Document Number: SB0410-SB0800SWUG
Rev. 1.0, 1/2016

MC34ValveController Processor Expert Component

Contents

1 Overview

This documentation describes how to install and use Processor Expert in conjunction with the MC34ValveController component.

The MC34ValveController component supports the following analog parts:

- MC34SB0410: Quad Valve Controller System On Chip
- MC34SB0800: Octal Valve Controller System On Chip

The TWR-SB0410-36EVB and TWR-SB0800-36EVB tower boards are evaluation platforms based on these chips. See the related user guides and data sheets for detailed information.

2 MC34ValveController Compatibility

2.1 Peripheral Requirements

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

- **SPI Module** is required for communication (SI, SO, SCLK, CSB)
- **GPIO** or **TPM/FTM** timer (PWM, single channel) are required for direct pump motor pre-driver control (PDI, ADIN1)
- **GPIO** is required for device reset (RSTB)
- **TPM/FTM** timer (periodic interrupts, single channel) is required by the flutter current feature

2.2 Supported Devices

The MC34ValveController supports the following devices:

MC34SB0410

- One pump motor pre-driver (up to 16 kHz)
- Four low-side drivers for inductive loads. Channel 1 to 4 serve as current regulators (with PI regulator) or as PWMs (Pulse Width Modulators)
- Two low-side drivers for resistive loads

MC34SB0800

- One high-side driver (to control the fail-safe switch for overall solenoid path)
- One pump motor pre-driver (up to 500 Hz)
- Eight low-side drivers to drives inductive loads. Channel 1 to 4 serve as current regulators (with PI regulator) or as PWMs (Pulse Width Modulators)
- One low-side driver for resistive loads
- One high-side driver for general purpose usage

2.3 Supported MCUs

The MC34ValveController supports the MCUs listed in Table 1. The listed MCUs are a subset of MCUs supported by Processor Expert for Kinetis using the logic device driver (LDD) layer.

Table 1. Supported MCUs

Supported MCUs	CodeWarrior Support	Kinetis Design System Support
TWR-KL25Z48M	Yes	Yes
TWR-KV31F120M	No	Yes
TWR-KV10Z32	Yes	Yes
TWR-K64F120M	Yes	Yes
TWR-K20D72M	Yes	Yes
TWR-K22F120	Yes	Yes
TWR-K70	Yes	Yes

See Table 2 for pin compatibility between Valve Controller tower boards and selected MCUs.

Table 2. Pin Compatibility of Valve Controller Tower Boards with Selected MCUs

Pin Function	TWR-KL25Z48M (1)	TWR-KV31F120M (1)	TWR-KV10Z32 (2)	TWR-K64F120M (2)	TWR-K20 (1)	TWR-K22F120 (2)	TWR-K70 (2)
RSTB	PTC7	PTB2	PTB0	PTE11	PTB22	PTE3	PTE0
MISO	PTD3	PTE19	PTD3	PTD3	PTD3	PTD3	PTD14
MOSI	PTD2	PTE18	PTD2	PTD2	PTD2	PTD2	PTD13
CSB	PTE0	PTC0	PTE24	PTE0	PTC9	PTE25	PTE28
SCLK	PTD1	PTE17	PTC5	PTD1	PTD1	PTD1	PTD12
ADIN1/PDI	PTD4	PTD4	PTD4	PTA7	PTD4	PTC1	PTA7

Notes
1. Example provided for this MC. See Section 4.2, Downloading the Components and Example Projects, page 14
2. Create a new project based on an existing project and change the pin selection accordingly.

2.4 Tower Board Settings

Jumper blocks on the TWR-SB0800-36EVB and the TWR-SB0410-36EVB provide a means of configuring the boards for use with additional MCUs. Jumper settings on the blocks define the routing of chip select SPI signals, the reset signal from the MCU, and the motor control signal which can either be simple GPIO (low, high) or PWM. On both the TWR-SB0800-36EVB and the TWR-SB0410-36EVB, this jumper block is labelled J13.

In addition, jumper J10 on the TWR-SB0800-36EVB and J6 on the TWR-SB0410-36EVB selects between 3.3 V and 5.0 V depending on the requirement of the MCU being used.

Make sure to set jumper J10 or J6 to the proper voltage level and set the jumpers on J13 to the appropriate positions for the selected MCU. Check the schematic of each tower elevator board to assure all signals are correctly connected. Figure 1 shows the selection options on the TWR-SB0800-36EVB and the TWR-SB0410-36EVB.

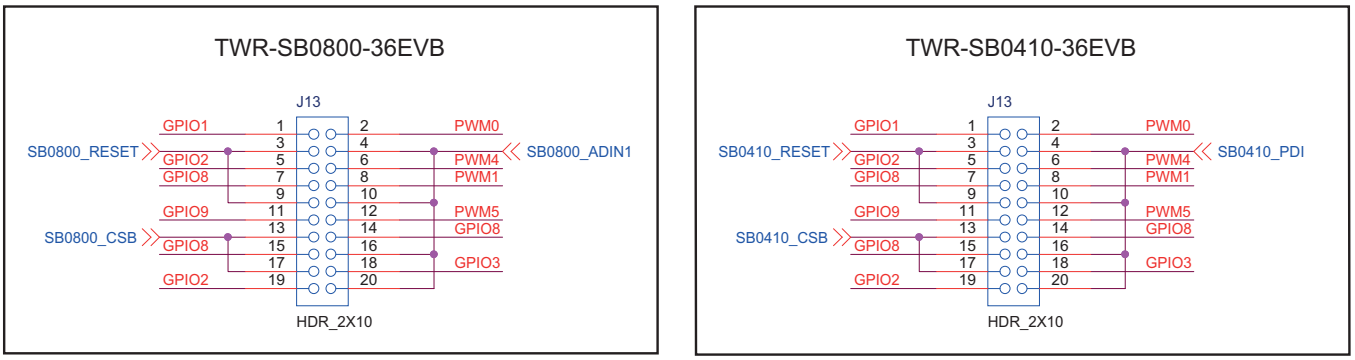

Figure 1. Jumpers for IO Selection

Table 3 shows appropriate J13 jumper settings for compatible tower boards. These settings are important because the **Reset** (RSTB) and **Chip Select** (CSB) signals must be routed to MCU IO header positions capable of handling such signals. Note that the **ADIN1** pin on the MS34SB0800 can be used either to directly control the **Pump Motor Pre-Driver** or to measure external voltage. The PDI pin controls the **Pump Motor Pre-Driver** on the MC34SB0410.

Table 3. MCU Tower Board Jumper Selection

	TWR-KL25Z48M	TWR-KV31F120M	TWR-KV10Z32	TWR-K64F120M	TWR-K20	TWR-K22F120	TWR-K70
RSTB	GPIO1	GPIO1	GPIO1	GPIO1	GPIO8	GPIO1	GPIO2
CSB	GPIO2	GPIO2	GPIO2	GPIO2	GPIO9	GPIO2	GPIO3
ADIN1	PWM4	PWM4	PWM4	PWM4	PWM4	PWM0	PWM4

3 MC34ValveController Component

The MC34ValveController is located under the Components folder in the active projects window (see Figure 2). This folder contains two sub-folders: Referenced_Components (containing components to configure SPI communication properties) and VC1:MC34ValveController (containing the component to configure MC34SB0800 and MC34SB0410 features). The functionality of the MC34ValveController depends on the component property settings assigned through Processor Expert.

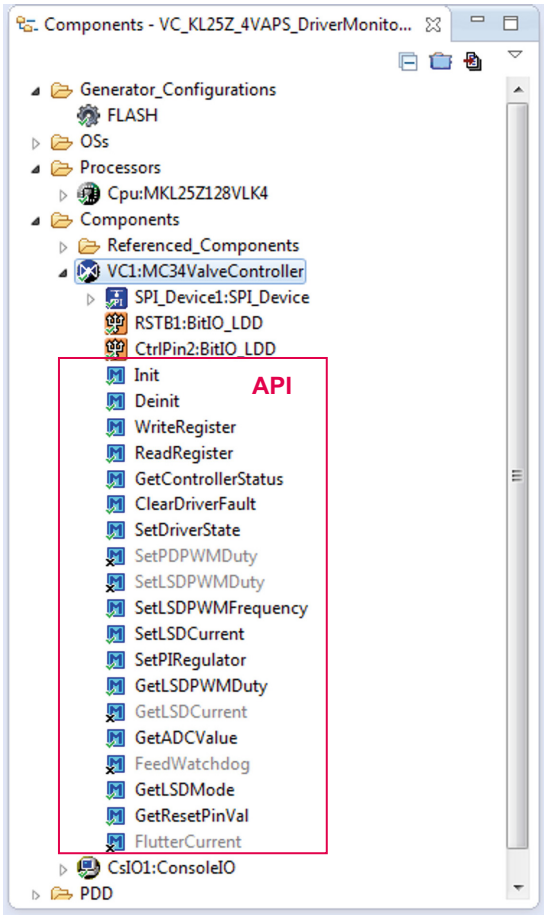

Figure 2. MC34ValveController Processor Expert Component

The component also offers Help documentation, which can be accessed by right-clicking on the MC34ValveController in the component tree. The **Help on Component** window provides information on all properties and methods of the component. Access the **Typical Usage** section to view examples showing how to work with API methods.

3.1 Component Settings

Selecting the MC34ValveController component in the component tree gains access to properties in the Component Inspector. 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.

