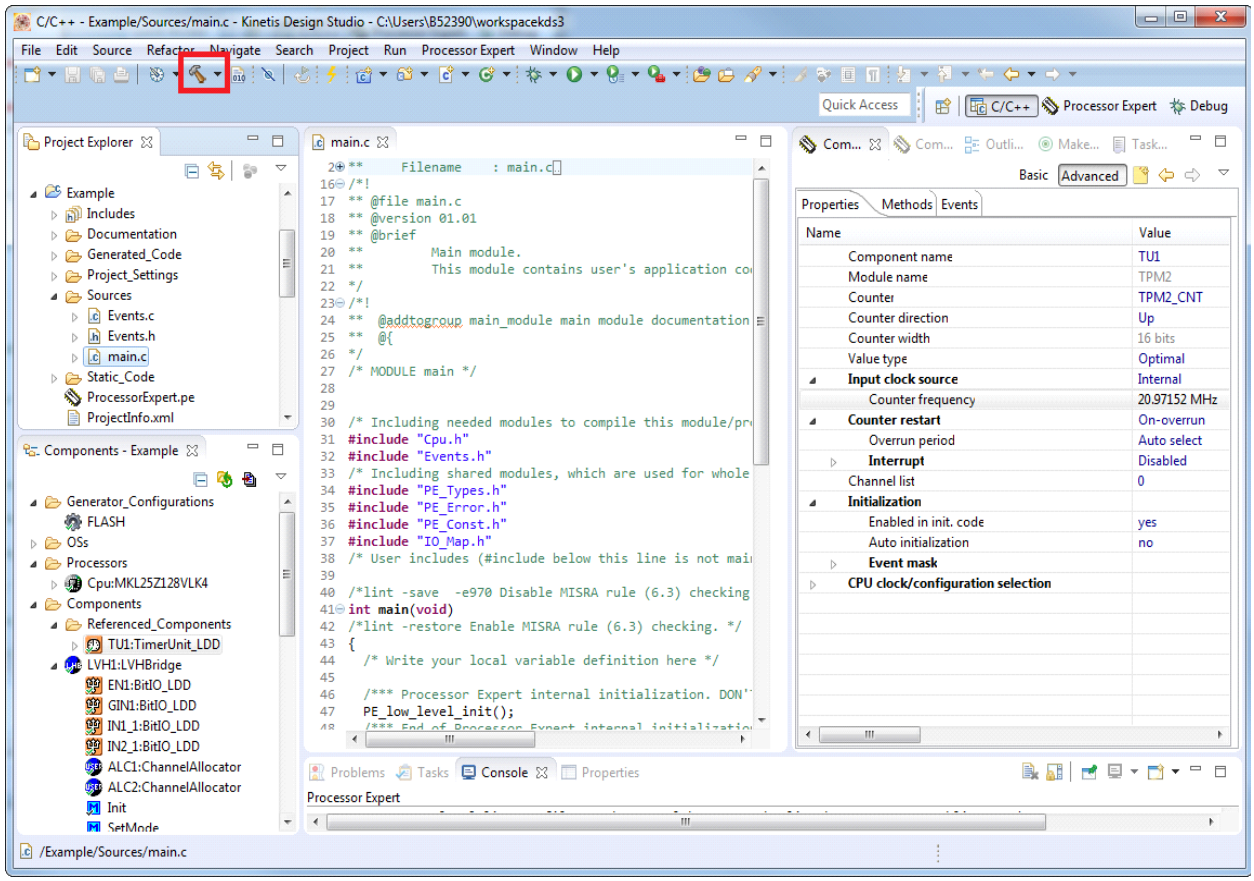

Figure 31. Compiling and downloading the application

The process for downloading an application onto the board in Kinetis Design Studio may differ according to the MCU board being used. For more details, refer to the Kinetis Design Studio user's guide (see Section 5, "References").

