Document Number: PEXBC3770SWUG
Rev. 1.0, 2/2016

BC3770 and FRDM-BC3770-EVB Processor Expert components

Contents

1 Overview

This documentation describes how to install and use Processor Expert in conjunction with the BC3770 components.

The FRDM_BC3770 and BC_MC32BC3770 Processor Expert components are software drivers that encapsulate the functionality of the MC32BC3770CS battery charger device and its companion evaluation board, the FRDM-BC3770-EVB. These components serve as an API layer between the hardware and the user application. The BC3770 components facilitate application development by offering an easy-to-use interface for setting options related to charging parameters, register settings, measurements and testing.

The BC_MC32BC3770 component offers MC32BC3770CS battery charger methods that allow the user to set charger modes, interrupts and charging parameters. The FRDM_BC3770 component supports FRDM-BC3770-EVB Freedom board functionality. It contains methods for current, voltage and temperature measurement. This component uses Current Sense Amplifiers, enabling current and voltage sensing on the power supply (VBUS), the battery charger output (VSYS) and the battery (VBAT). FRDM_BC3770 component methods also offer electronic load settings making it possible to test user applications in real time.

This component supports the NXP MC32BC3770CS device, a 2.0 A switch-mode Li-ion/Li-polymer battery charger. NXP's FRDM-BC3770-EVB board is designed around this device. For detailed descriptions, see the related hardware user guides and datasheets.

2 BC_MC32BC3770 and FRDM_BC3770 compatibility

2.1 Peripheral requirements

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

- Seven **GPIOs** are required for Battery Charger input pins and interrupt pin handling.
- Two **I^{2}C** modules are required for I^{2}C communication. One I^{2}C peripheral controls communication with the MC32BC3770CS Battery Charger device. The other I^{2}C peripheral handles ELOAD (Electronic Load) and CSA (Current Sense Amplifier) communication.
- **ADC** is required for temperature measurement.

2.2 Supported devices

The NXP MC32BC3770CS is a fully programmable switching charger with dual-path output for single-cell Li-Ion and Li-Polymer battery. The dual-path output allows mobile applications with a fully discharged battery to boot up the system.

The main features of the device are:

- High-efficiency synchronous switching regulator offers reduced heat and increased current capacity
- Single input with 20 V maximum protection input voltage and up to 2.0 A load current
- Dual-path output to enable a system with a dead battery to power-up
- Charge parameters are programmable over an I^{2}C interface operating at up to 400 kHz
- Integrated overvoltage protection (OVP) and Power FETs
- Support for 1.5 MHz switching

2.3 Supported MCUs

The BC3770 supports the MCUs listed in Table 1. The listed MCUs are a subset of the MCUs supported by Processor Expert for Kinetis using the logic device driver (LDD) layer.

Table 1. BC3770 components MCU compatibility

Component vs. Freedom Board	FRDM-KL25Z	FRDM-KL26Z	FRDM-KL46Z	FRDM-K20D 50M	FRDM-K22F	FRDM-K64F	FRDM-KL05Z	FRDM-KL43Z
BC_MC32BC3770	yes	yes	yes	yes	yes	yes	yes	yes
FRDM_BC3770	yes	yes	yes	no	no	no	no	no

3 FRDM_BC3770 and BC_MC32BC3770 components

This chapter explains the BC3770 components properties and methods and how to use them. Additional information is available through the Processor Expert Help feature. The Help feature is accessed by right-clicking on the component in the **Components** panel, then selecting the **Help on Component** menu item. The **Help on Component** window provides information on all component properties and methods. Access the **Typical Usage** section to view examples showing how to work with API methods.

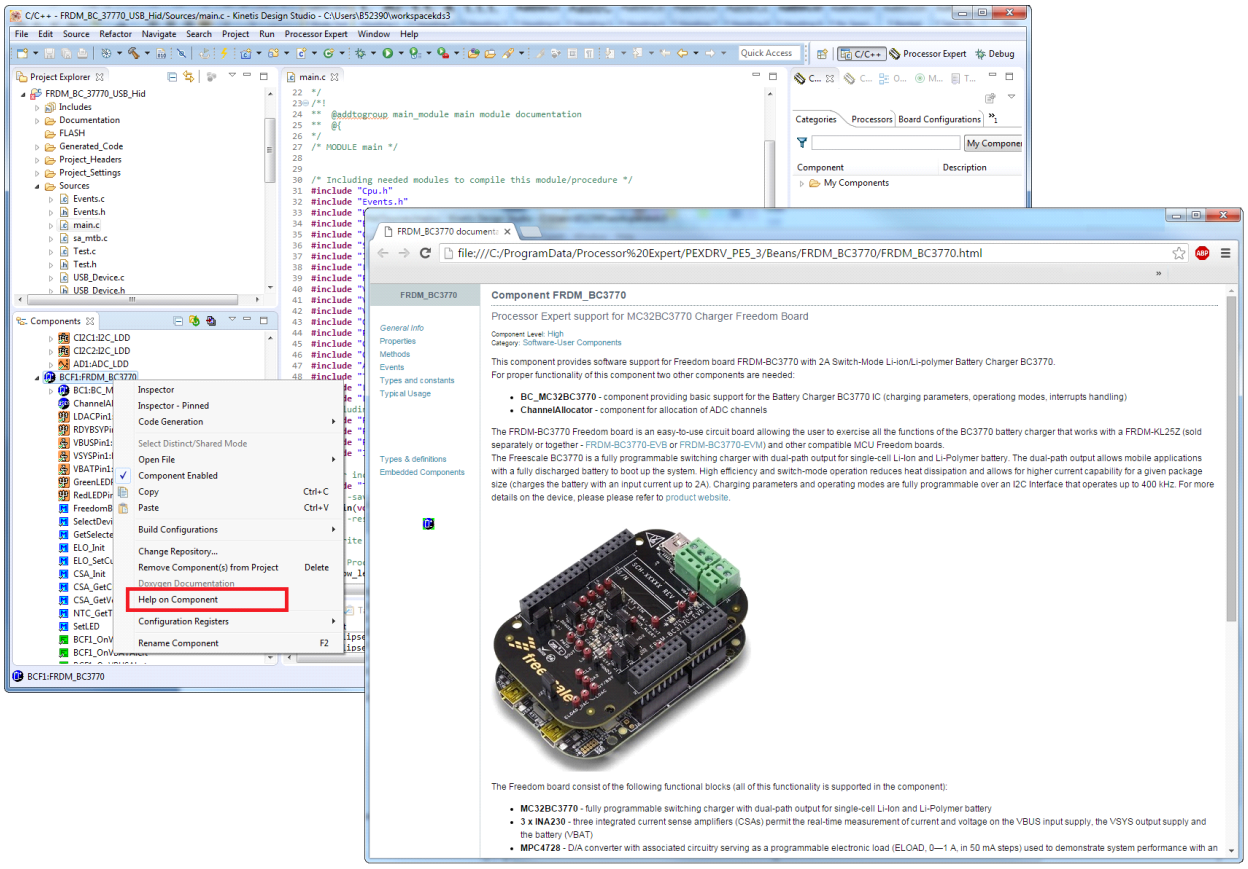

Figure 1. Help on component

The BC_MC32BC3770 component offers MC32BC3770CS battery charger methods that allow the user to set charger modes, interrupts and charging parameters. The FRDM_BC3770 component supports FRDM-BC3770-EVB Freedom board functionality. It contains methods for current, voltage and temperature measurement. This component uses Current Sense Amplifiers, enabling current and voltage sensing on the power supply (VBUS), the battery charger output (VSYS) and the battery (VBAT). FRDM_BC3770 component methods also offer electronic load settings making it possible to test user applications in real time.

Figure 8 depicts the components' functional design. FRDM_BC3770 inherits the ADC component for temperature measurement, the BitIO component for LEDs, LDAC synchronization and ready/busy pin handling. It also inherits the Channel Allocator component for ADC channel allocation. Linking to the I^2C component enables an I^2C peripheral to be used for communication with other devices besides Electronic Load (ELOAD) and external Analog-to-Digital converters (ADCs). The BC_MC32BC3770 component inherits the BitIO component for Charger Enable and Shutdown pin handling and the ExtInt component for interrupt notification from the charger. As with the FRDM_BC3770 component, the I^2C component allows for communication with other devices besides ELOAD and external ADCs.

3.1 BC_MC32BC3770 component

3.1.1 BC_MC32BC3770 component settings

Selecting the BC_MC32BC3770 component in the Processor Expert Components panel provides access to properties in the **Component Inspector** panel. These properties determine the component's general settings and its behavior after initialization. Some of these properties can later be changed in the application code by using the provided API.

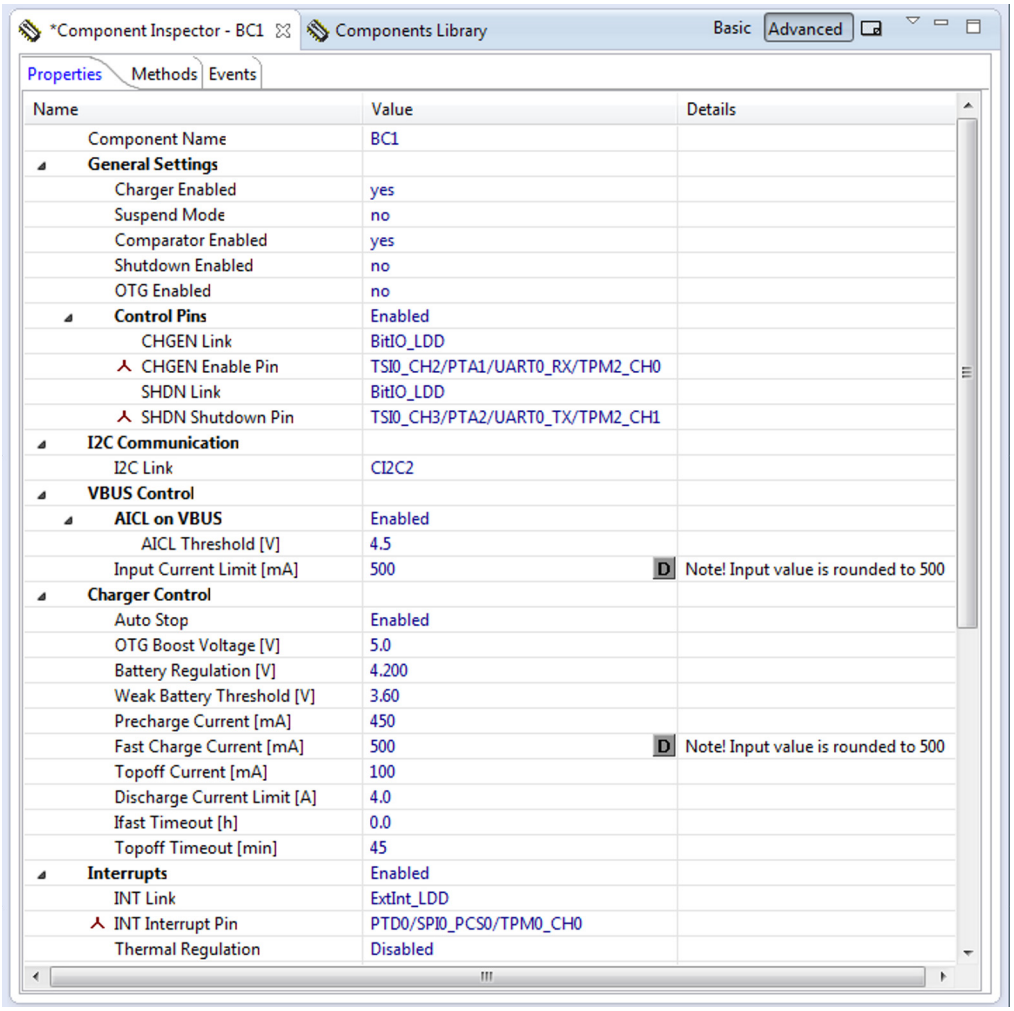

Figure 2. Component properties (not all interrupts are displayed in this figure)

Component properties are grouped into five main sections: **General Settings**, **I²C Communication**, **VBUS Control**, **Charger Control** and **Interrupts**.