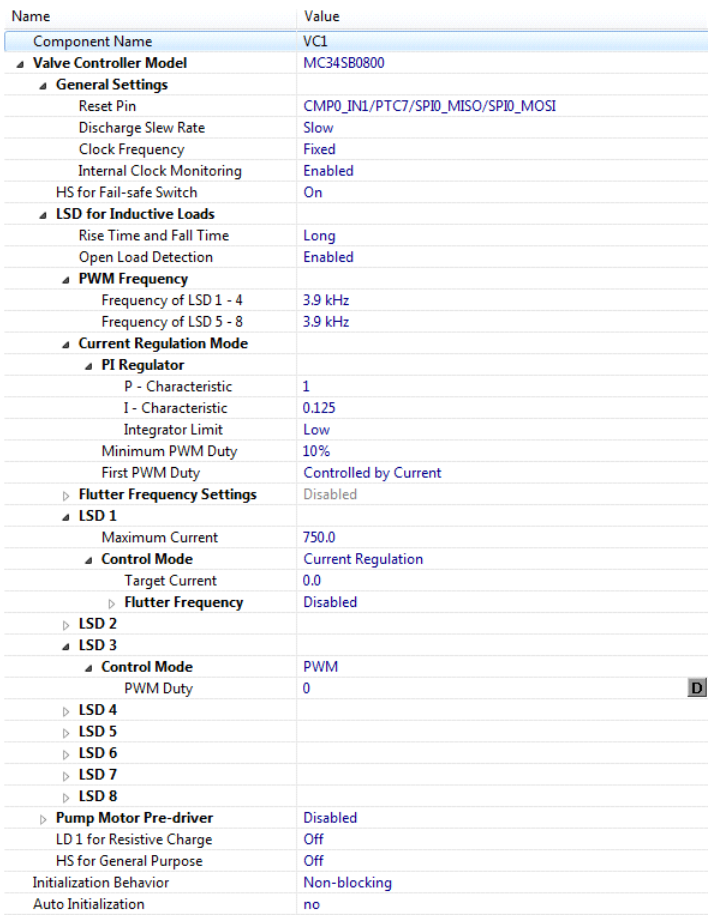

Figure 3. Valve Controller Component Properties

Valve Controller Model (MC34SB0410 / MC34SB0800) - This component supports two models of the Valve Controller. Both have much in common, but differ in number and type of provided drivers. Therefore model selection affects which properties are available/hidden, or enabled/disabled.

General Settings

- **Reset Pin** - This pin has dual functionality. It can be used for an explicit reset of the device, in which case it works as an output. Alternatively, it can be used as a fault indication in which case it works as an input. The direction of this pin is automatically handled by the component.
- **Discharge Slew Rate** - The slew rate used by the pump motor pre-driver and high-side driver for general purpose modules. When the power FET is switched off, the gate capacitance of the FET is discharged by a constant current that is controlled as either fast (typically 2.0 mA) or slow (typically 100 μA). This feature is available for MC34SB0800 only.
- **Clock Frequency** - The frequency of the clock modules, i.e. the main supply clock (CLK1) and the auxiliary clock (CLK2). Clock frequency is 14 MHz when the fixed option is selected. Otherwise frequency modulation is used. Two deviation frequencies (350 kHz and 700 kHz) are available to spread the oscillator energy over a wide frequency band.
- **Internal Clock Monitoring** - The internal clock monitoring function (i.e. enable or disable CLK2). Valve Controller utilizes two clocks: the main supply clock (CLK1) and the auxiliary clock (CLK2). CLK2 monitors main clock faults and resets the controller (using the **RST_CLK** function) when a fault is detected. Disabling CLK2 has no effect on other functionality (except for the clock monitoring function) because the main clock (CLK1) remains active.

HS for Fail-safe Switch - The initial state of the high-side driver intended to control the fail-safe switch for the overall solenoid path. (Available on the MC34SB0800 only)

LSD for Inductive Loads

- **Rise Time and Fall Time** - The rise time and fall time of the low-side drivers. Long rise and fall times are typically 1.7 μs (rise) and 1.35 μs (fall). Short rise and fall times are typically 0.5 μs (rise) and 1.0 μs (fall).
- **Open Load Detection** - Enables or disables sink current for open load detection

PWM Frequency

- **Frequency of LSD 1—4** - From 3.0 kHz to 5.0 kHz.
- **Frequency of LSD 5—8** - This setting is available for the MC34SB0800 only.

Current Regulation Mode - The current regulation mode setting for low-side drivers. Load current is sensed by the internal low-side sense FET and digitized by the internal A/D converter. The digital current regulation circuitry compares the actual load current with the target current value and steers the low-side power switch duty cycle. The PI regulator is used.

- **PI Regulator**
  - **P - Characteristic** - The proportional characteristic of the PI regulator.
  - **I - Characteristic** - The integral characteristic of the PI regulator. The regulator stays idle until a non-zero value is applied.
  - **Integrator Limit** - The set integrator limit. Possible values are low (1023) and high (2047).
- **Minimum PWM Duty** - The minimum duty cycle of the low-side driver (1 to 4) outputs. This option applies to the time interval during which the current measurement occurs. Note that the maximum duty cycle is 100%.
- **First PWM Cycle** - The first duty cycle of the low-side driver (1 to 4) outputs. The first duty cycle is either controlled by current or limited to a fixed duty cycle in which the target current is transformed.

Flutter Frequency Settings - The setting of the flutter current function, which influences mechanical friction inside the valve. The goal is to achieve a more precise movement. When this function is used, current is changed periodically around the target current. This results in the current varying as a sine wave. This property is enabled automatically by the component when enabling the flutter current feature for one or more LSDs. See Section 3.6, Flutter Current Feature, page 12.

- **Timing Device** - The name of the timing device used by **TimerInt_LDD** component.
- **Control Mode** - The control mode for the flutter current function. **Auto** means the process is automatically handled internally. **Polling** means the user application code must call handler functions to achieve the desired behavior.

LSD 1 - 4 - The common settings for current regulation or PWM mode

- **Maximum Current** - The current limitation value in mA. This value is used in component methods.
- **Control Mode** - Current Regulation/PWM) - Sets the low-side driver mode. Either current regulation or PWM mode can be enabled in time.
- **Target Current -** Target current in mA. Minimum value is 0 mA, maximum 2250.6 mA and step is 2.2 mA. The value is rounded to the nearest available value.
- **Flutter Frequency** - Enables or disables flutter current feature for selected the LSD.
  - **Frequency** - The frequency of the sinusoidal current curve in Hz. When the flutter current function is used by two or more LSDs, the frequency value is corrected, because it must fit the interrupt frequency given by the LSD with the maximum flutter frequency.
  - **Points per Period -** The number of points per period. Single point corresponds to the deviation of current given by the actual position on the sinusoidal curve.
  - **Amplitude** - The amplitude of the sinusoidal curve in mA. This value defines the maximum variation from the target current. The admissible range is from 5.0 mA to 100 mA.
- **PWM Duty** - The target PWM duty cycle. The admissible range is from 0 to 255. Representing 0% respective 100% duty value.

LSD 5 - 8 - The common settings (only PWM mode) - These drivers are available on MC34SB0800 only.

- **PWM Duty** - The target PWM duty cycle. The admissible range is from 0 to 255. Representing 0% respective 100% duty value.

Pump Motor Pre-driver - The settings of the Pump Motor Pre-driver.

- **Overcurrent Masking Time** - The masking time from the direct input turn-on against the malfunction on transient time. This masking time is used by the over-current detection logic. Possible values depend on the selected valve controller model.
- **Overcurrent Filter Time** - The overcurrent filter time of the pump motor pre-driver. The drain-source voltage of the FET on **PD_G** is checked when the pre-driver is switched on. If the measured voltage exceeds the overcurrent voltage threshold, output of the comparator is enabled. If the output of the comparator is active longer than the defined filter time, **PD_G** is turned-off. For the MC34SB0410, the filter time has a fixed value. This setting is available for MC34SB0800 only.
- **Control Mode** - SPI/Direct) - The pre-driver can be driven either directly by the MCU pin or through the SPI interface.
  - **Initial State** - The initial state of the driver output in SPI control mode.
  - **Input Control** - GPIO/PWM) - The type of direct control mode. The possible values are GPIO (general purpose input/output pin) or PWM (pulse width modulated).

- **Control Pin** - The pin for direct control. This pin is called PDI for MC34SB0410 and **ADIN1** for MC34SB0800. Note that if **ADIN1** pin is used for direct control, it cannot be used to measure the external voltage simultaneously. To use **ADIN1** for measurement, set the **Pump Motor Pre-driver Control Mode** to the SPI.
- **PWM Frequency** - The PWM frequency. The maximum value depends on the valve controller model (16 kHz for MC34SB0410, 500 Hz for MC34SB0800).
- **PWM Duty** - The target PWM duty cycle. The admissible range is from 0 to 255. Representing 0% respective 100% duty value.
- **Initial State** - The initial state of driver output in direct control mode.

- **LD 1 for Resistive Charge** - The initial state of the low-side driver 1 for a resistive charge.
- **LD 2 for Resistive Charge** - The initial state of the low-side driver 2 for a resistive charge. This setting is available for the MC34SB0410 only.

HS for General Purpose - The initial state of the general purpose high-side driver. This setting is available for the MC34SB0800 only.
Initialization Behavior - Defines the behavior of the Init method. This method selects between internally blocking and unblocking while waiting on the **Reset** pin to clear. If blocking version is selected, the method may hang if an error fails to clear. If the unblocking version is selected, the method uses a timeout functionality to avoid infinite waiting.
Auto Initialization - Selects whether component initialization should be automatically called from the CPU component initialization function **PE_low_level_init** or whether the user is responsible for calling the initialization method.

3.2 SPI Configuration

The Valve Controller uses the SPI communication protocol to communicate with the MCU. This protocol is implemented by the **SPIMaster_LDD** component which can be found in the referenced components (shared components) folder in the Components panel (see Figure 2). However, this component does not handle arbitration for simultaneous communication requests on the SPI bus. This functionality is implemented by the **SPI_Device** component, which is exclusively inherited by the **MC34ValveController** component.