To download and debug on KL25Z48M MCU board, do the following:

1. Click the arrow next to the debug icon in the tool bar and select **Debug Configurations…** (see Figure 32).

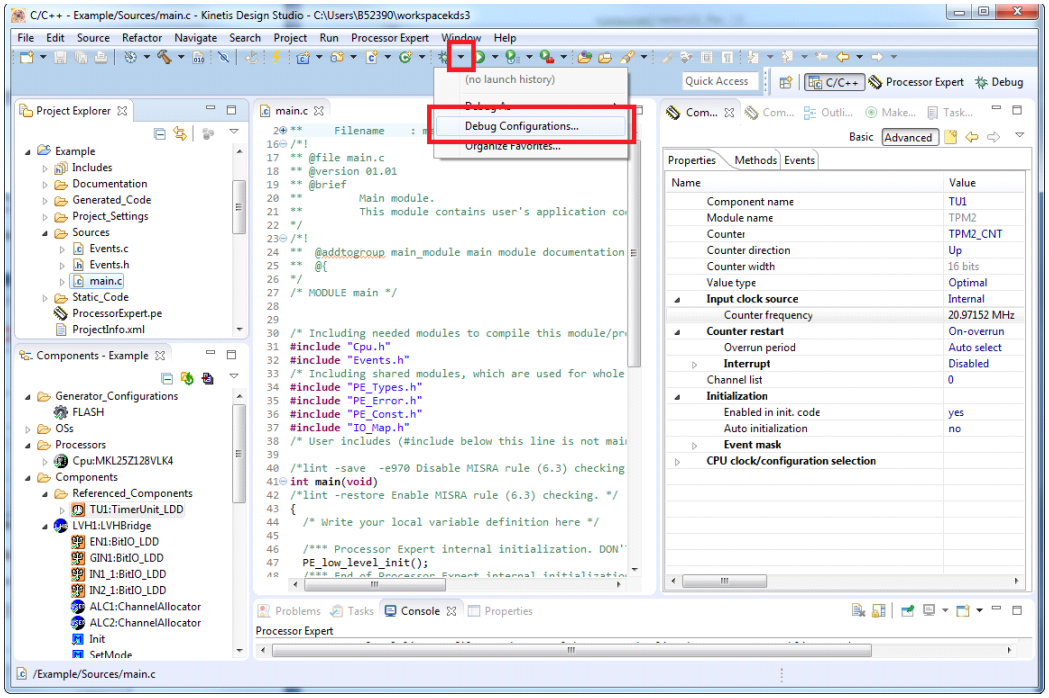

Figure 32. Downloading the application (a)

2. In the **Debug Configurations** dialog box, click **Example_Debug_PNE** under **GDB PEMicro Interface Debugging**.
3. Make sure that **C/C++ Application** contains path to the project's .elf file (see Figure 33).

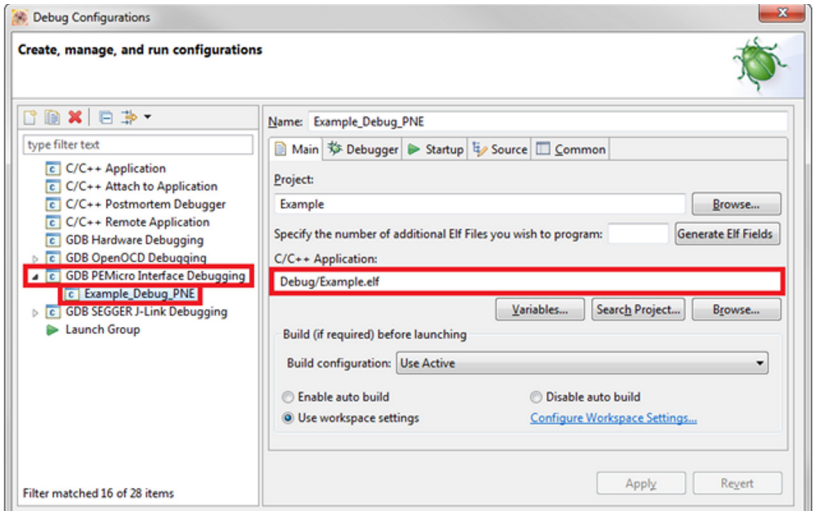

Figure 33. Downloading the application (b)