General Settings encompasses charger mode settings. Setting the **Charger Enabled** property to **No** disables the battery charger and prevents the battery from being charged. However, other circuits and blocks (I²C, AICL, etc.) remain fully functional. If the **Charger Enabled** property is set to **Yes**, the battery charger either charges the battery or maintains constant voltage on the fully charged battery. In **Suspend** mode, PMID output is bypassed to VBUS, meaning the charger does not affect output voltage or current. In Boost (**OTG Enabled**) mode the device provides a regulated output voltage to VBUS from the battery. In **Shutdown Enabled** mode, if there is no valid input source, the charger stays functional except for the I²C interface, which is turned off to minimize power consumption. Setting the **Shutdown Enabled** mode is not effective as long as a valid input source is present.

I²C Communication allows the selection of linked **I2C_LDD** components, which are used for communication with the battery charger.

VBUS Control contains options for setting the Adaptive-Input Current Limit (AICL) feature of the battery charger. This feature is useful when the current and voltage power supply is limited. Under such circumstances, AICL prevents the power supply from collapsing when the required input current exceeds the maximum output current of the supply. The battery charger in Start-up mode automatically starts incrementing the input current limit to either the default value or a pre-programmed value. If a pre-programmed value is used, the input current is incremented until either the input current limit (IIN_LIM) is detected or the VBUS voltage (VAICL_TH) detects the AICL threshold (see Figure 3). If the input current exceeds the power supply current limit, the AICL function takes over and lowers the charge current below the programmed value. See the battery charger data sheet for more information.

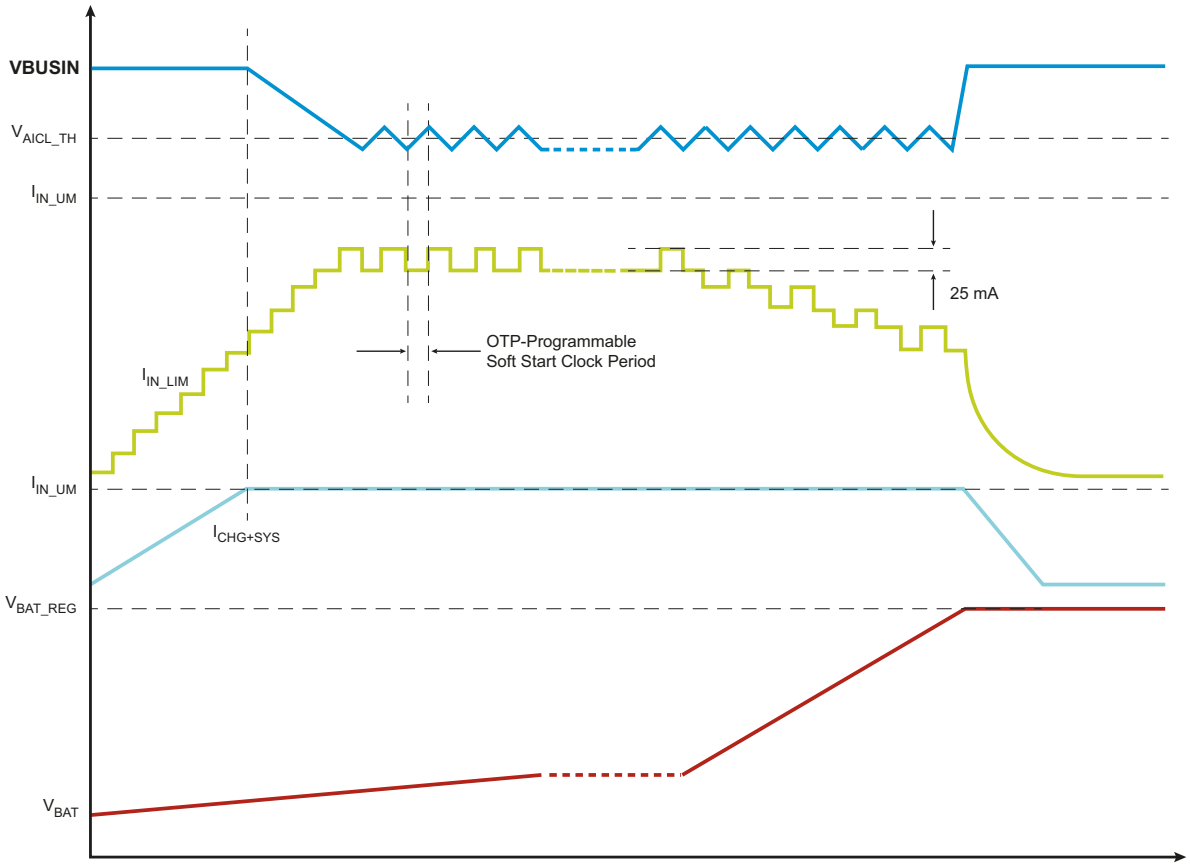

Figure 3. Adaptive-Input current limit

The **Charger Control** section contains charging parameter options. **Auto Stop** controls battery charger behavior after the charging process is finished. When **Auto Stop** is enabled, the battery charger turns off and goes into the DONE state when the Topoff timer expires (i.e. the charging process is finished). Otherwise, the battery charger remains on continuously and stays in Constant Voltage mode after charging, which means it maintains a certain voltage level on the battery. This section also contains options for current settings in **Pre-charge**, **Fast charge** and **Top-off** mode. Note that you cannot switch between these modes. The battery charger continually transits from one mode to the next based on the battery voltage. **IFast Timeout [h]** and **Topoff Timeout [min]** are safety timers. If the battery voltage does not reach a certain voltage threshold before the timer expires, charging is suspended and a fault signal is asserted.

Battery Regulation [V] sets the voltage maintained on the battery when the battery charger is on (**Auto Stop** is disabled) and the battery is fully charged. If the battery charger is off after charging, the battery voltage decreases until it reaches the **Weak Battery Threshold [V]** value, at which point an interrupt is asserted. For more information please see the MC32BC3770 battery charger data sheet.

3.1.2 BC_MC32BC3770 component API

Table 2 describes all of the public methods and events related to the BC_MC32BC3770 component. Component methods are also listed in the **Components** panel, as depicted in Figure 4. Methods and events marked with green tick marks are included when source code is generated. Methods and events marked with black crosses are not included (see Figure 4). To change these settings, go to the **Component Inspector** panel and select the **Methods** tab (see Figure 5). Note that some of the methods/events are always generated because they are needed for proper functionality.