In **SPIMaster_LDD**, the (**MISO**, **MOSI**, **CLK**) pins and timing settings must be set according to MC34SB0410/MC34SB0800 data sheet recommendations. The maximum admissible communication frequency is 10 MHz.

The **CSB** (chip select) pin has to be set separately in the **BitIO_LDD** component exclusively inherited by **SPI_Device**. Because of component implementation limitations, the user must initialize the **CSB** pin value to 1 as specified in the data sheet.

Name	Value
Device	SPI0
Interrupt service/event	Enabled
Settings	
Input pin	Enabled
Pin	PTD3/SPI0_MISO/UART2_TX/TPM0_CH3/SPI0_MOSI
Output pin	Enabled
Pin	PTD2/SPI0_MOSI/UART2_RX/TPM0_CH2/SPI0_MISO
Clock pin	
Pin	ADC0_SE5b/PTD1/SPI0_SCK/TPM0_CH1
Chip select list	0
Attribute set list	1
Attribute set 0	
Width	8 bits
MSB first	yes
Clock polarity	Low
Clock phase	Capture on leading edge
Parity	None
Chip select toggling	yes
Clock rate index	0
Clock rate	5.24288 MHz
Initialization	
Auto initialization	yes

Figure 4. SPI Configuration

3.3 Component API

The Valve Controller component provides API functions allowing the application code to dynamically configure a device in real-time. The available methods and events can be viewed by clicking to expand the component in the Component folder of the Components Panel (see Figure 2).

Some of those methods/events are marked with ticks and others with crosses, which distinguishes which methods/events are supposed to be generated. Change this setting in the Processor Expert Inspector. Note that methods with grey text are always generated because they are needed for proper functionality. This forced behavior depends on various combinations of component property settings. For summarization of available API methods and events and their descriptions, see Table 4.

Table 4. MC34ValveController Component API

Method	Description
Init	Initializes the device and applies settings selected in the component properties. This includes initialization of inherited components and other features.
Deinit	Deinitializes the device. Sets the reset pin to low and consequently clears all registers of device.
WriteRegister	Writes a value to the selected register. Allocates the SPI bus and calls the internal function VC_write_register.
ReadRegister	Reads a value from the selected register. Allocates the SPI bus and calls the internal function VC_read_register.
GetControllerStatus	Gets status information, reads two registers with related information.
ClearDriverFault	Clears selected fault flags. Handles only faults related to the driver's modules. It is not intended to clear supervision module faults.
SetDriverState	Sets the selected driver output state. Internally handles the driver either through SPI communication or directly by output of the MCU.
SetPDPWMDuty	Sets the PWM duty cycle for the pump motor pre-driver when the direct control mode is used.
SetLSDPWMDuty	Sets the PWM duty cycle for the selected low-side driver for inductive loads. LSD has to be in PWM mode.
SetLSDPWMFrequency	Sets the PWM frequency for the selected group of low-side drivers for inductive loads.
SetLSDCurrent	Sets the target current for the selected low-side driver for inductive loads. LSD has to be in current regulation mode.
SetPIRegulator	Sets parameters of HW PI regulator used for LSDs in current regulation mode.
GetLSDPWMDuty	Gets PWM duty cycle of selected LSD, which has to be in current regulation mode.
GetLSDCurrent	Gets measured current of selected LSD, which has to be in PWM mode.
GetADCValue	Reads and interprets ADC value of selected measured item (temperature, voltage) by device.
FeedWatchdog	Feeds watchdog. Sends MCU monitoring result computed for LFSR output received from device.
GetLSDMode	Gets mode (current regulation, PWM) for selected LSD.
GetResetPinVal	Gets level of reset pin. Low level means that device is in fault state.
FlutterCurrent	Checks whether to adjust target current of LSDs with enabled flutter current feature according to predefined settings.

3.4 MC34ValveController Components

The MC34ValveController consists of the valve controller component, which allows configuring MC34SB0800 and MC34B0410 capabilities, and a set of referenced components, which configure SPI communication functions. Figure 5 illustrates these components and their relationship to each other. A description of the inherited and referenced components used by the MC34ValveController appears immediately below Figure 5. The functionality of the MC34ValveController in terms of communication, control, etc. depends on these components.

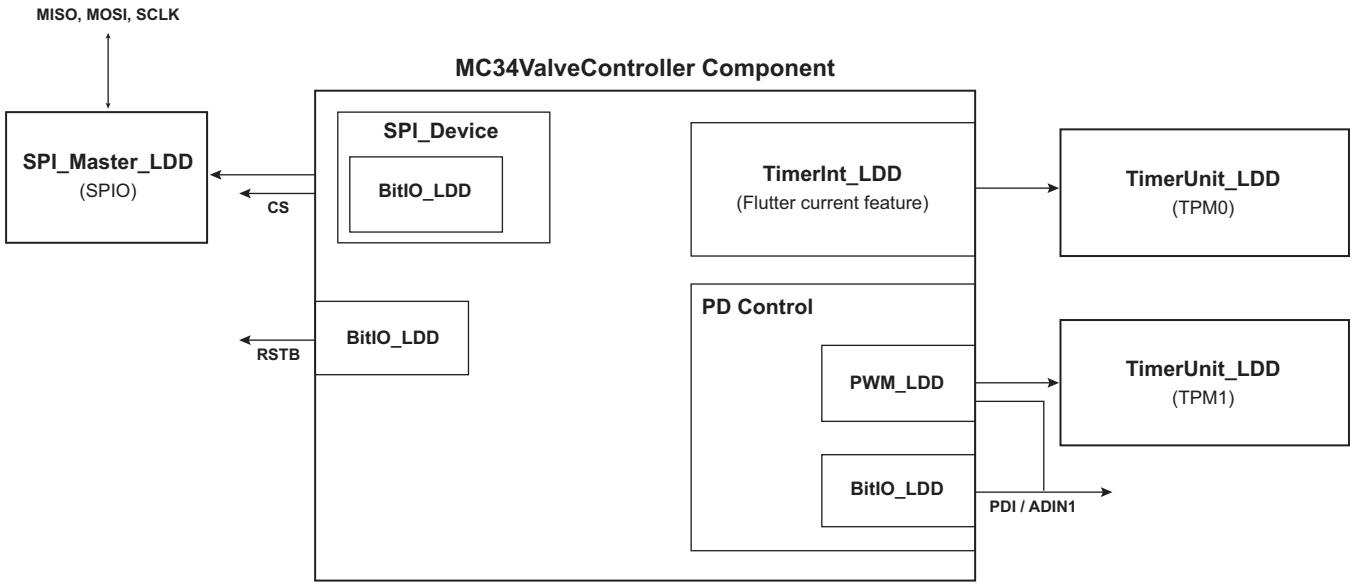

Figure 5. Components used by the MC34ValveController

Referenced components

SM1:SPIMaster_LDD	Configuration of SPI communication—Referenced by **SPI_Device**
TU1:TimerUnit_LDD	Referenced by **FlutterFreq1:TimerInt_LDD**
TU2:TimerUnit_LDD	Referenced by **CtrlPin1:PWM_LDD**

VC1:MC34ValveController components

SPI_Device1:SPI_Device	Adds bus allocation of SPI communication
CSPin1:BitIO_LDD	Software chip select
RTSB1:BitIO_LDD	Input/output reset pin
FlutterFreq1:TimerInt_LDD	Flutter current feature, periodic interrupts
CtrlPin1:PWM_LDD or **CtrlPin1:BitIO_LDD**	Direct control of pump motor pre-driver. Either PWM or on/off logic

3.5 Fault Detection and Handling

The Valve Controller component provides methods allowing application code to read device status information and react to faults. Table 5 lists these methods and their functionality.

Table 5. Valve Controller Methods

Method	Function
VC_GetControllerStatus()	Reads status information related to drivers.
VC_ClearDriverFault()	Clears faults related to the driver module.
VC_GetGetResetPinVal()	Reads fault information provided by the supervision module.
VC_GetControllerStatus()	
VC_Init()	Recovers from a fault that caused a register reset and sets reset pin to low.

3.6 Flutter Current Feature

Under constant current conditions, mechanical friction inside a valve may result in movement less precise than expected. The Flutter Current feature helps smooth out valve movement by introducing periodic minor deviations from the target current. These deviations are both above and below the target so the overall current average matches the target current.

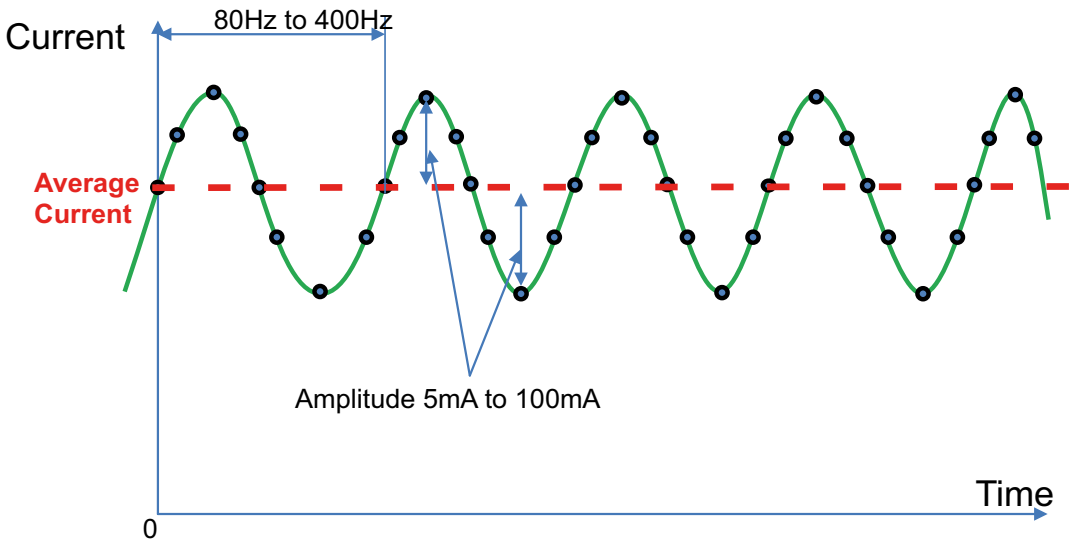

Figure 6. Flutter Current Function

As Figure 6 shows, the parameters of this function are frequency (of the base sinusoidal wave), resolution (the number of current changes per period) and amplitude (the maximum current deviation). The flutter current parameters have the following limitations and admissible ranges:

Frequency - 80 Hz to 400 Hz
Resolution - 8.0 to 48 points (with step 4)
Amplitude - 5.0 mA to 100 mA

Notice that the final current change frequency equals the sinusoidal curve frequency multiplied by the selected resolution.