4. Click the **Debugger** tab and set Interface option to **OpenSDA Embedded Debug - USB Port**. Then click the **Refresh** button next to the **Port** setting to update the list of available USB ports (see Figure 34).
5. Make sure that the **Target** is set to the processor selected earlier. If not, change the target by clicking the **Select Device** button. In the **Select Target Device** dialog box, highlight the appropriate processor and click on the **Select** button.
6. Click the **Debug** button. Kinetis Design Studio downloads and launches the program onto board.

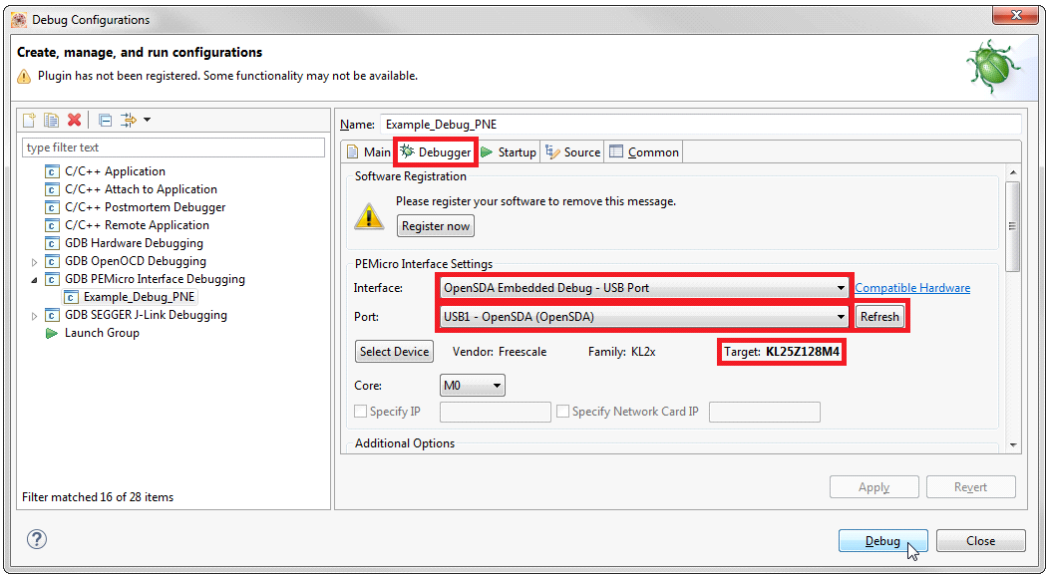

Figure 34. Downloading the application (c)

5 References

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

Table 5. References

NXP.com Support Pages	Description	URL
FRDM-17C724-EVB	Tool Summary Page	www.nxp.com/FRDM-17C724-EVB
FRDM-17510EJ-EVB	Tool Summary Page	www.nxp.com/FRDM-17510EJ-EVB
FRDM-17511EP-EVB	Tool Summary Page	www.nxp.com/FRDM-17511EP-EVB
FRDM-17511EV-EVB	Tool Summary Page	www.nxp.com/FRDM-17511EV-EVB
FRDM-17529EV-EVB	Tool Summary Page	www.nxp.com/FRDM-17529EV-EVB
FRDM-17531EP-EVB	Tool Summary Page	www.nxp.com/FRDM-17531EP-EVB
FRDM-17531EV-EVB	Tool Summary Page	www.nxp.com/FRDM-17531EV-EVB
FRDM-17533EV-EVB	Tool Summary Page	www.nxp.com/FRDM-17533EV-EVB
FRDM-34933EP-EVB	Tool Summary Page	www.nxp.com/FRDM-34933EP-EVB
Kinetis Design Studio	Software	www.nxp.com/kinetis
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	2/2016	• Initial release

How to Reach Us:

Home Page:
NXP.com

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

Document Number: PEXLVHBRIDGESWUG
Rev. 1.0
2/2016