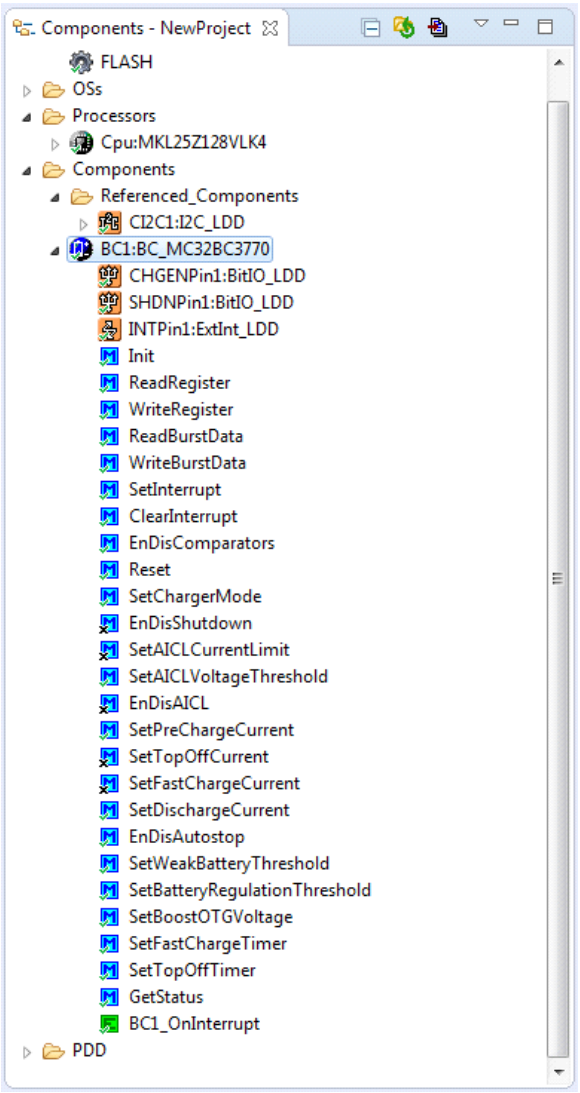

Figure 4. BC_MC32BC3770 methods in the components panel

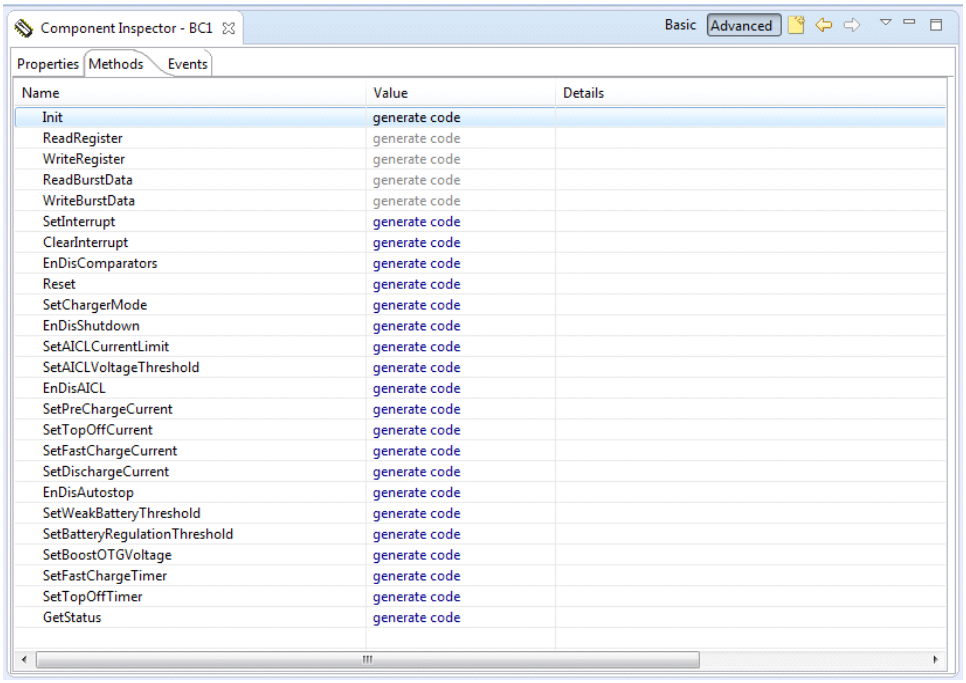

Figure 5. BC_MC32BC3770 generated methods

Table 2. BC_MC32BC3770 methods and events

Methods/Events	Description
Init	Initializes the device according to the component properties. This method writes data according to the component properties into registers via I^2C. When auto initialization is enabled, this method is called automatically within the PE initialization function - PE_low_level_init().
ReadRegister	Reads data from a single register defined by RegAddr argument. A returned status of ERR_OK doesn't necessarily mean the reception was successful. Use events (OnMasterSendComplete or OnError) to detect the actual state of the reception.
WriteRegister	Writes data to a single register defined by RegAddr argument.
ReadBurstData	Reads data from multiple registers via I^2C. The first parameter is an address of the first read register. Addresses of following registers increment automatically, so incoming bytes of data represent the content of consecutive registers.
WriteBurstData	Writes data to multiple registers via I^2C. The first parameter is an address of the first register to be written. Addresses of following registers increment automatically, so out coming data is written to consecutive registers automatically
SetInterrupt	Enables or disables an interrupt. Interrupts can be set individually or all at once. One interrupt or all interrupts are the only options. It is not possible, for example, to set only two interrupts at once.
ClearInterrupt	Clears interrupt flags in INT1…3 register. Interrupt flags can be cleared either individually or all at once. It is not possible to clear, for example, two interrupts.
EnDisComparators	Enables/disables comparators enabled by default. The comparators detect weak battery, supply voltage status, battery overvoltage protection (OVP) and discharge limit.
Reset	Resets the device's registers, except INTMASK and STATUS.
SetChargerMode	Sets the Charger mode. The charger can be on or off. When the charger is on, it charges the battery or maintains constant voltage on the battery. In Suspend mode, the PMID output is bypassed to VBUS, which means the charger does not influence output voltage and current. In Boost mode, the device provides a regulated output voltage to VBUS from the battery. In Shutdown mode, if there is no valid input source, the charger is functional except for the I^2C interface, which is turned off to minimize power consumption. The device enters Charge mode when a valid input source is present.

Table 2. BC_MC32BC3770 methods and events (continued)

Methods/Events	Description
EnDisShutdown	Enables/disables the shutdown pin, which means the device is put in Shutdown mode. In Shutdown mode the I^2C interface is turned off to minimize power consumption. However, this applies only when an invalid input power source is applied. This pin is not effective as long as a valid input power source is present.
SetFastChargeTimer	Fast charge timer watches the device during Fast-charge mode. An interrupt is asserted if the battery voltage does not reach its required value within this “fast-charge” time frame. Possible values are 3.5, 4.5 and 5.5 hours, or the timer can be disabled.
SetTopOffTimer	Top-off timer watches the device during Top-off mode. An interrupt is asserted if the battery voltage does not reach its required value within this “top-off” time frame. Possible values are: 10, 20, 30, or 45 minutes.
GetStatus	Returns the content of the status register.
SetPreChargeCurrent	Pre-charge current is current charging the battery in pre-charge mode. The battery charger enters Pre-charge mode when battery voltage is higher than 2.5 V. If the battery voltage does not exceed the VSYS_MIN threshold before the pre-charge timer expires, charging is suspended and a fault signal is asserted via the INTB pin.
SetFastChargeCurrent	Fast-charge current is current charging the battery in Fast-charge mode. The Fast-charge mode is entered when the battery voltage exceeds the VSYS_MIN threshold of a typical 3.6 V. If the battery voltage does not reach the VBAT_REG threshold before the timer expires, charging is suspended and a fault signal is asserted via the INTB pin.
SetTopOffCurrent	Top-off-charge current is current charging the battery in Top-off mode. After the top-off timer expires, the top-off event is reported to the processor via the INTB pin, which means the battery is fully charged. As soon as the processor reads the interrupt registers, the processor is able to turn off the charger.
SetDischargeCurrent	Sets the discharge current limit in Discharge mode.
SetBatteryRegulation Threshold	Based on this threshold, the charger transits from Fast-charge mode (Constant-current mode) to Full-charge mode (Constant-voltage mode). In Full-charge mode, the fast charge current is reduced to a programmable top-off current. Up to this threshold the VSYS output tracks the battery voltage in Trickle and Pre-charge mode.
SetWeakBatteryThreshold	Sets the weak battery threshold voltage. The threshold ranges from 3.0 V to 3.75 V in 50 mV steps. A weak battery detection function allows the processor to acknowledge the low-battery condition by asserting an INTB event.
SetBoostOTGVoltage	Sets OTG voltage in Boost (OTG) mode. In Boost mode the device provides a regulated output voltage to VBUS from the battery.
EnDisAutostop	Enables or disables the autostop feature. If autostop is enabled after the top-off timer expires, the charger turns off and goes into DONE state. If it is disabled, the charger is on continuously and stays in CV mode after the top-off timer is expires.
EnDisAICL	Enables/disables adaptive-input current limit (AICL). AICL is mostly used at the beginning of the charging process when current dissipation is higher than the current the power source can provide. This feature prevents the power source from collapsing.
SetAICLVoltage Threshold	Sets AICL threshold voltage on VBUS. To keep the device functional with a current and voltage limited VBUS source, the device in Start-up mode automatically starts incrementing the input current limit. The current may be incremented to the default input current limit. Alternatively, it may be incremented to a pre-programmed value until either the input current limit is detected or the VBUS voltage detects the AICL threshold. This keeps input supply voltage as a valid power source to provide the load for the application. The device allows the maximum current the input supply can possibly provide without severely collapsing.
SetAICLCurrentLimit	Sets the input current limit by writing to VBUSCTRL register. This value limits the fast-charge current when the device is in Fast-charge mode. It also sets the limit for the adaptive-input current limit (AICL) when a device in Start-up mode automatically starts incrementing the input current limit to either the default or pre-programmed value until either the input current limit is detected or the VBUS voltage detects the AICL threshold. This keeps input supply voltage as a valid power source to provide the load for the application.
OnInterrupt	Interrupt event handler from the battery charger. This event is invoked every time there is a falling edge on the INT interrupt pin. Contents of the registers in the device structure are updated prior to this event, so the interrupt registers directly from this structure (without sending I^2C command) can be read.

3.1.3 Interrupt handling

If an interrupt from the battery charger occurs, an **OnInterrupt** event is invoked. (This interrupt handler is located in the **Events.c** file in the **Sources** folder of your project.) When such an interrupt occurs, there are two options. The first is to read the Interrupt registers directly from Battery charger. The second is to read the interrupt registers from the data structure **DeviceData**, which is updated prior to the **OnInterrupt** event being invoked. The **OnInterrupt** event is useful when it is necessary to set a flag to report some event has occurred.

3.2 FRDM_BC3770 component

Just as the BC_MC32BC3770 component provides an interface for the battery charger, the FRDM_BC3770 component covers all the functionality of the FRDM-BC3770-EVB Freedom board. The FRDM_BC3770 component facilitates user application testing and evaluation by providing an API for current, voltage and temperature measurement and for Electronic load settings.

3.2.1 FRDM_BC3770 component settings

This section summarizes main features of the FRDM_BC3770 component and provides an overview of the properties appearing in the Component Inspector view. Table 3 lists all of the methods and events related to the component. Figure 6 shows typical Component Inspector properties for a project using the FRDM_BC3770 component.

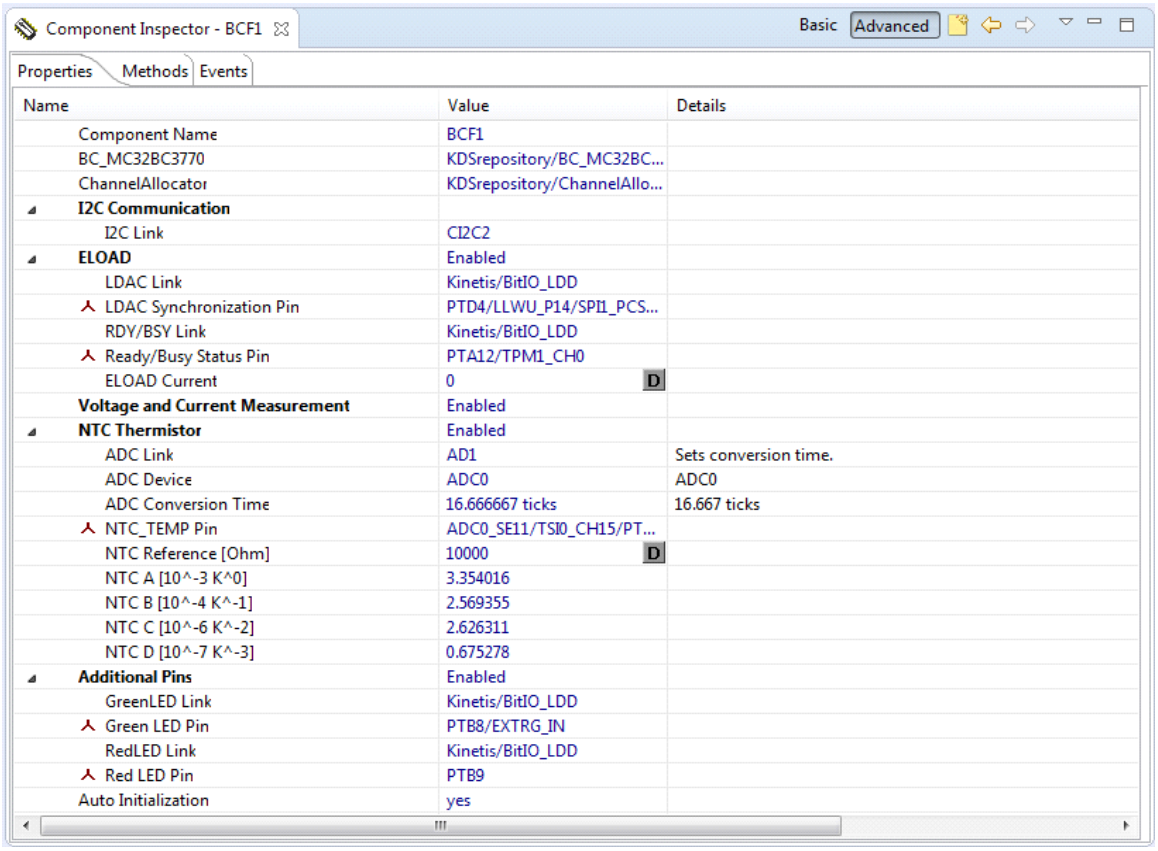

Figure 6. FRDM_BC3770 properties

In the Component Inspector view, the FRDM_BC3770 component properties are divided into five sections:

1. **I^2C Communication** provides selection of the linked **I2C_LDD** component used for communication with the Current Sense Amplifiers and Electronic Load.
2. **ELOAD** (Electronic Load) contains pin settings for the electronic load's **LDAC** and the **Ready/Busy** pins. **LDAC** is used as a flag for transferring the contents of the input registers to their corresponding DAC output registers. The **Ready/Busy** pin is a status indicator of EEPROM programming activity. **ELOAD Current** sets the amount of current the electronic load sinks from either battery or VSYS output.

3. **Voltage and Current Measurement** enables the Current Sense Amplifiers (CSA) to measure VBUS, VSYS and VBAT voltage and current.
4. **NTC Thermistor** contains settings for temperature measurement. The **ADC Link** property links the **ADC_LDD** component to the FRDM_BC3770. The **NTC A/B/C/D** properties are coefficients of the Steinhart-Hart equation used as an approximation for temperature calculation from voltage measured on NTC thermistor. Find these coefficients in the data sheet for the NTC thermistor. Note: make sure the number format of coefficients in the data sheet corresponds to the format in component properties. For example, if the first coefficient NTC A is given as 0.5E-04 (or $0.5x10^{-4}$), convert it to 0.05E-3 and enter only the significant part of the number (i.e. 0.05). Otherwise the calculated value cannot correspond to the real temperature. The NTC reference value is thermistor resistance at 25 °C.
5. **Additional Pins** offers pin settings for the green and red LEDs, which can be used as an indicator of an event in the user application.