To efficiently manage interrupt resources, only a single timer interrupt is used for all LSDs when the flutter current function is enabled. This introduces a dependency among the LSD flutter frequencies in which the sum of all flutter frequencies must equal the maximum flutter frequency. This restriction is enforced by component logic adjusting frequencies accordingly. Table 6 shows the specified ranges of parameters in terms of the timer interrupt requirements.

Table 6. Requirements on MCU Interrupts

Frequency (Hz)	Interrupt Frequency (Interrupts per Second)					
	8 Points	16 Points	24 Points	32 Points	40 points	48 Points
80	640	1280	1920	2560	3200	3840
100	800	1600	2400	3200	4000	4800
120	960	1920	2880	3840	4800	5760
140	1120	2240	3360	4480	5600	6720
160	1280	2560	3840	5120	6400	7680
180	1440	2880	4320	5760	7200	8640
200	1600	3200	4800	6400	8000	9600
220	1760	3520	5280	7040	8800	10560
240	1920	3840	5760	7680	9600	11520
260	2080	4160	6240	8320	10400	12480
280	2240	4480	6720	8960	11200	13440
300	2400	4800	7200	9600	12000	14400
320	2560	5120	7680	10240	12800	15360
340	2720	5440	8160	10880	13600	16320
360	2880	5760	8640	11520	14400	17280
380	3040	6080	9120	12160	15200	18240
400	3200	6400	9600	12800	16000	19200

There are two ways to implement this feature in your application code:

1. Select **Auto** as the value for the **Control Mode** property under **Flutter Frequency**, in which case all of the flutter current functions are handled automatically. With **Auto** selected, current changes take place directly in the internal interrupt routine. This solution is considered more precise in terms of current change timing because the code is executed with the priority of a raised interrupt routine.
2. Select **Polling** as the value for the **Control Mode** property under **Flutter Frequency**, in which case the application code is responsible for checking whether it is necessary to enable the flutter current functions. With **Polling** selected, an internal interrupt routine raises a flag indicating the maximum final flutter current frequency. Application code must continually call the **FlutterCurrent** API Method to check the status of this flag and must call the internal flutter current feature handler when needed. This solution is considered less precise in terms of current change timing because the code executes with user application code priority and therefore may occasionally be interrupted.

Note that the precision and performance of this function depends on the frequency of SPI communication and the CPU clock.

3.7 Known Issues

The MC34ValveController component has following issues which must be taken in consideration before usage.

1. The Flutter Current (Flutter Frequency) function cannot share the **TimerUnit_LDD** component (a timer) with the pump motor pre-driver when the driver is in PWM mode.

4 Installing the Processor Expert Software

This chapter describes the installation of Kinetis Design Studio and the 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 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 3.0.0 in this guide). The procedure for CodeWarrior installation is very similar.

NOTE

The component and some examples in the component package are intended for CodeWarrior 10.6 (or above) and Kinetis Design Studio 3.0.0 (and above). If CodeWarrior 10.6 and Kinetis Design Studio 3.0.0 are already installed on the system, skip this section.

1. Obtain the latest Kinetis Design Studio 3.0.0 installer file from the Freescale 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
2. Run the executable file and follow the instructions.

4.2 Downloading the Components and Example Projects

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

1. Go to the NXP website www.nxp.com/MC34VALVECONTROLLER-PEXPERT
2. Download the zip file containing components and example projects.
3. Unzip the downloaded file and check to see that the folder contains the files listed in Table 7.

Table 7. MC34ValveController Example Project and Components

Folder Name	Folder Contents
Components	
MC34ValveController_b160104.PEupd	This component configures MC34SB0800 and MC34SB0410 features.
SPI_Device_b1401.PEupd	This component configures SPI communication properties.
CodeWarrior Examples	
VC_KL25Z_4VAPS_DriverControl	This project demonstrates the use of the MC34ValveController component in conjunction with MC34SB0410 valve controller drivers. The target MCU is the TWR-KL25Z48M.
VC_KL25Z_4VAPS_DriverMonitoring	This project demonstrates how to monitor status of the MC34SB0410 valve controller using MC34ValveCotontroller component. The target is the TWR-KL25Z48M MCU board.
VC_KL25Z_4VAPS_FlutterCurrent	This project shows how to use Flutter Current function to control low-side drivers for inductive loads using MC34ValveController component. The targets are the MC34SB0410 device and the TWR-KL25Z48M MCU board.
VC_KL25Z_8VAPS_DriverControl	This project shows how to work with drivers of MC34SB0800 valve controller using MC34ValveController component. The target MCU is the TWR-KL25Z48M.
VC_KL25Z_8VAPS_DriverMonitoring	This project shows how to use the MC34ValveController component to monitor MC34SB0800 valve controller status. The target is the TWR-KL25Z48M MCU board.
VC_KL25Z_8VAPS_FlutterCurrent	This project shows how to use Flutter Current function to control low-side drivers for inductive loads using the MC34ValveController component. The target is the MC34SB0800 device and the TWR-KL25Z48M MCU board.
Kinetis Design Studio Examples	
VC_K20D72M_4VAPS_DriverControl	This demo project shows how to work with drivers of MC34SB0410 valve controller using MC34ValveController component. Target MCU is TWR- K20D72M.

Table 7. MC34ValveController Example Project and Components (continued)

Folder Name	Folder Contents
VC_ K20D72M _4VAPS_DriverMonitoring	The purpose of this project is to show how to monitor status of MC34SB0410 valve controller using MC34ValveCotontroller component. It is intended for TWR- K20D72M MCU board.
VC_ K20D72M _4VAPS_FlutterCurrent	This example project shows how to use Flutter Current function to control low-side drivers for inductive loads using MC34ValveController component. It is intended for MC34SB0410 device and TWR- K20D72M MCU board.
VC_ K20D72M _8VAPS_DriverControl	This demo project shows how to work with drivers of MC34SB0800 valve controller using MC34ValveController component. Target MCU is TWR- K20D72M.
VC_ K20D72M _8VAPS_DriverMonitoring	The purpose of this project is to show how to monitor status of MC34SB0800 valve controller using MC34ValveCotontroller component. It is intended for TWR- K20D72M MCU board.
VC_ K20D72M _8VAPS_FlutterCurrent	This example project shows how to use Flutter Current function to control low-side drivers for inductive loads using MC34ValveController component. It is intended for MC34SB0800 device and TWR- K20D72M MCU board.
VC_KL25Z_4VAPS_DriverControl	This demo project shows how to work with drivers of MC34SB0410 valve controller using MC34ValveController component. Target MCU is TWR-KL25Z48M.
VC_KL25Z_4VAPS_DriverMonitoring	The purpose of this project is to show how to monitor status of MC34SB0410 valve controller using MC34ValveCotontroller component. It is intended for TWR-KL25Z48M MCU board.
VC_KL25Z_4VAPS_FlutterCurrent	This example project shows how to use Flutter Current function to control low-side drivers for inductive loads using MC34ValveController component. It is intended for MC34SB0410 device and TWR-KL25Z48M MCU board.
VC_KL25Z_4VAPS_FlutterCurrentAuto	This example project shows how to use Flutter Current function in automatic mode. It is intended for MC34SB0410 device and TWR-KL25Z48M MCU board.
VC_KL25Z_4VAPS_FreeMASTER	This project demonstrates features of MC34SB0410 valve controller with TWR-KL25Z48M MCU board. It uses FreeMASTER tool for visualization and control. Latest Freemaster installation package: http://www.nxp.com/products/power-management/wireless-charging-ics/freemaster-run-time-debug-ging-tool:FREEMASTER
VC_KL25Z_4VAPS_SW_PID_Current Regulation	Example of PID current regulation driven by TWR-KL25Z48M MCU board with use of MC34SB0410 valve controller.
VC_KL25Z_8VAPS_DriverControl	This demo project shows how to work with drivers of MC34SB0800 valve controller using MC34ValveController component. Target MCU is TWR-KL25Z48M.
VC_KL25Z_8VAPS_DriverMonitoring	The purpose of this project is to show how to monitor status of MC34SB0800 valve controller using MC34ValveCotontroller component. It is intended for TWR-KL25Z48M MCU board.
VC_KL25Z_8VAPS_DriverMonitoring_iar	The purpose of this project is to show how to monitor status of MC34SB0800 valve controller using MC34ValveCotontroller component. It is intended for TWR-KL25Z48M MCU board and IAR compiler instead of GNU C.
VC_KL25Z_8VAPS_FlutterCurrent	This example project shows how to use Flutter Current function to control low-side drivers for inductive loads using MC34ValveController component. It is intended for MC34SB0800 device and TWR-KL25Z48M MCU board.
VC_KL25Z_8VAPS_FlutterCurrentAuto	This example project show how to use Flutter Current function in automatic mode. It is intended for MC34SB0800 device and TWR-KL25Z48M MCU board.
VC_KL25Z_8VAPS_FreeMASTER	This project demonstrates features of MC34SB0800 valve controller with TWR-KL25Z48M MCU board. It uses FreeMASTER tool for visualization and control. Latest Freemaster installation package: http://www.nxp.com/products/power-management/wireless-charging-ics/freemaster-run-time-debug-ging-tool:FREEMASTER
VC_KL25Z_8VAPS_PD_SPI_PWM	The purpose of this example project is to show how to implement PWM for Pump Motor Pre-driver when SPI control is used.
VC_KL25Z_8VAPS_SW_PID_Current Regulation	Example of PID current regulation driven by TWR-KL25Z48M MCU board with use of MC34SB0800 valve controller.
VC_KV31F_4VAPS_DriverControl	This demo project shows how to work with drivers of MC34SB0410 valve controller using MC34ValveController component. Target MCU is TWR- KV31F120M.
VC_KV31F_4VAPS_DriverMonitoring	The purpose of this project is to show how to monitor status of MC34SB0410 valve controller using MC34ValveCotontroller component. It is intended for TWR- KV31F120M MCU board.

