AN0821: Simplicity Studio™ C8051F85x Walkthrough

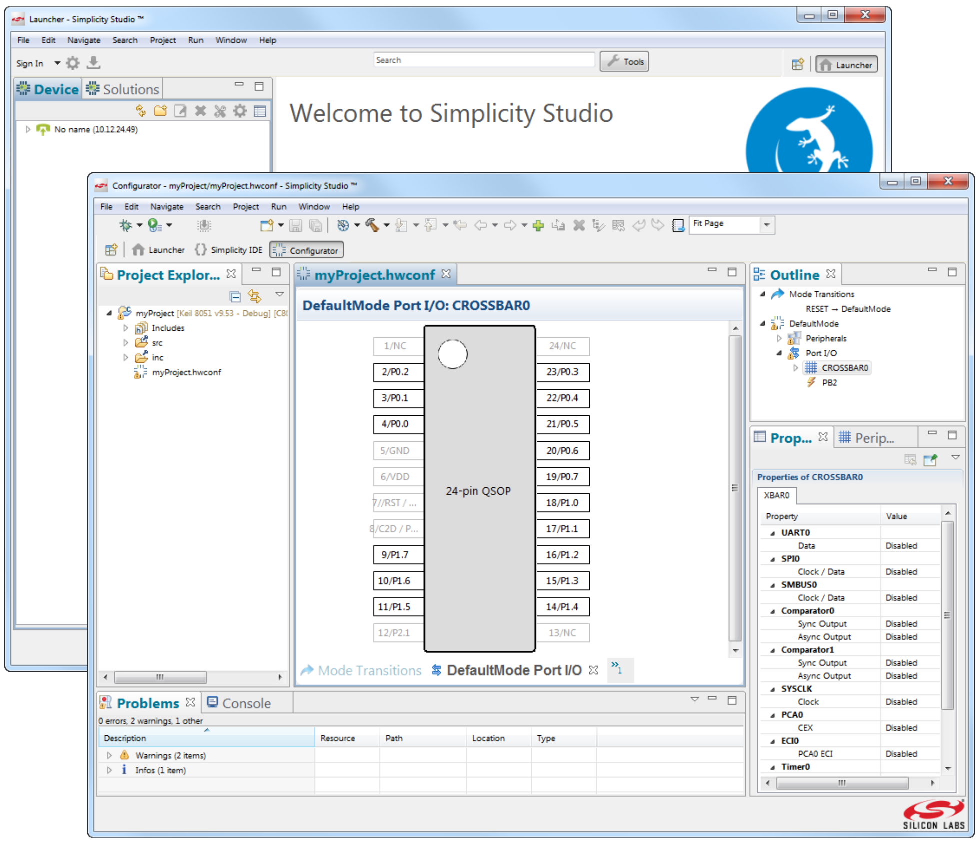
This document provides a step-by-step walkthrough that shows how to develop a basic embedded project using Simplicity Studio (IDE and Configurator) to run on the C8051F850 ToolStick Daughter Card (TOOLSTICK850-DC).

The same steps can be applied to develop a project to run on the C8051F850 UDP MCU card (UPMU-F850 MCU). The project uses a potentiometer sampled via the ADC to control the brightness of an LED by controlling the duty cycle of an output pin using the PWM. Simplicity Configurator will be used to generate hardware initialization code for hardware peripherals, and Simplicity IDE will be used to develop, build, download, and debug the firmware project.

Download and install Simplicity Studio from: http://www.silabs.com/simplicity-studio.

KEY POINTS

- Simplicity Studio makes the development process easier, faster, and more efficient.
- Simplicity Configurator enables all peripheral and pin initialization in a simple GUI.
- To improve download times and installation size on disk, Simplicity Studio downloads and installs the minimum components for your current task. The software will automatically download new components as needed.



Property	Value
UART0	
Data	Disabled
SPI0	
Clock / Data	Disabled
SMBUS0	
Clock / Data	Disabled
Comparator0	
Sync Output	Disabled
Async Output	Disabled
Comparator1	
Sync Output	Disabled
Async Output	Disabled
SYSCLK	
Clock	Disabled
PCA0	
CEX	Disabled
ECI0	
PCA0 ECI	Disabled
Timer0	

1. Relevant Documents

- Simplicity IDE Guide—In Simplicity IDE, select [**Help**]>[**Help Contents**] to display the Simplicity IDE Guide as well as any installed documentation, including data sheets and user's guides.
- Simplicity Configurator Guide—In the Simplicity IDE, select [**Help**]>[**Help Contents**] to display this guide.
- *AN0822: Simplicity Studio User's Guide*—In addition to the documentation within the tool itself, this document provides a discussion of the Simplicity Studio tool. Application Notes are available on www.silabs.com/32bit-appnotes and www.silabs.com/8bit-appnotes.
- *AN0823: Simplicity Configurator User's Guide*—In addition to the documentation within the tool itself, this document provides a discussion of the Simplicity Configurator tool. Application Notes are available on www.silabs.com/32bit-appnotes and www.silabs.com/8bit-appnotes.
- C8051F850 Data Sheet—http://www.silabs.com/products/mcu/smallmcu/pages/c8051f85x-86x.aspx (Documentation Tab)