3.2.2 FRDM_BC3770 component API

Table 3 describes all of the public methods and events related to the FRDM_BC3770 component. Component methods also appear in the **Components** panel as depicted in Figure 7. Methods and events marked with green tick marks are included when source code is generated; Methods and events marked with black crosses are not included (see Figure 7). To change these settings, go to the **Component Inspector** panel and select the **Methods** tab. Note that some methods/events are always generated, because they are needed for proper functionality.

Table 3. BC_MC32BC3770 Methods and Events

Methods/Events	Description
FreedomBoard_Init	Initializes devices on the board (current sense amplifiers and electronic load) and assigns a user defined data structure describing the board.
SelectDevice	Selects one of the devices on the board: electronic load or some of the current sense amplifiers to measure current or voltage.
GetSelectedDevice	Returns last selected device.
ELO_Init	Initializes electronic load (ELOAD). Initializes device MPC4728 with default values. Outputs B, C and D are in Power-down mode. Only output A, used for ELOAD control, is in Normal mode.
ELO_SetCurrent	Sets the ELOAD set point (the amount of current ELOAD sinks). This method is the same as the internal ELO_SetPointEload except for the function parameter, which is a real number [mA]. This method sends a 'new value' command to the output register of channel A. Resolution of the output voltage is 12 bits. The 'send new value' command sets default settings for other registers.
CSA_Init	Initializes the device and sets the default configuration (see INA230 documentation) for VBUS, VSYS, BATTERY amplifier and the settings of the calibration register.
CSA_GetCurrent	Adds the content of the Current Register for the selected device and converts it to current in milliamperes [mA]. To change the selected device, use the CSA_SelectDevice method.
CSA_GetVoltage	Reads the Bus Voltage or Shunt Voltage register of the selected device according to the first parameter. The value of the register is converted to voltage in millivolts [mV].
CSA_ReadRegister	Reads the content of the selected Current Sense Amplifier register.
CSA_WriteRegister	Writes value to the selected Current Sense Amplifier register.
NTC_GetTemperature	Measures the voltage on the NTC thermistor and calculates temperature according to the Steinhart-Hart equation. Returned temperature is in Kelvin scale - T[K]. The precision of the resulting temperature depends on the NTC constants of the equation. Add the correct constants A, B, C, D specified in the NTC thermistor data sheet.
SetLED	Turns on/off the green or red LED.

3.2.3 I²C configuration

The FRDM_BC3770 component requires two different I2C_LDD components — one for FRDM_BC3770 and the other for BC_MC32BC3770. This is because there are two I²C interfaces used on the Freedom evaluation board. One interface is used for communication with the battery charger and the other handles communication with the current sense amplifiers and the electronic load. If the project is not configured with two different I2C_LDD components, Processor Expert reports an error.

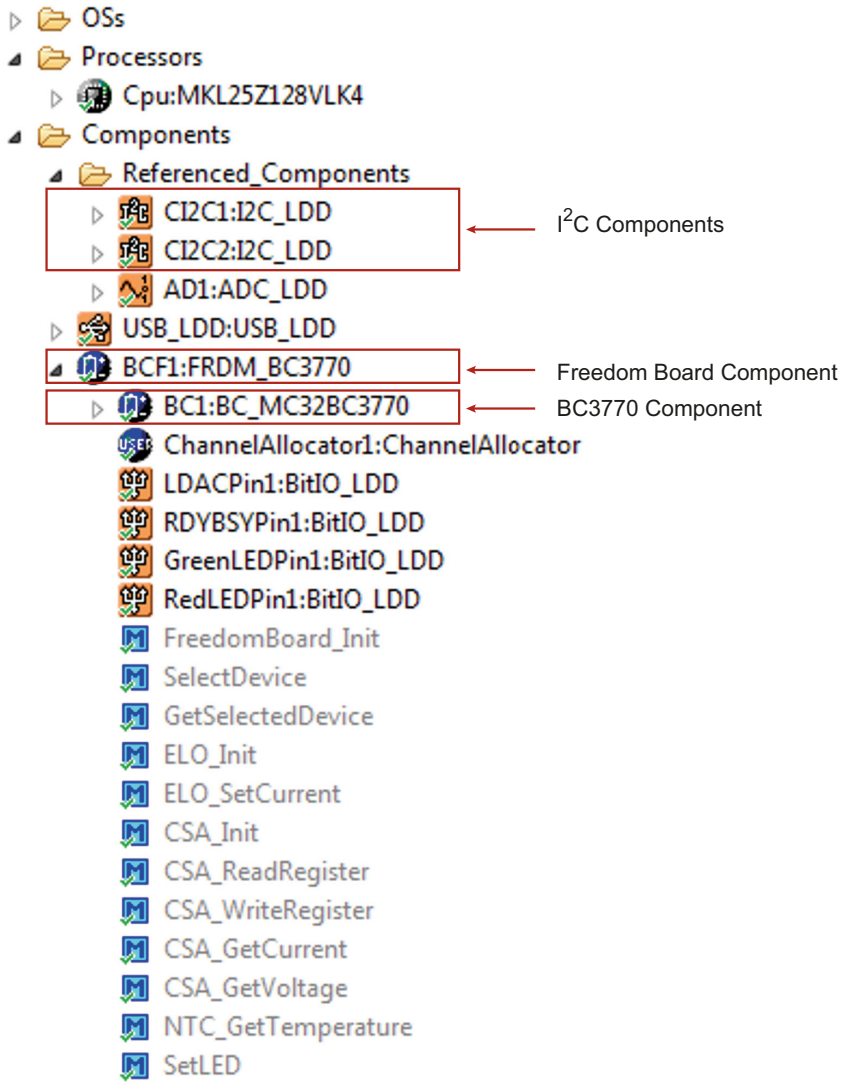

Figure 7. FRDM_BC3770 Components panel with two I²C components

3.3 Utilized components

Figure 8 depicts the functional design of the components. FRDM_BC3770 inherits the BC_MC32BC3770 component, the ADC component for temperature measurement, the BitIO component for LEDs, LDAC synchronization and ready/busy pin handling. It also inherits the Channel Allocator component for ADC channels allocation. Linking to the I^2C component allows I^2C peripherals to communicate with other devices besides electronic load and external ADCs. The BC_MC32BC3770 component inherits the BitIO component for charger enable and shutdown pins handling and the ExtInt component for interrupt notification from the charger. As with the FRDM_BC3770, the I^2C component enables I^2C peripherals to communicate with other devices besides electronic load and external ADCs.

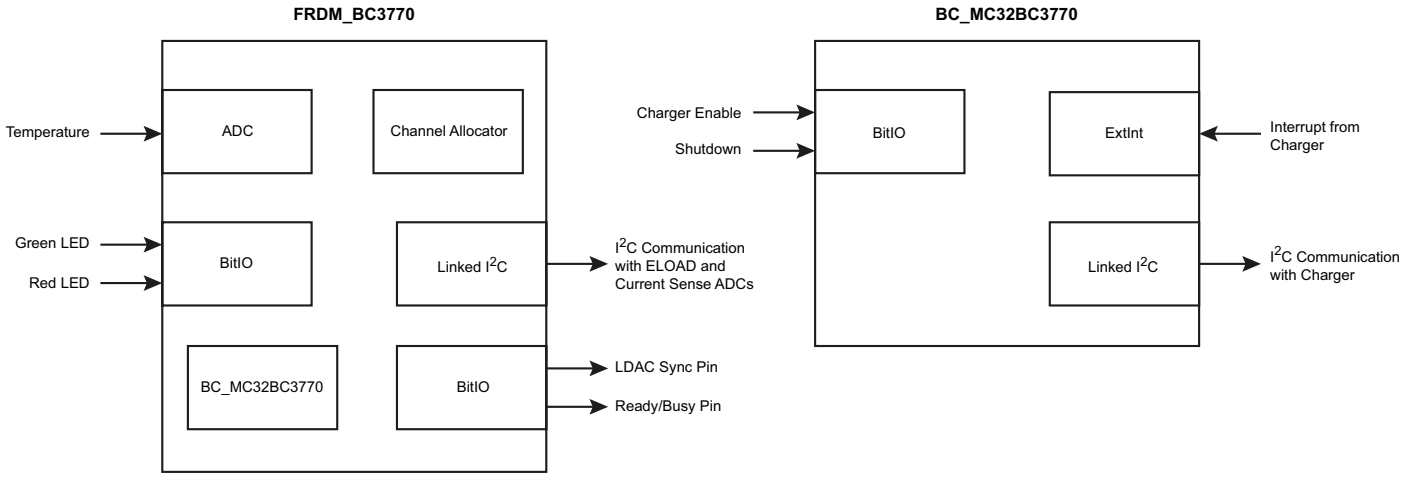

Figure 8. Block diagram of BC3770 components

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/processorexpert.

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/KDS.
2. Run the executable file and follow the instructions.

4.2 Getting the component and example projects

This procedure explains how to obtain the latest version of the components and example projects.

1. Download the zip file with components and example projects from the following link: www.nxp.com/BC3770-PExpert.
2. Unzip the downloaded file into a folder. Check to assure that the folder contains all the files listed in Table 4.

Table 4. Content of the downloaded .zip file

Folder Name	Folder Contents
Components	Components folder
BC_MC32BC3770_b15xx.PEupd	Battery charger BC3770 component
FRDM_BC3770_b15xx.PEupd	Freedom board FRDM-BC3770 component
ChannelAllocator_b15xx.PEupd	Component for ADC channel allocation
Battery_Charger_BC3770_Control	Folder containing application files used in BCF_KLxxZ_Battery_Charger_BC3770_Control_Usb_Hid example
CodeWarrior_Examples	Example projects for CodeWarrior 10.6 or above
BCF_KL25Z_BC3770_GUI_Usb_Hid	Example with BC3770_GUI for FRDM-KL25Z
BCF_KL*xx*Z_Battery_Charger_BC3770_Control_Usb_Hid	Example showing usage of BC_MC32BC3770 and FRDM_BC3770 methods with Battery_Charger_BC3770_Control application for FRDM-KL25Z, FRDM-KL26Z and FRDM-KL46Z (where *xx* is the MCU)
BCF_KL*xx*Z_Monitoring_CDC	Example showing current, voltage and temperature measurement with output to terminal for FRDM-KL25Z, FRDM-KL26Z and FRDM-KL46Z (where *xx* is the MCU)
KDS_Examples	Example projects for Kinetis Design Studio 3.0 or above
BCF_KL25Z_BC3770_GUI_Usb_Hid	Example with BC3770_GUI for FRDM-KL25Z
BCF_KL25Z_Battery_Charger_BC3770_Control_Usb_Hid_IAR	Example showing usage of BC_MC32BC3770 and FRDM_BC3770 methods with Battery_Charger_BC3770_Control application for FRDM-KL25Z and IAR Embedded Workbench.
BCF_KL*xx*Z_Battery_Charger_BC3770_Control_Usb_Hid	Example showing usage of BC_MC32BC3770 and FRDM_BC3770 methods with Battery_Charger_BC3770_Control application for FRDM-KL25Z, FRDM-KL26Z and FRDM-KL46Z (where *xx* is the MCU)
BCF_KL*xx*Z_Monitoring_CDC	Example showing current, voltage and temperature measurement with output to terminal for FRDM-KL25Z, FRDM-KL26Z and FRDM-KL46Z (where *xx* is the MCU)
Readme.pdf	Readme file with installation instructions.