Table 7. MC34ValveController Example Project and Components (continued)

Folder Name	Folder Contents
VC_KV31F_4VAPS_FlutterCurrent	This example project shows how to use Flutter Current function to control low-side drivers for inductive loads using MC34ValveController component. It is intended for MC34SB0410 device and TWR- KV31F120M MCU board.
VC_KV31F_8VAPS_DriverControl	This demo project shows how to work with drivers of MC34SB0800 valve controller using MC34ValveController component. Target MCU is TWR- KV31F120M.
VC_KV31F_8VAPS_DriverMonitoring	The purpose of this project is to show how to monitor status of MC34SB0800 valve controller using MC34ValveCotontroller component. It is intended for TWR- KV31F120M MCU board.
VC_KV31F_8VAPS_FlutterCurrent	This example project shows how to use Flutter Current function to control low-side drivers for inductive loads using MC34ValveController component. It is intended for MC34SB0800 device and TWR- KV31F120M MCU board.

4.2.1 Import the MC34ValveController Components into the Processor Expert Library

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 folder MC34ValveController_PEx_SW\Component. Select **MC34ValveController_bxxxx.PEupd** and SPI_Device_bxxxx.PEupd files then click **Open** (see Figure 7).

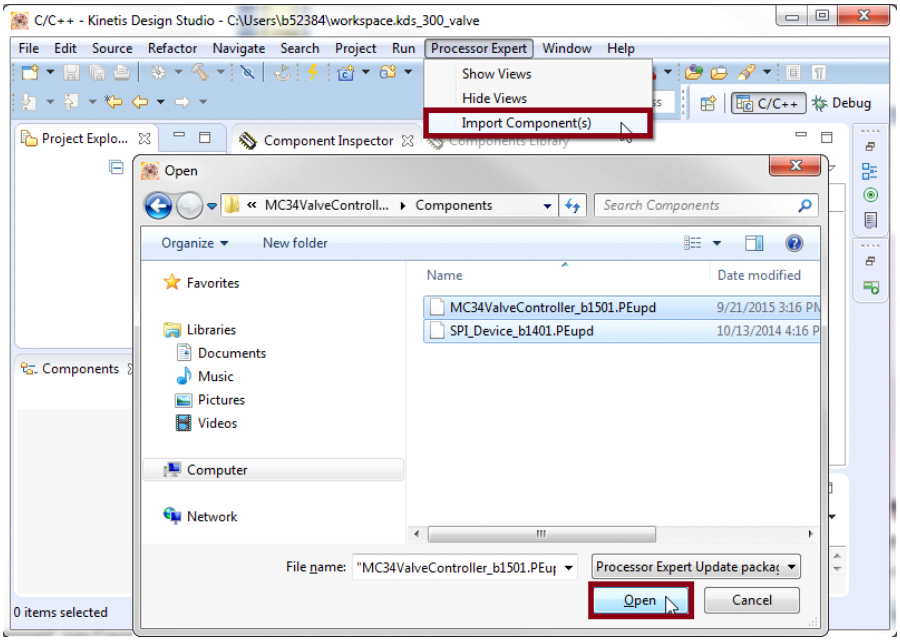

Figure 7. Import the MC34ValveController Components

3. If the import is successful, the MC34ValveController component appears in Components Library -> SW -> User Component (see Figure 8). The component is now ready to use.

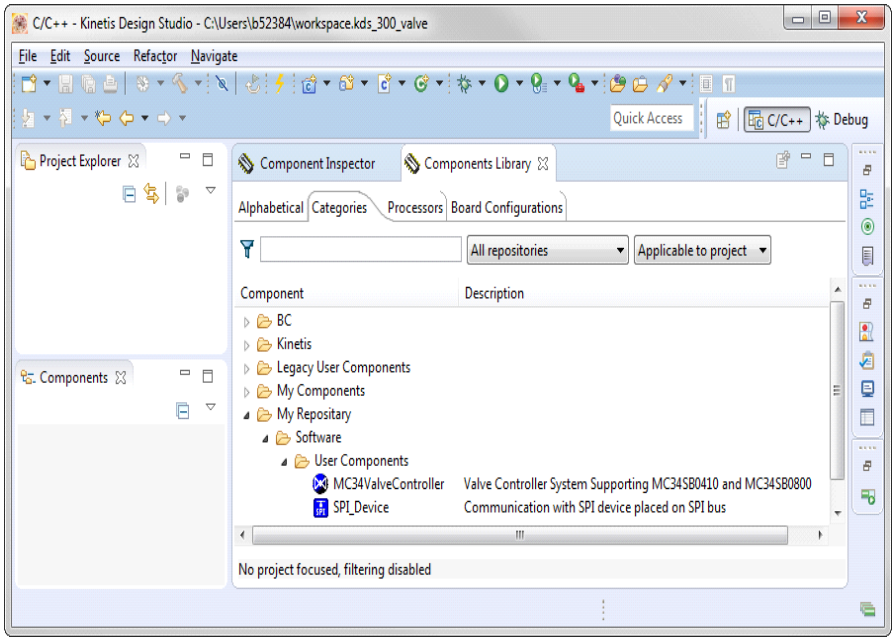

Figure 8. MC34ValveController Component Location After Importing to Kinetis Design Studio

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**.

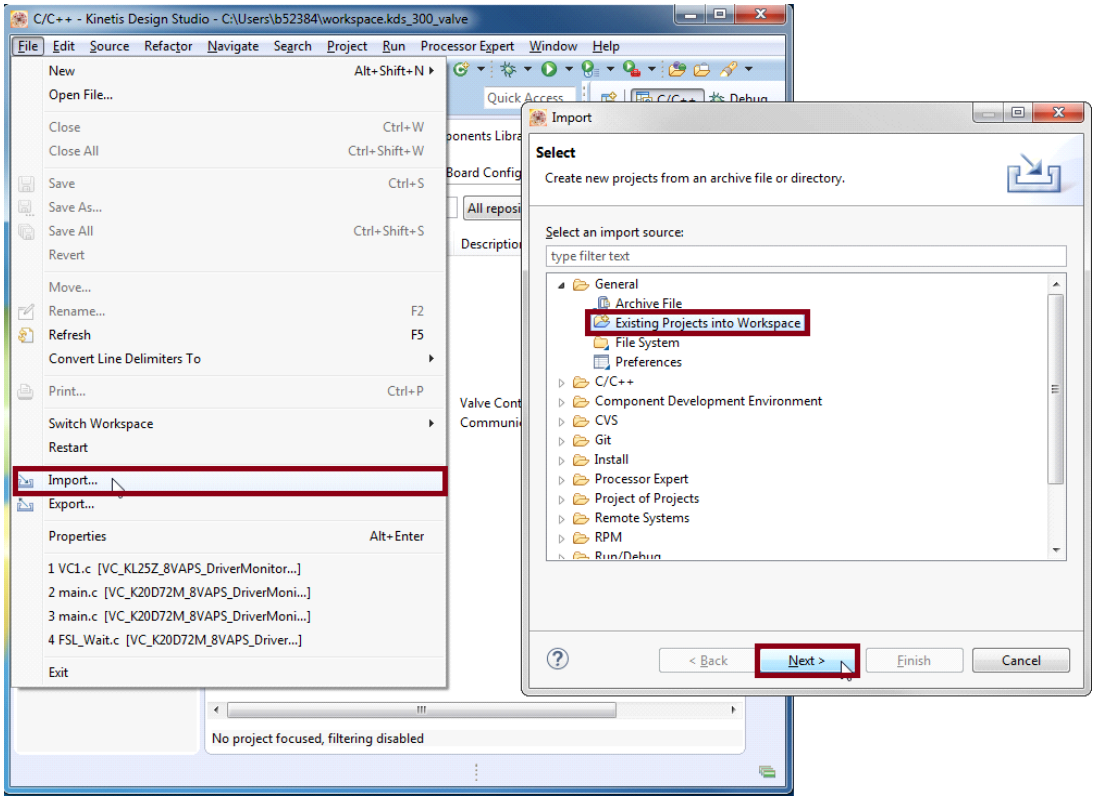

Figure 9. Importing an Example File (a)

2. Click **Browse** and locate the folder wherewith unzipped downloaded example files are located. Find the folder MC34ValveController_PEx_SW\KDS_Examples and select a project to import. (see Figure 10, which shows VC_K20D72M_4VAPS_DriverControl as the imported project). Then click **OK**.