2. Getting Started

2.1 Installing Simplicity Studio

1. Download and install Simplicity Studio from www.silabs.com/simplicity.
2. Run Simplicity Studio by selecting [**Start**]>[**Silicon Labs**]>[**Simplicity Studio**]>[**Simplicity Studio**] from the start menu or double-clicking the Simplicity Studio shortcut on the desktop.
3. The setup wizard automatically runs after Simplicity Studio starts.
4. Follow the instructions to install the software. Connect a board to install components for that board only or click the Family tab to install a series of devices.
5. Simplicity Studio can will detect if new content is needed to enable a taken action. Select [**Yes**] in these prompts to download the required functionality and continue developing.

2.2 Hardware Setup

Setup the C8051F850 ToolStick hardware:

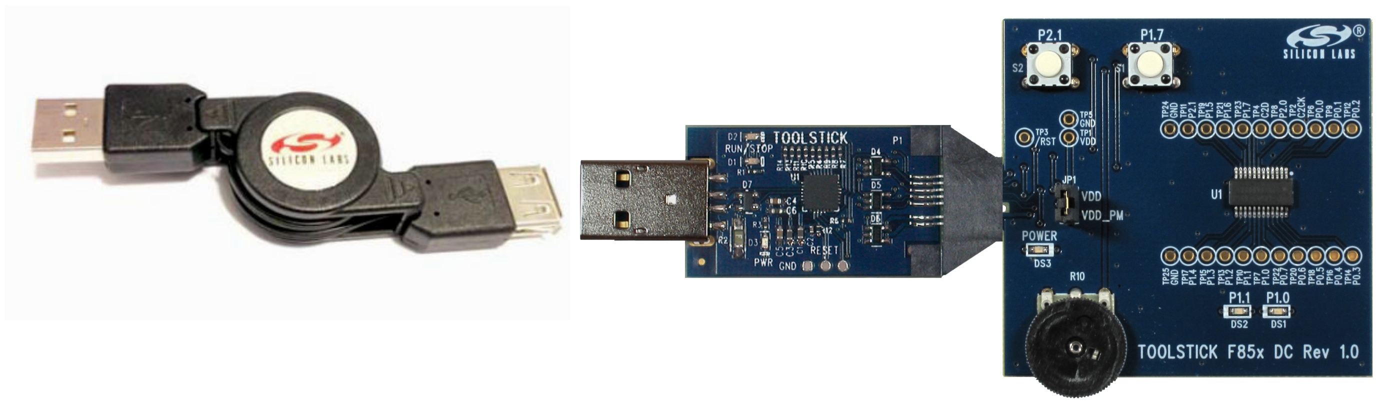


Figure 2.1. Hardware Setup

1. Verify that the shorting block on JP1 is installed on the TOOLSTICK850-DC daughter card.
2. Connect the daughter card to the ToolStick Base Adapter.
3. Connect one end of the USB extension cable to the ToolStick Base Adapter and connect the other end to a PC. The USB connection provides power to the ToolStick Debug Adapter and daughter card. The ToolStick Debug Adapter can be used to write to the C8051F850 flash and to enable on-chip debug support.

3. Using Configurator to Pulse Width Modulate (PWM) an LED

The PWM LED project uses a potentiometer sampled by the ADC to control the brightness of an LED by controlling the duty cycle of an output pin using the Programmable Counter Array (PCA) to generate a PWM waveform.

After configuring the Simplicity IDE, the next step is to create a new project using the [**New Project**] wizard. Each project has its own source files, target configuration, SDK configuration, and build configurations such as the Debug and Release build configurations. The IDE can be used to manage multiple projects in a collection called a workspace. Workspace settings are applied globally to all projects within the workspace. This can include settings such as key bindings, window preferences, and code style and formatting options. Project actions, such as build and debug are context sensitive. For example, the user must select a project in the Project Explorer view in order to build that project.

Some peripherals provide calculators such as baud rate calculators, timer overflow rate calculators, and SPI clock rate calculators that can be used to automatically calculate the necessary reload register value to generate the clock rate specified by the user. Configurator also provides real-time validation of properties to ensure that a configuration is valid before downloading code to the MCU.