4.3 Importing the components into 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 files (.PEupd) in the folder BC3770_PEx_SW\Component. Select **FRDM_BC3770_bxxxx.PEupd**, **BC_MC32BC3770_bxxxx.PEupd** and **ChannelAllocator_bxxxx.PEupd** files then click **Open** (see Figures 9).
3. Select a repository to import the components into and confirm by clicking **OK**.
   - Note that in CodeWarrior, components are imported to the predefined default repository.
   - In Kinetis Design Studio the user is offered a choice to select a repository.
4. If the import is successful, the BC3770 components are in Components Library -> SW -> User Component (see Figures 10). The components are ready to use.

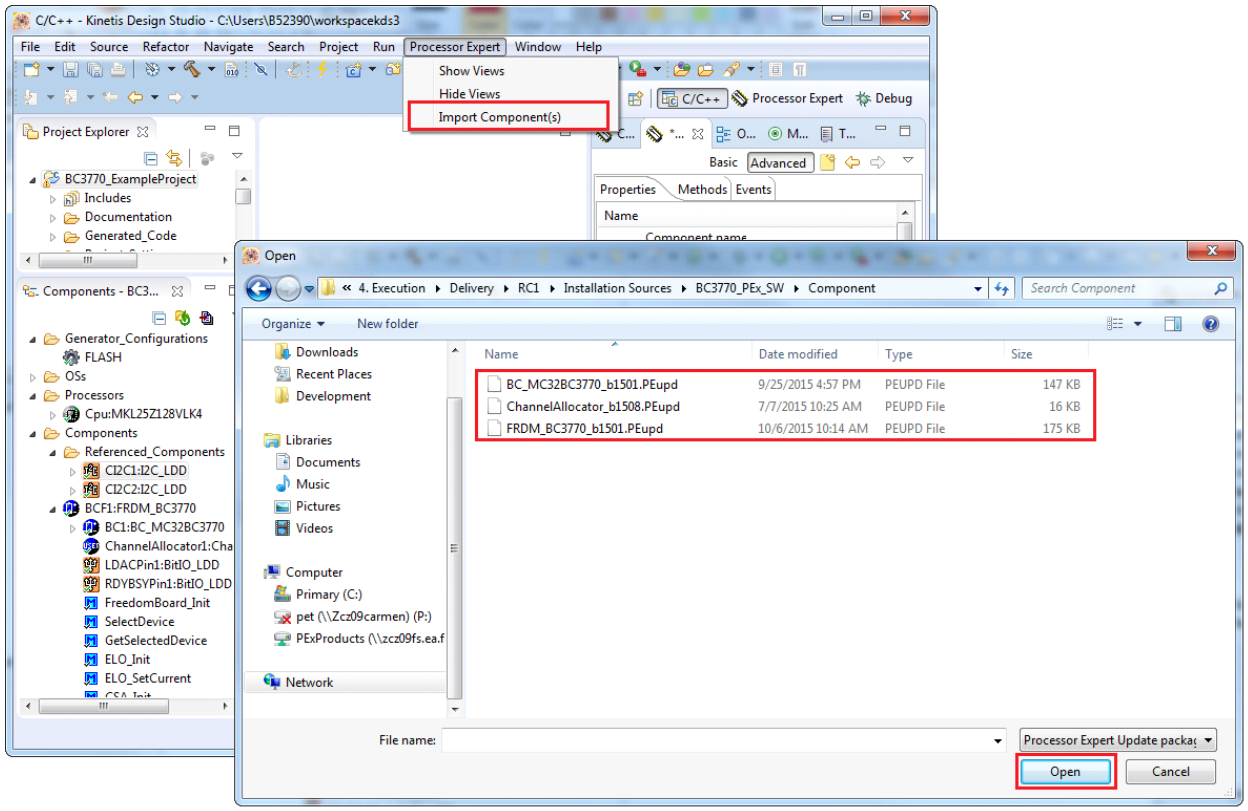

Figure 9. Importing the BC3770 components

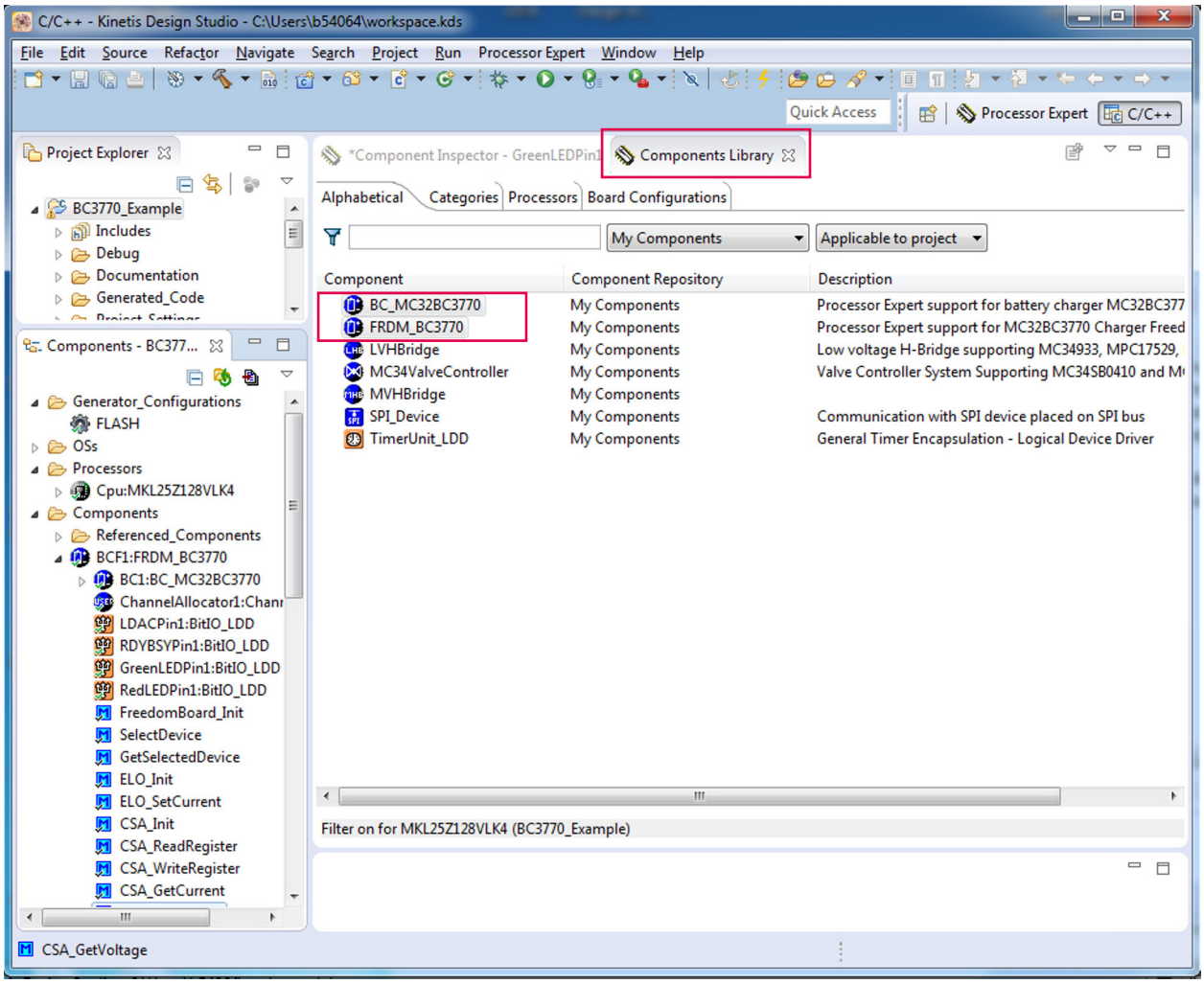

Figure 10. Component location after KDS importing

4.4 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** (see Figures 11).
2. Locate the example in folder: BC3770_PEx_SW\Examples\KDS_Examples (see Figure 12). Then click **Finish**.

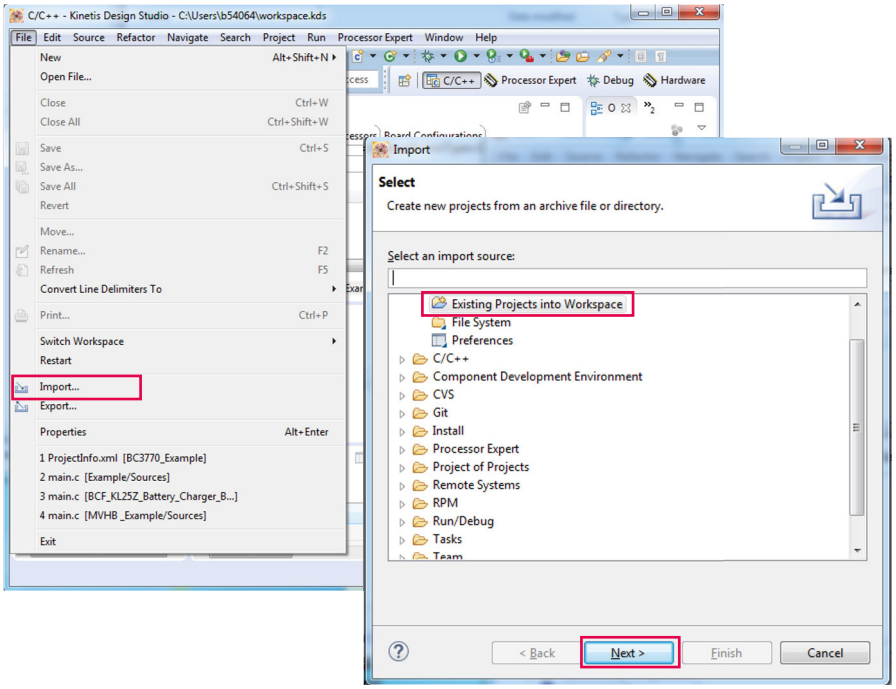

Figure 11. Example project import (a)

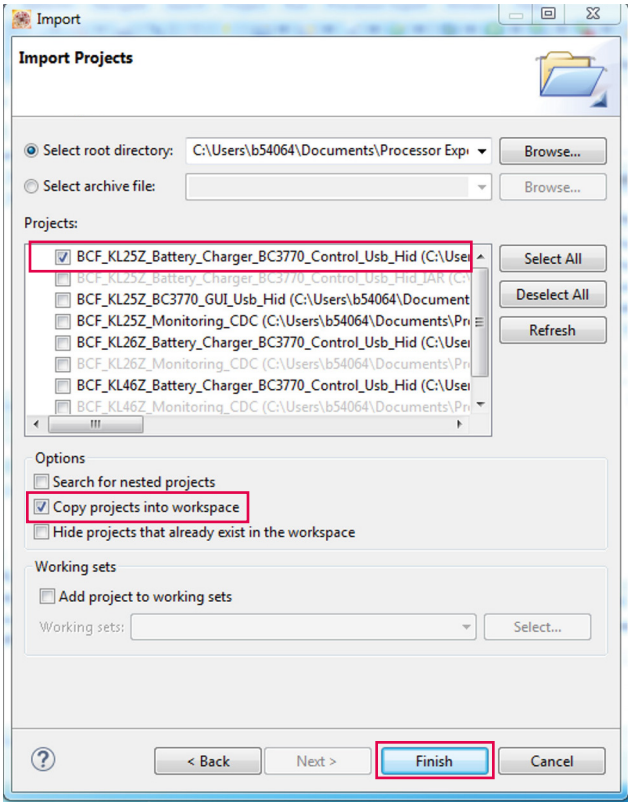

Figure 12. Example project import (b)

The project is now in the Kinetis Design Studio workspace where it can be built and run.