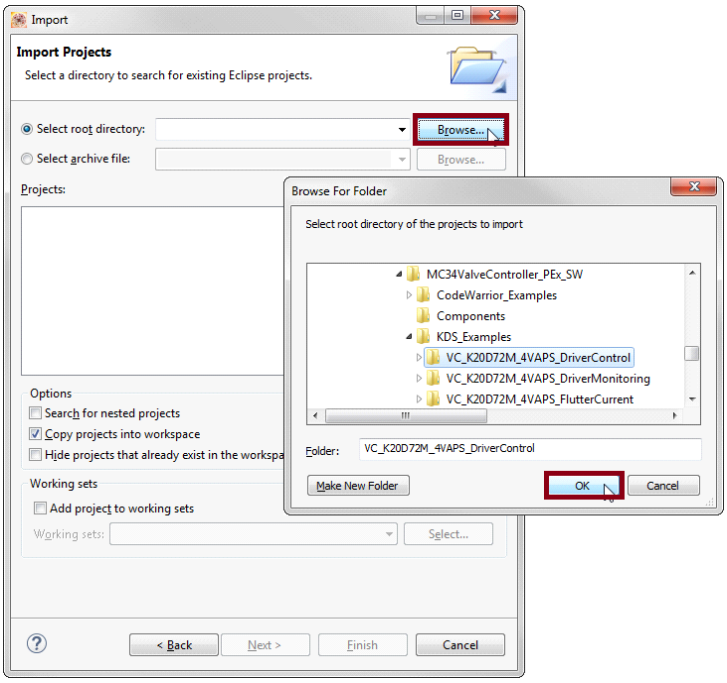

Figure 10. Importing an Example File (b)

3. With the project now loaded in the **Select root directory** box, click on the **Copy projects into workspace** check box. Then click **Finish**. Figure 11 shows the **Projects** panel and the **Components** panel after the project has been successfully imported. The project is now in the Kinetis Design Studio workspace where it can build and run.

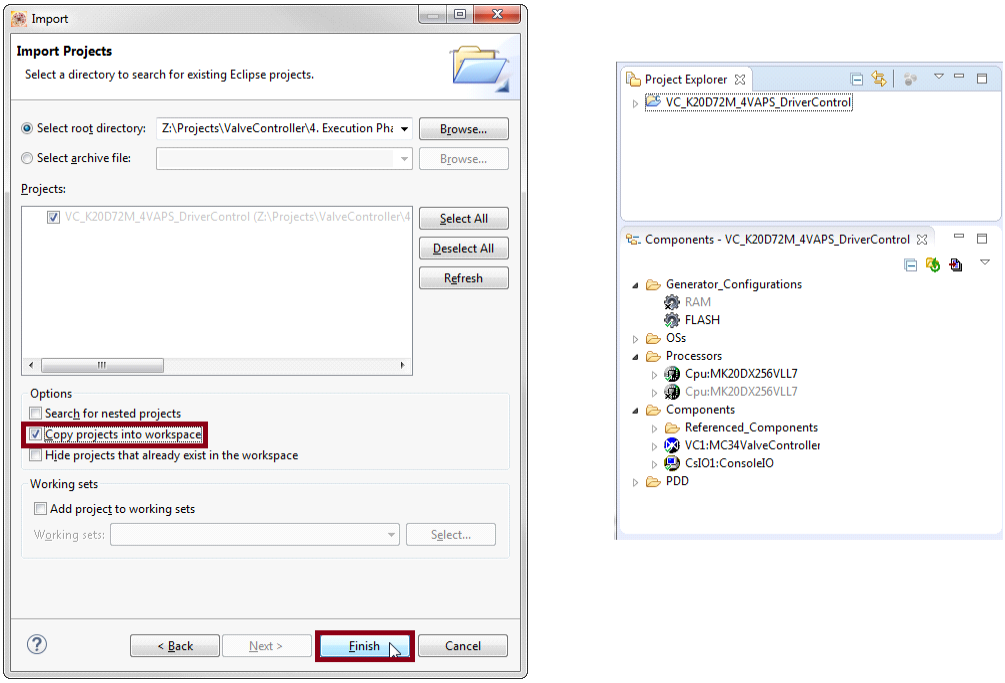

Figure 11. Importing an Example File (c)

4.3 Creating a New Project with Processor Expert and the MC34ValveController Components

If choosing not to use the example projects, the following instructions describe how to create and setup a new project using the MC34ValveController components. If the MC34ValveController does not have components in the Processor Expert Library, follow steps in Section 4.2.1, Import the MC34ValveController Components into the Processor Expert Library, page 16.

To create a new project do the following:

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**. (see Figure 12).

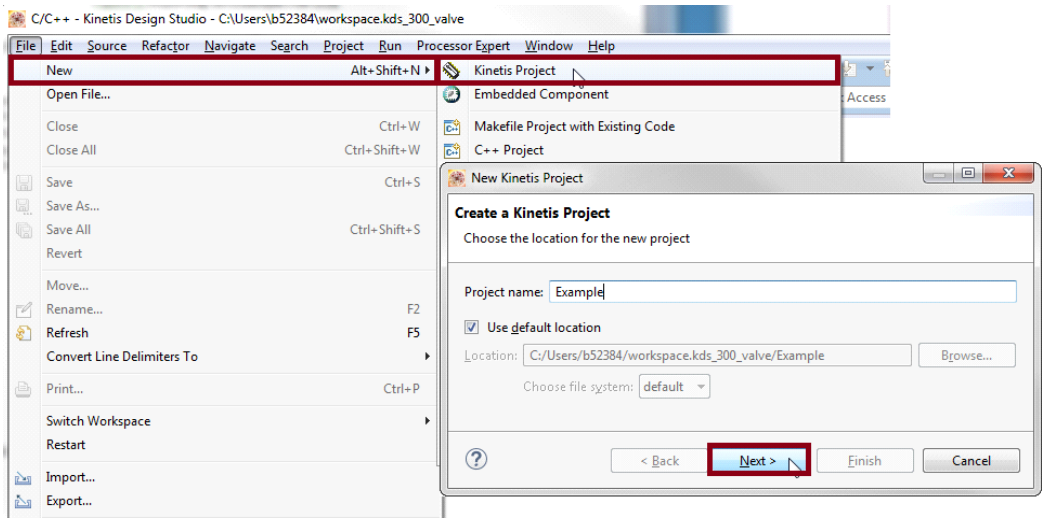

Figure 12. Creating a Kinetis Project

2. In the **Devices** dialog box, select the MCU class for the appropriate MCU. In Figure 13 MKL25Z128 has been selected. Then click **Next**.
3. In the **Rapid Application Development** dialog box, make sure that the **Processor Expert** option is selected. Then click **Next**.

4. In the **Processor Expert Target Compiler** dialog box, select a compiler to use (GNU C Compiler in Figure 13) and click **Finish**.

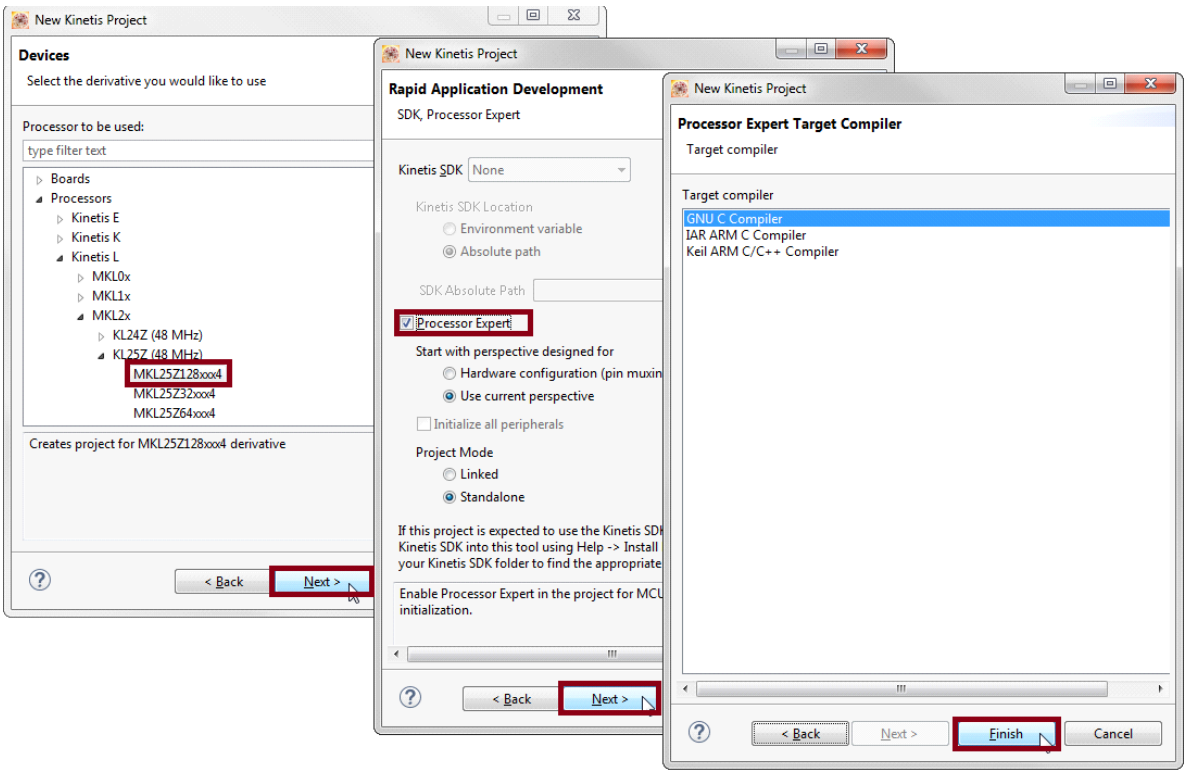

Figure 13. Selecting a Device, the Rapid Application Development Options and Compiler

5. Figure 14 shows the **Projects Explorer** panel and the **Components** panel after the project has been successfully created.

Before the project can be built and run, add the component (imported in Section 4.2.2, Importing an Example Project into Kinetis Design Studio, page 18) into the project. Section 4.3.1, Adding a MC34ValveController Component into the Project, page 22 outlines this procedure.

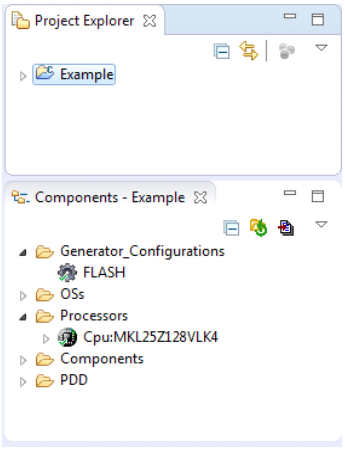

Figure 14. Project Explorer and Components Panels with Project Created

4.3.1 Adding a MC34ValveController Component into the Project

1. Find the MC34ValveController component in the **Components Library** and add it into the project (see Figure 15). It is located in the workspace directory selected when importing the component (My Repository in the example).

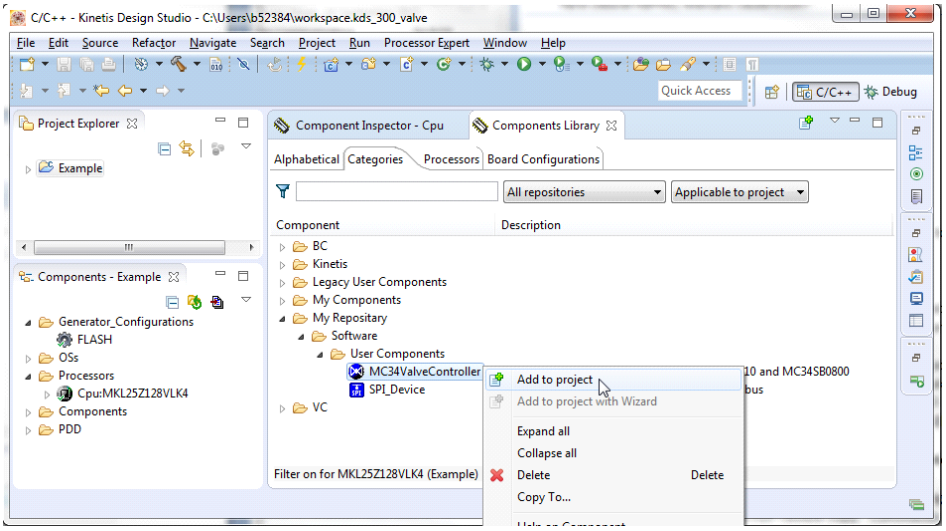

Figure 15. Add the MC34ValveController Component to the Project

2. Figure 16 shows the **Components** panel after the component was added. To view the **Component Inspector** options, double-click on the MC34ValveController component in the **Components** panel.

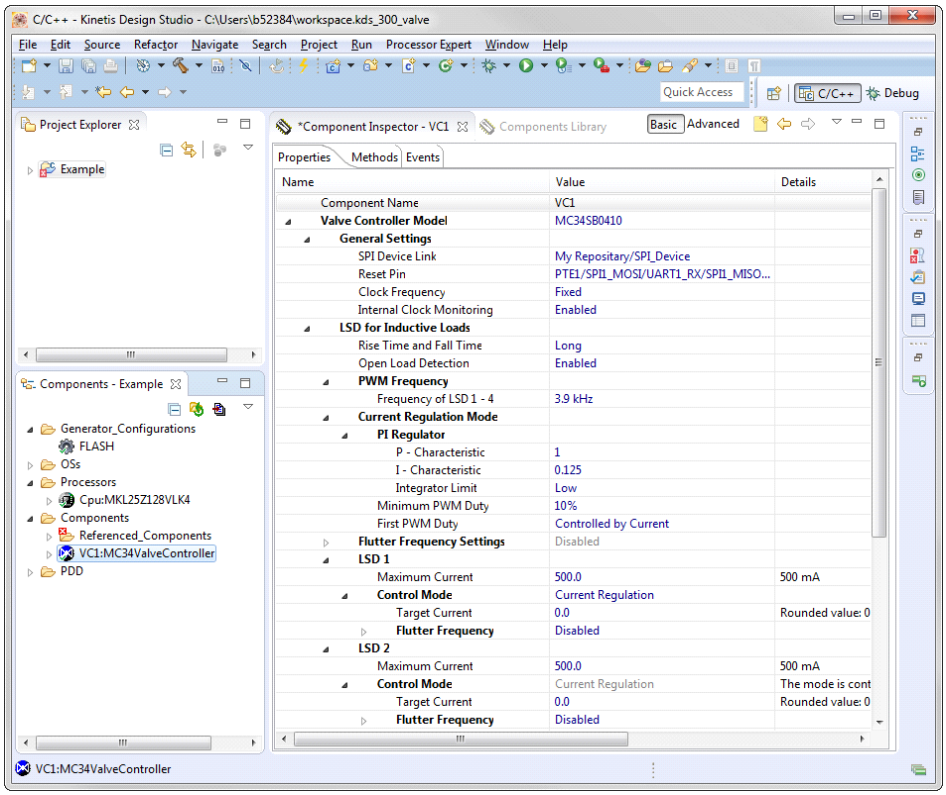

Figure 16. Show the Component Inspector

4.4 Setting up the Project

Once the new project has been created and the MC34ValveController component has been added into it, the component properties in the project must be set up. Make sure to read Section 3.1, Component Settings, page 7, which describes the component's capabilities and what must be done to configure its properties.

MC34ValveController uses several components (see Figure 17). Configure all the components in the following order:

1. Set up the **MC34ValveController** component.
2. Set up the referenced **SPI_Master_LDD** component.
3. Set up the CS pin under the inherited **SPI_Device** component.

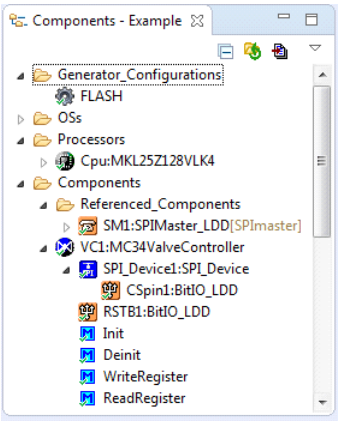

Figure 17. Setting up the Components

4.5 Generating Driver Source Code

After having completed configuring the components, the application is ready to generate the driver code to be incorporated. The process is as follows:

1. Click on the **Generate Processor Expert Code** icon in the upper right corner of the **Components** panel.

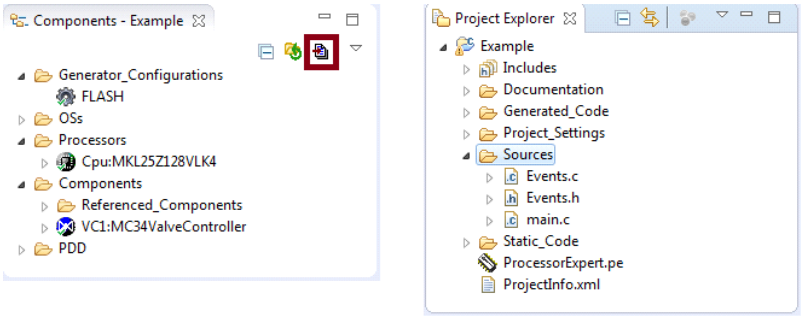

Figure 18. Generating the Source Code and Code Location

2. The driver code for the device 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 18 shows the locations of the generated driver source and the application code.

4.6 Writing the Application Code

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

To add a component method into the application source code:

1. In the **Components** panel for the project, click on **Components**. Find the desired 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 a list of the required parameters.

For example, open the MC34ValveController component method list, drag and drop **ReadRegister** into **main.c** and add the necessary parameters. (See Figure 19).

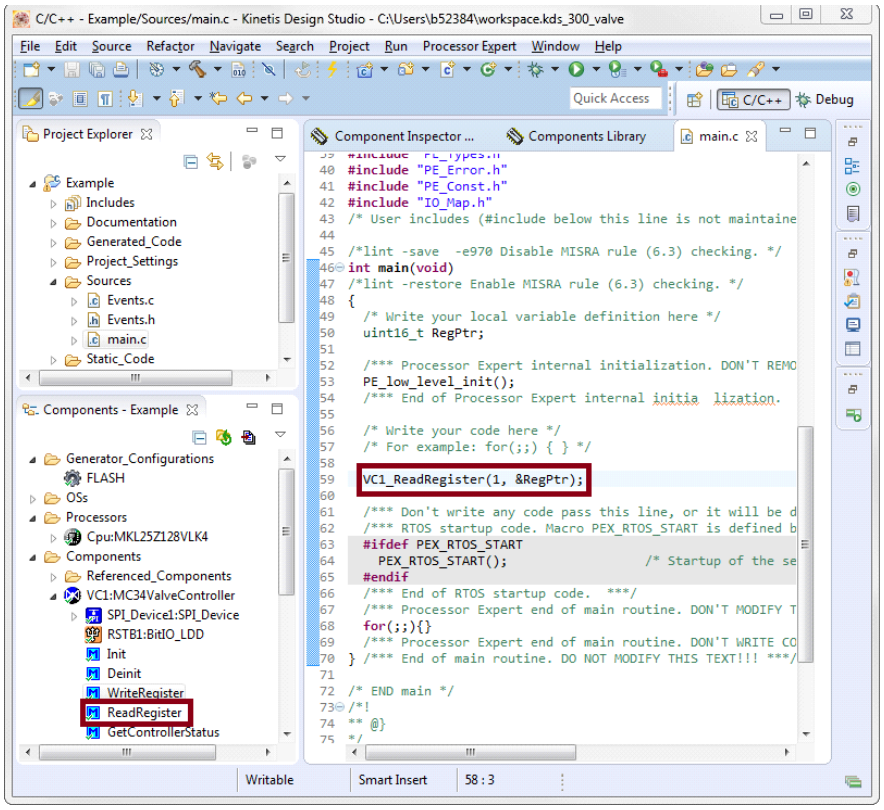

Figure 19. Adding Component Methods

Hovering the mouse over any of the methods displays a description of the method, including a list of required parameter.