4.5 Creating a new project with Processor Expert and the BC3770 components

If you choose not to use the example project, the following instructions describe how to create and setup a new project that uses the a BC3770 component. If the BC3770 components are not in the Processor Expert Library, follow steps in Section 4.3, "Importing the components into 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** (Figure 13).

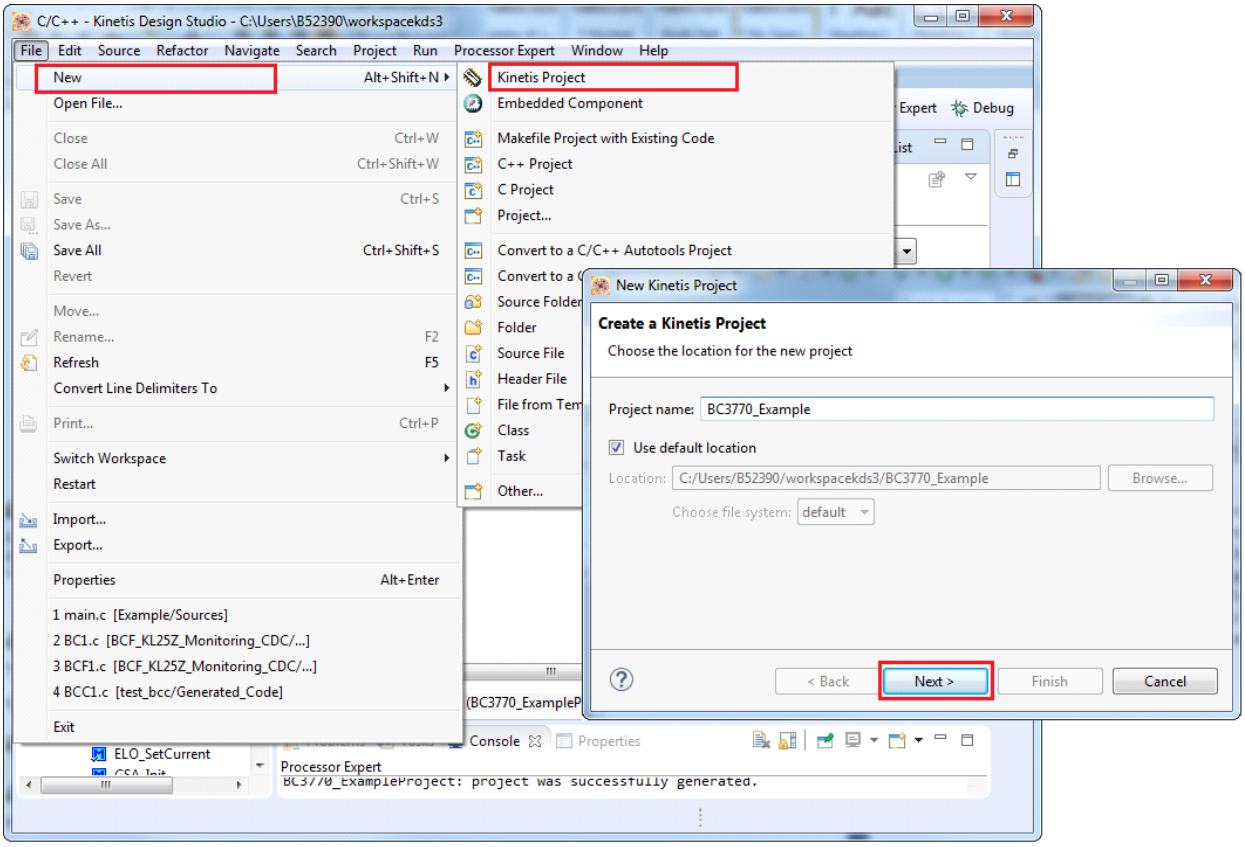

Figure 13. New project creation (a)

2. In the **Devices** dialog box, select the MCU class the project is using (in this example, MKL25Z128xxx4 is selected). Then click **Next**.
3. In the **Rapid Application Development** dialog box, select the **Processor Expert** option. Then click **Finish** (Figure 14).
4. In the **Processor Expert Target Compiler** dialog box, select a compiler (GNU C Compiler in Figure 14) and click **Finish**.

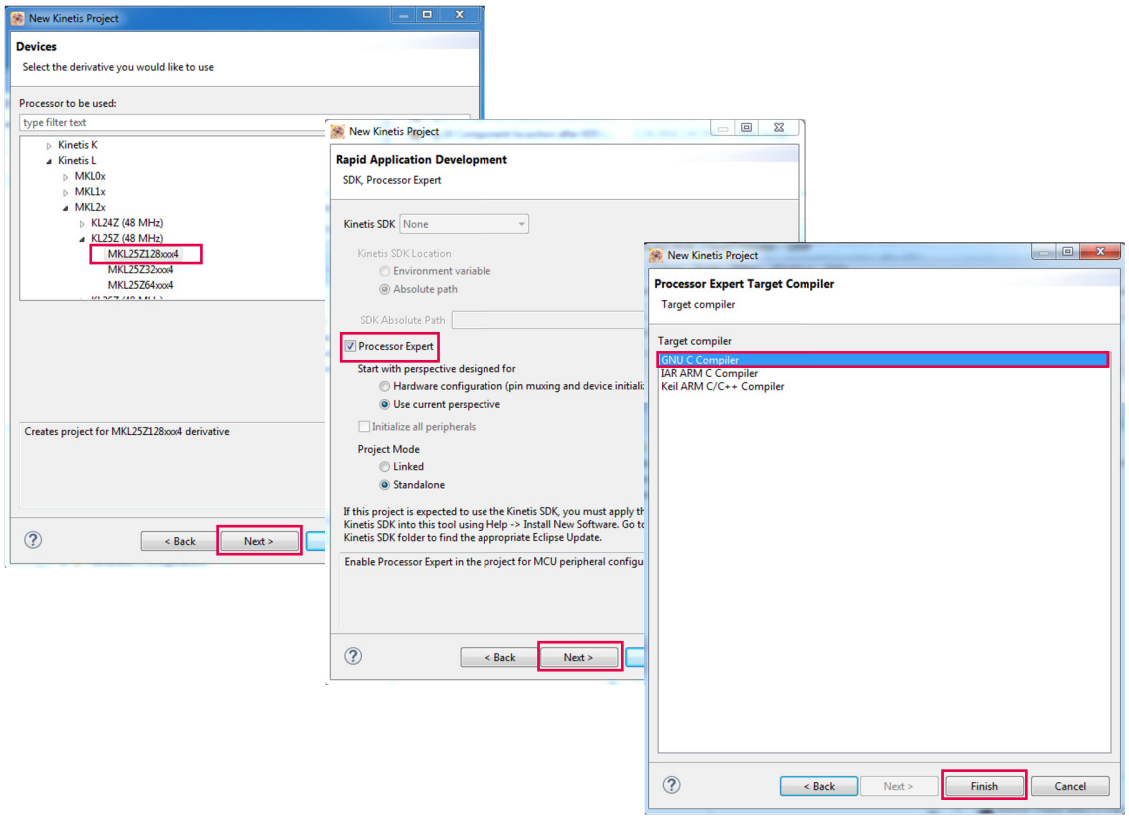

Figure 14. New project creation (b)

4.5.1 Adding the BC3770 component into the project

1. Find FRDM_BC3770 and BC_MC32BC3770 in the **Components Library** and add it into the project by double clicking, then dragging and dropping, or by right-clicking the mouse button and selecting the **Add to Project** option (see Figure 15).

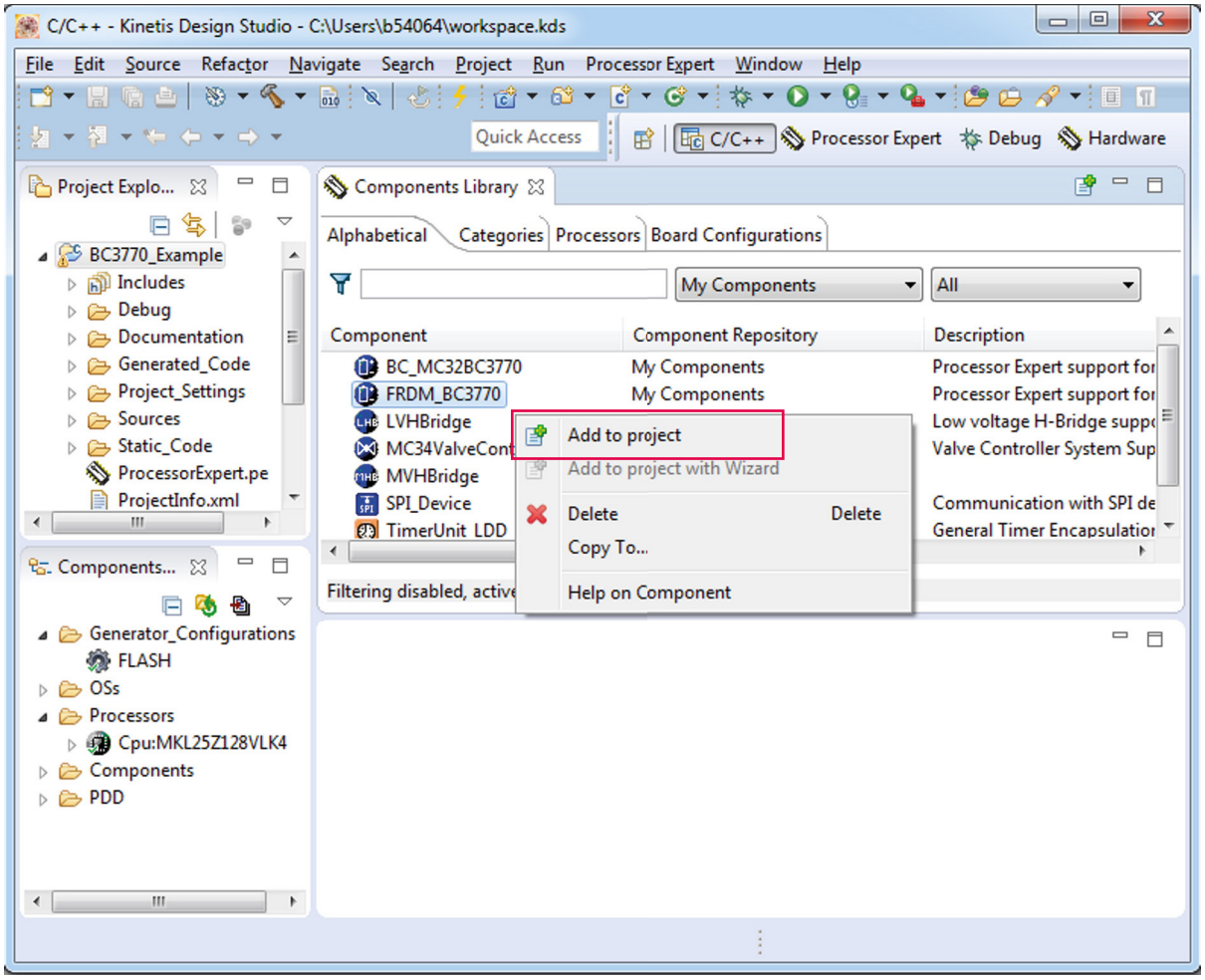

Figure 15. Add the BC3770 component to the project

2. Click the BC3770 component in the **Components** window to show the configuration in the **Component Inspector** view (see Figure 16).

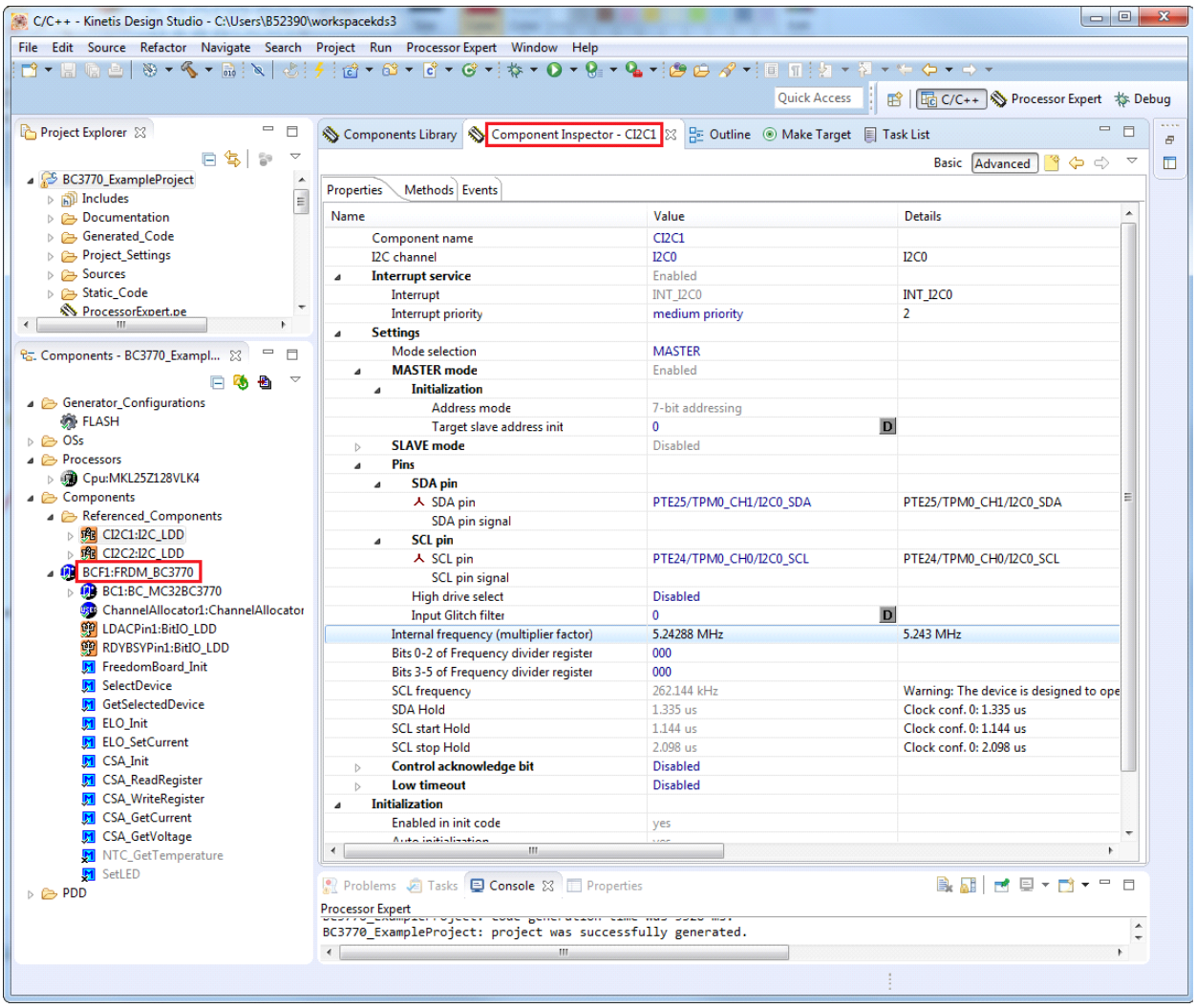

Figure 16. Component Inspector view

4.5.2 Generating driver source code

After configuring the components, the next step is to generate driver code to incorporate into the application. The process is as follows

1. Click on the **Generate Processor Expert Code** icon in the upper right corner of the **Components** panel (see Figure 17).

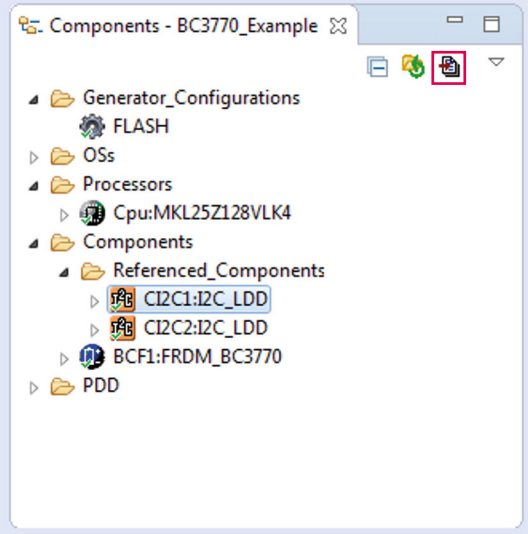

Figure 17. Generating the driver code

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

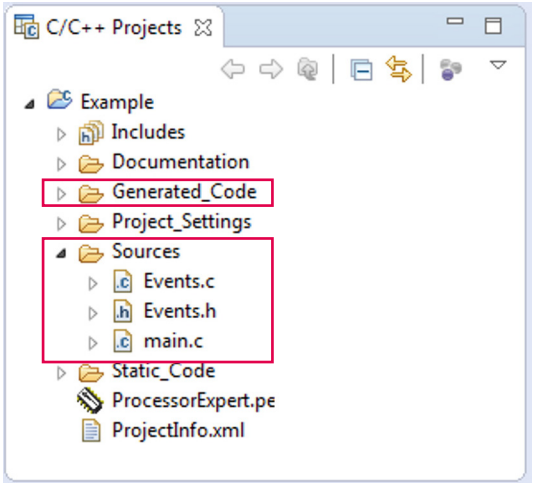

Figure 18. Source code locations

4.5.3 Writing the application code

All application code must reside in the **Sources** folder in the project directory. The user may modify the code in **main.c** and **Events.c**, but should 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 method to add to the source code.
2. Drag and drop the method directly into the source code panel.
3. Add the appropriate parameters to the method. (Hovering the mouse over the method displays a list of the required parameters.)

For example, open the FRDM_BC3770 component method list, drag and drop **SelectDevice** to **main.c** and add the necessary parameters (see Figure 19).