The MC34ValveController component encompasses a help, which describes component properties, methods and typical usage. To show the help, do the following:

1. In the **Components** view, right-click MC34ValveController component and select **Help on Component**.
2. A web page with the Help information displays.

4.6.1 Compiling, Downloading and Debugging

To compile a project, click on the compile icon in the tool bar (see Figure 20).

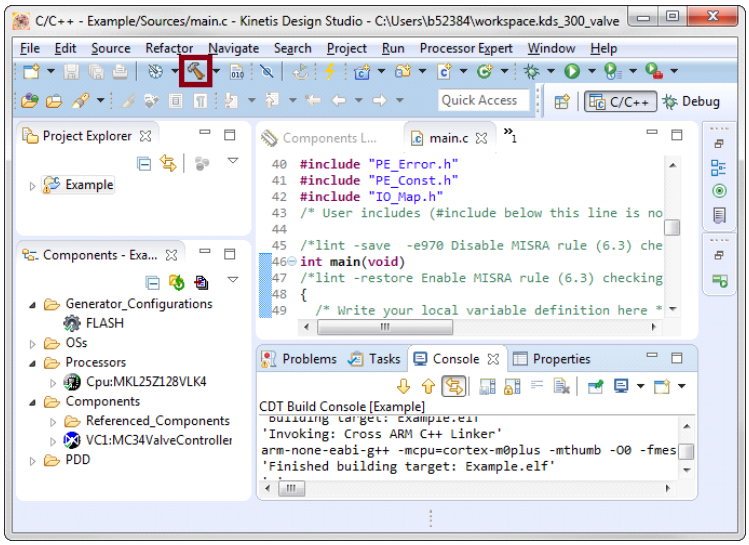

Figure 20. Compiling the Application

The process for downloading an application on board in Kinetis Design Studio may differ according to MCU board used. For any questions, see the Kinetis Design Studio user's guide.

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

1. Click the arrow next to the debug icon in the tabular and select Debug Configurations… (see Figure 21)

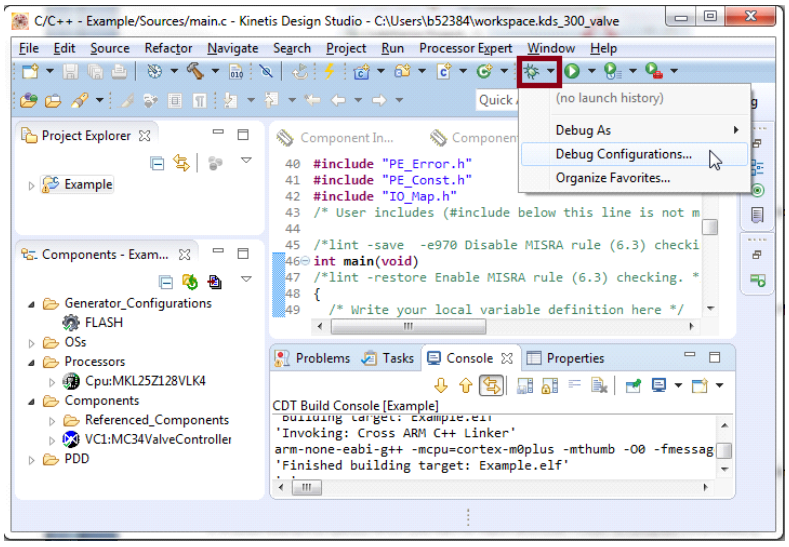

Figure 21. Downloading the Application (a)

2. In the Debug Configurations dialog box, click Example_Debug_PNE under GDB PEMicro Interface Debugging (see Figure 22).

3. Make sure that C/C++ Application contains a path to the .elf file of the project (see Figure 22).

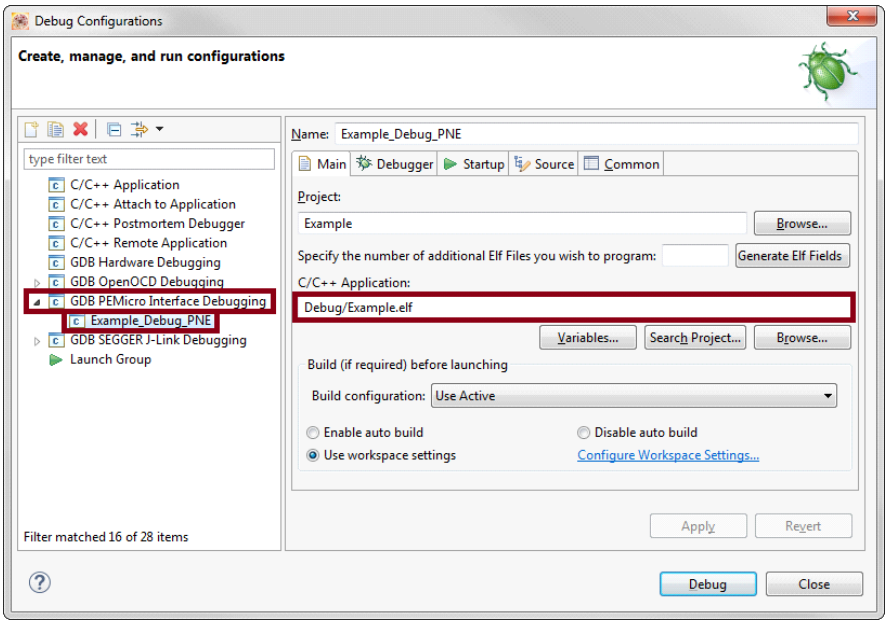

Figure 22. Downloading the Application (b)

4. Click the **Debugger** tab and set Interface option to **OpenSDA Embedded Debug - USB Port**. Then click **Refresh** button next to the **Port** setting to update list of available USB ports (see Figure 23).
5. Make sure the **Target** is set to **KL25Z128M4**. If not, change the target with use of the **Select Device** button. Click the button, in the **Select Target Device** dialog box go to Freescale -> KL2x -> KL25Z128M4 and confirm with the **Select** button.
6. Click **Debug**. Kinetis Design Studio will download and launch the program on board.

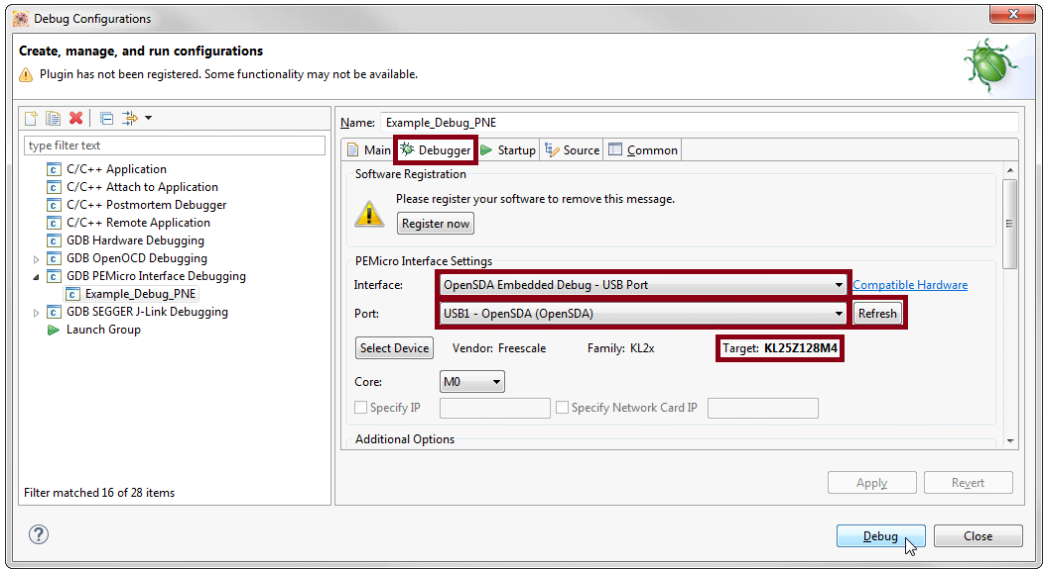

Figure 23. Downloading the Application (c)

5 References

Following are URLs where you can obtain information on related NXP products and application solutions:

Table 8. References

NXP.com Support Pages	Description	URL
MC34SB0410	Product Summary Page	www.nxp.com/MC34SB0410
MC34SB0800	Product Summary Page	www.nxp.com/MC34SB0800
TWR-SB0410-36EVB	Tool Summary Page	www.nxp.com/TWR-SB0410-36EVB
TWR-SB0800-36EVB	Tool Summary Page	www.nxp.com/TWR-SB0800-36EVB
Tower System	Tower System Modular Development Board Platform	www.nxp.com/tower
Kinetis Design Studio	Software	www.nxp.com/kinetis
CodeWarrior	Software	www.nxp.com/codewarrior
Processor Expert Code Model	Code Walkthrough Video	www.freescale.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	1/2016	• Initial release

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: SB0410-SB0800SWUG
Rev. 1.0
1/2016