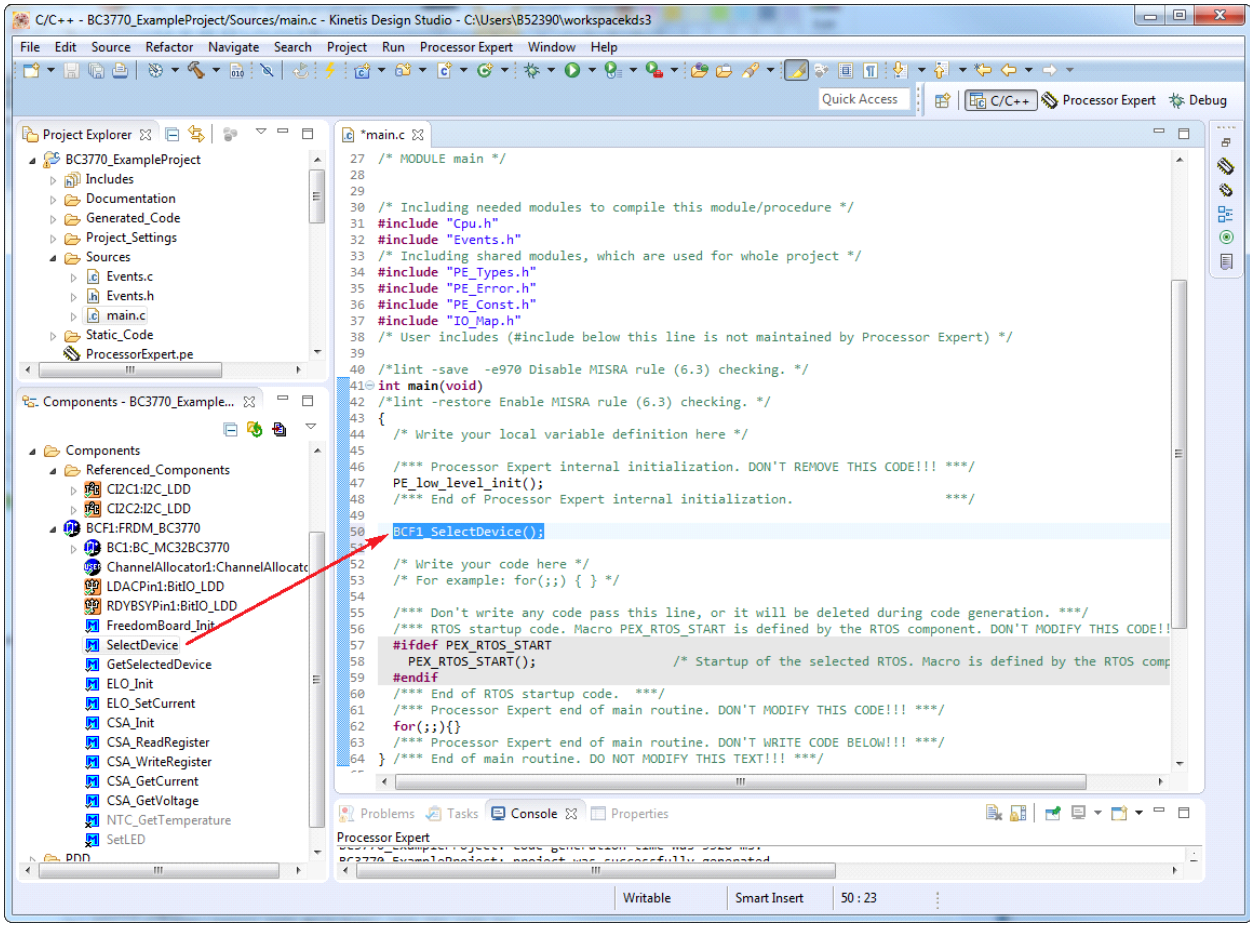

Figure 19. Adding component methods

4.5.4 Compiling, downloading and debugging

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

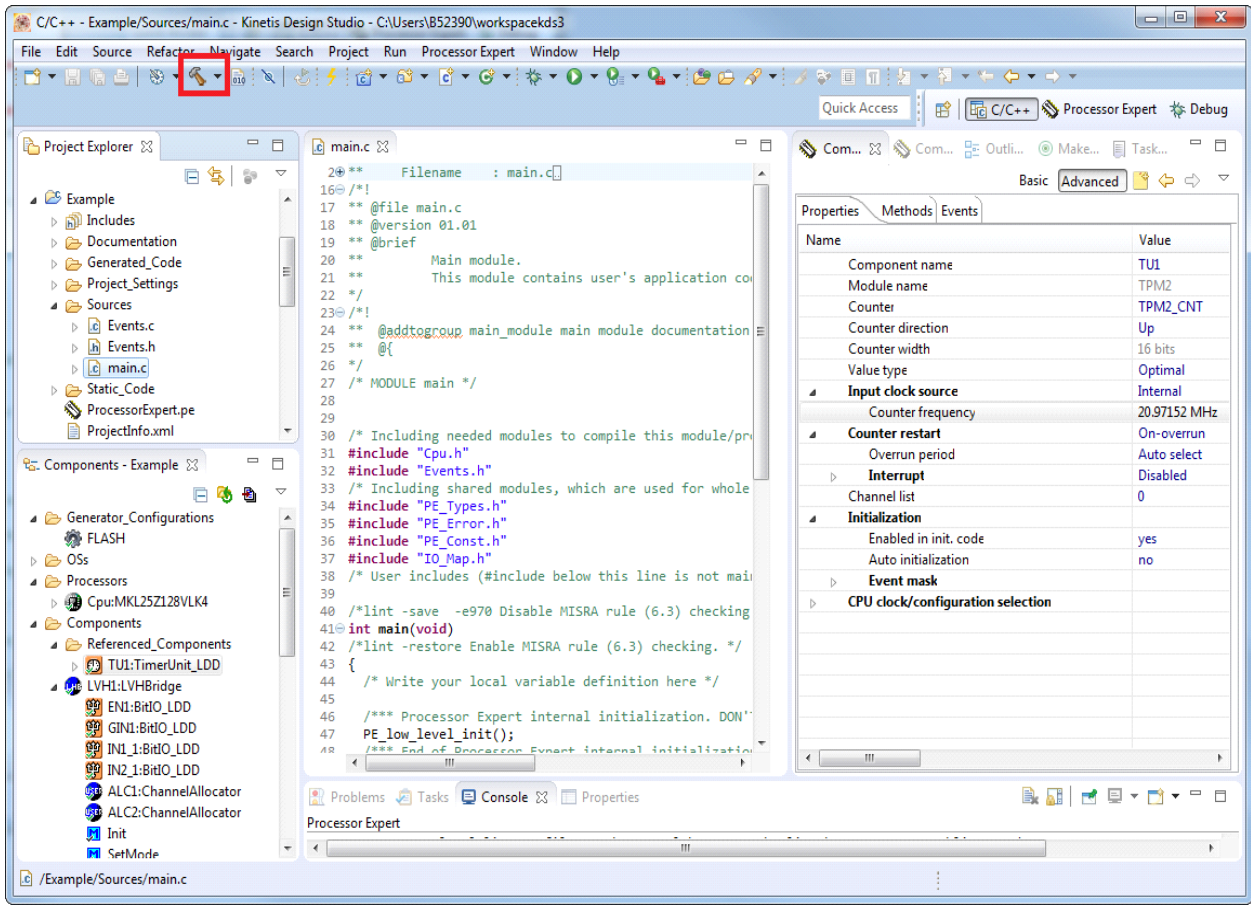

Figure 20. Compiling and downloading the application

The process for downloading an application onto a board in Kinetis Design Studio may differ according to the MCU board in use. For more information, see the Kinetis Design Studio user's guide.

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

1. Click the arrow next to the debug icon in the tool bar and select **Debug Configurations…** (Figure 21).

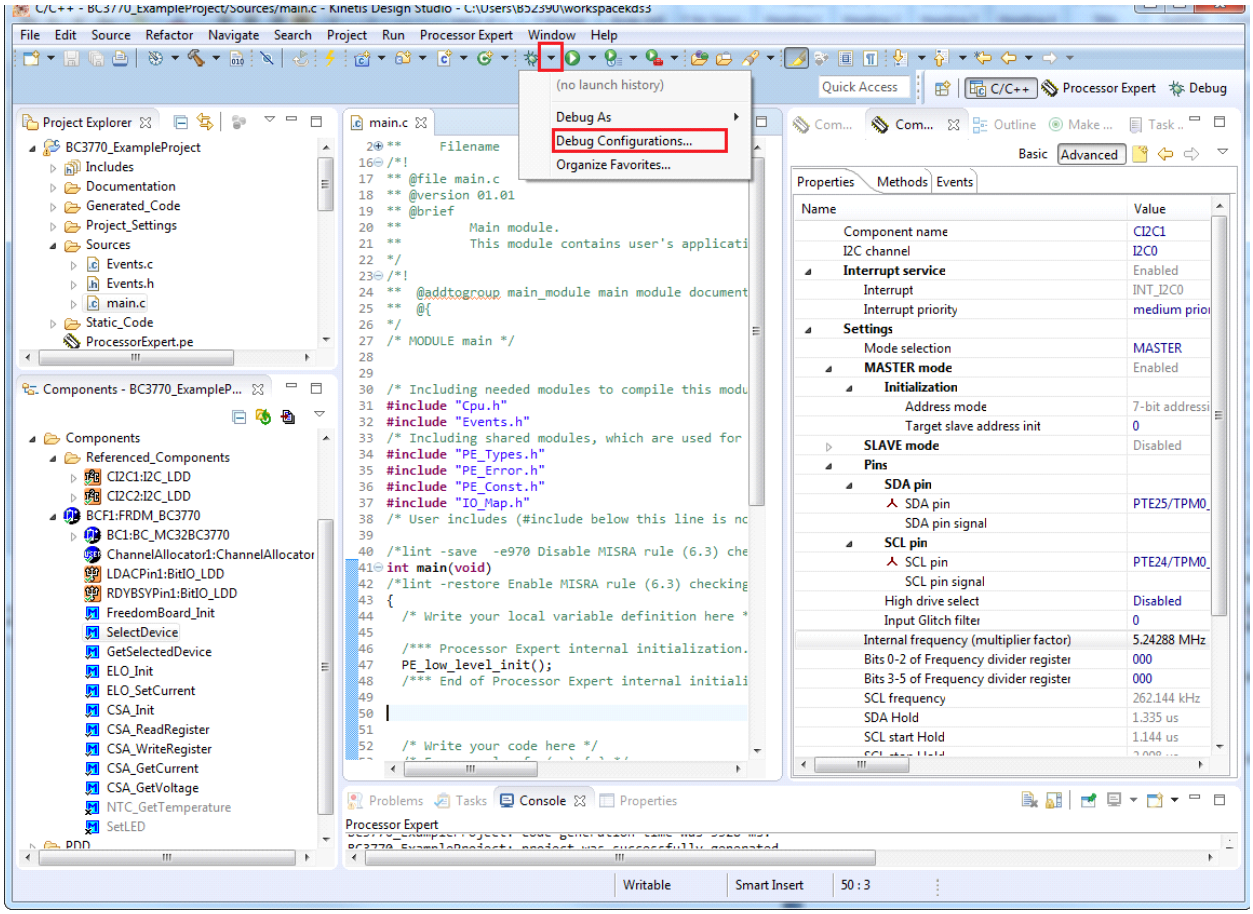

Figure 21. Downloading the application (a)

2. In the **Debug Configurations** dialog box, click **Example_Debug_PNE** under **GDB PEMicro Interface Debugging**.
3. Make sure that **C/C++ Application** contains the path to the project's .elf file (see Figure 22).

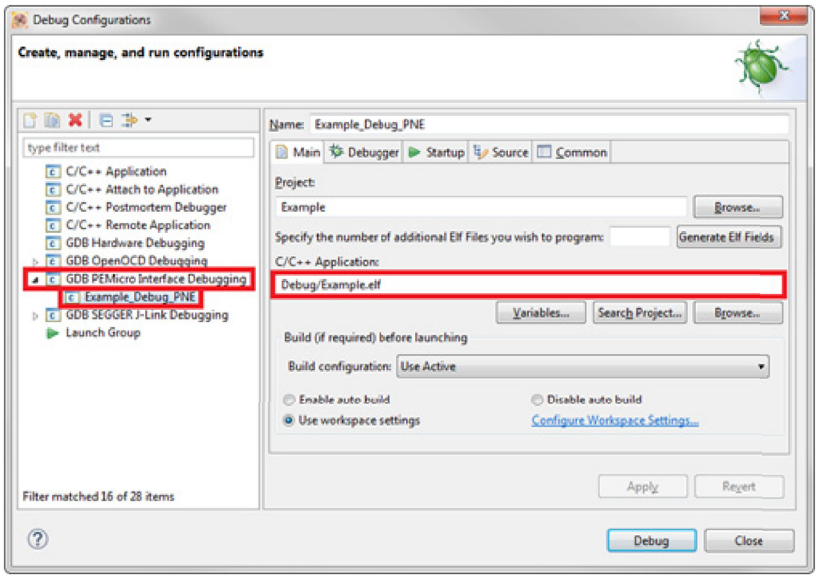

Figure 22. Downloading the application (b)

4. Click the **Debugger** tab and set the **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 23).
5. Make sure the **Target** is set to the processor you 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.

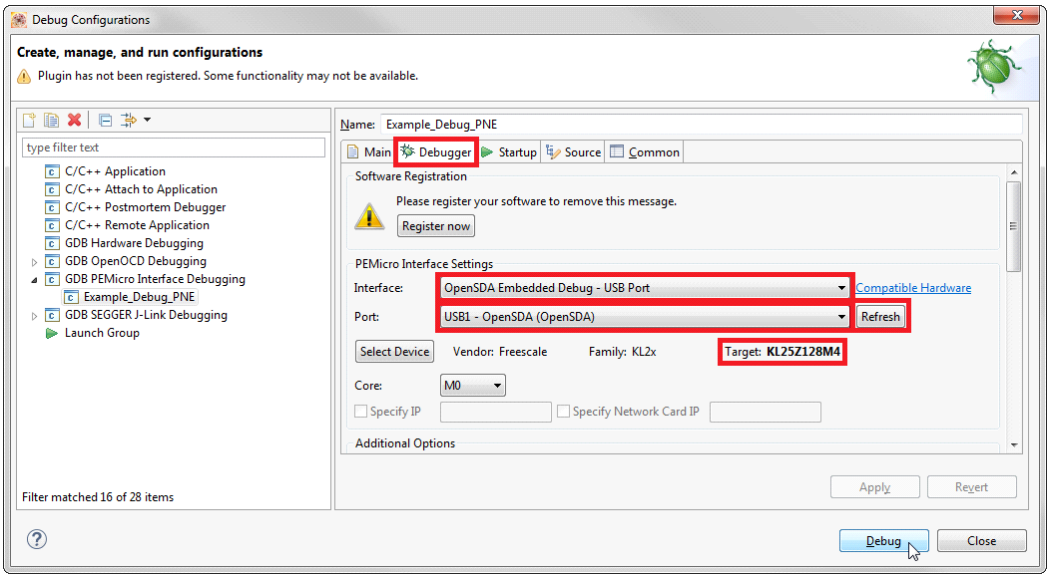

Figure 23. Downloading the application (c)

5 References

The following are URLs provide links to information on related NXP products and application solutions:

Table 5. References

NXP.com Support Pages	Description	URL
FRDM-BC3770-EVB	Tool Summary Page	www.nxp.com/FRDM-BC3770-EVB
BC3770	Product Summary Page	www.nxp.com/BC3770
Kinetis Design Studio	Software	www.nxp.com/kds
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	2/2016	• Initial release

RoHS

How to Reach Us:

Home Page:
NXP.com

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

Document Number: PEXBC3770SWUG
Rev. 1.0
2/2016

NXP