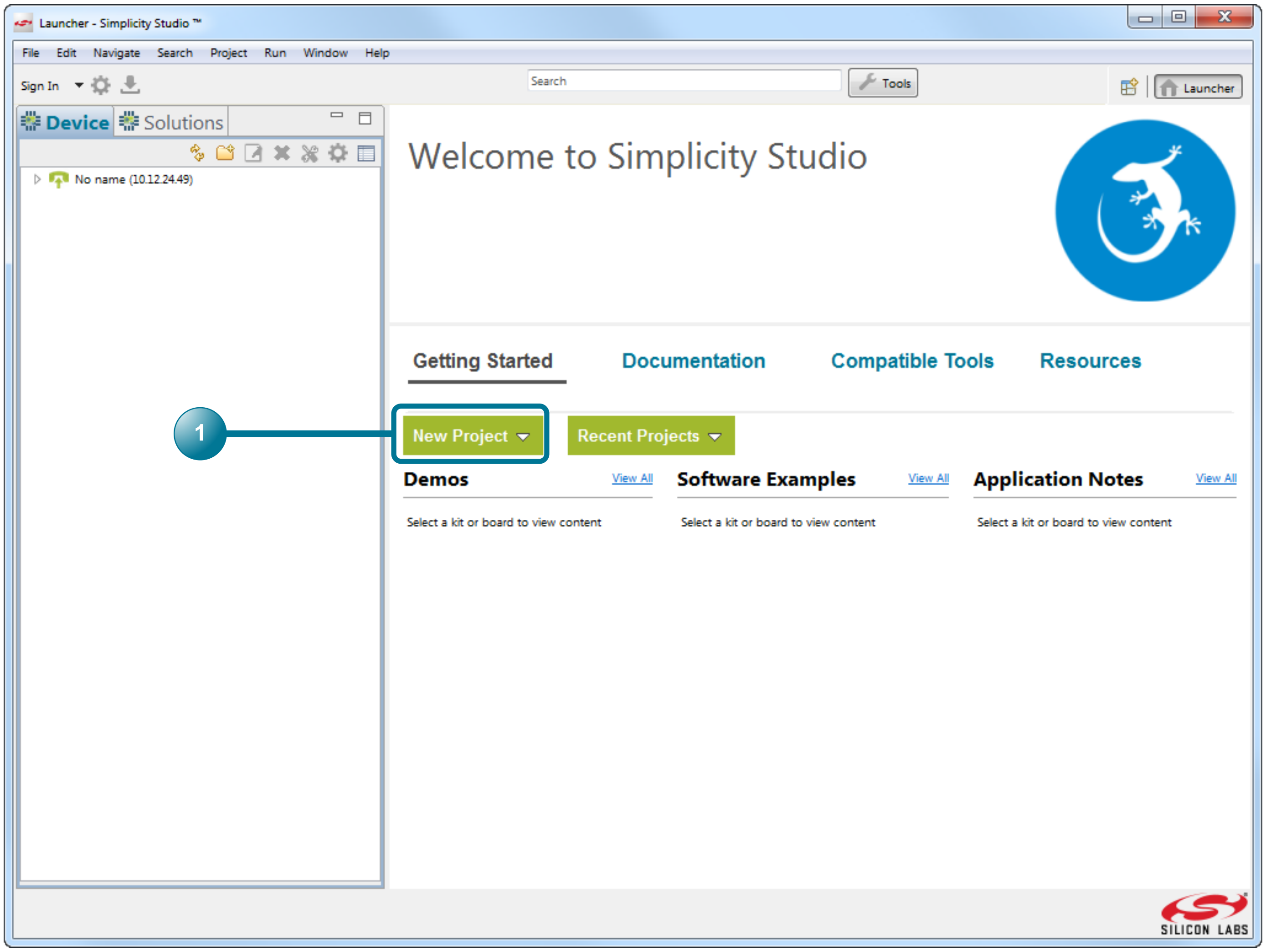
The following steps will detail the configuration of the C8051F850 peripherals to generate a PWM output based on the measured ADC input. In summary:

- Timers
  - Watchdog Timer
    - WDT Enable—Disable
  - TIMER 2 (16-bit Timer with Auto Reload, Overflow at 1 kHz)
    - Run Control—Start
    - Timer Overflow Frequency—1000
  - PCA 0 (8-bit PWM, Frequency = 1 kHz)
    - Channel 0 Capture/Compare Mode—Predefined 8~11-bit pulse modulator
    - PCA Counter/Timer Run Control—Start
- Analog
  - ADC 0 (8-bit Mode, 1x gain, Sample P1.2 on Timer 2 overflow)
    - Enable 8-bit Mode—8-bit Mode
    - Gain Control—1x gain
    - Resolution—8-bit
    - Enable ADC—Enabled
    - Start of Conversion—Timer 2 overflow
    - Positive Input Selection—ADC0.10 (P1.2)
  - Voltage Reference
    - Select Voltage Reference—VDD pin
- Core
  - Interrupts
    - Enable ADC0 Conversion Complete Interrupt—Enabled
    - Enable All Interrupts—Enabled
- Port I/O
  - Peripheral Mapping
    - PCA0_CEX0—Checked
  - Enable Crossbar—Enabled
  - P0.0 - P0.7
    - Skip—Skipped
  - P1.0
    - IOMode—Digital Push-Pull Output
  - P1.2
    - IOMode—Analog I/O
    - Skip—Skipped

After creating a new MCU project using Simplicity Configurator, Studio will automatically switch to the Configuration perspective. This perspective is tailored specifically for use with Configurator to initialize MCU peripherals. Special views will automatically be displayed in this perspective that allow peripheral registers to be configured. The default tab in a new Configurator project is the Port I/O tab.

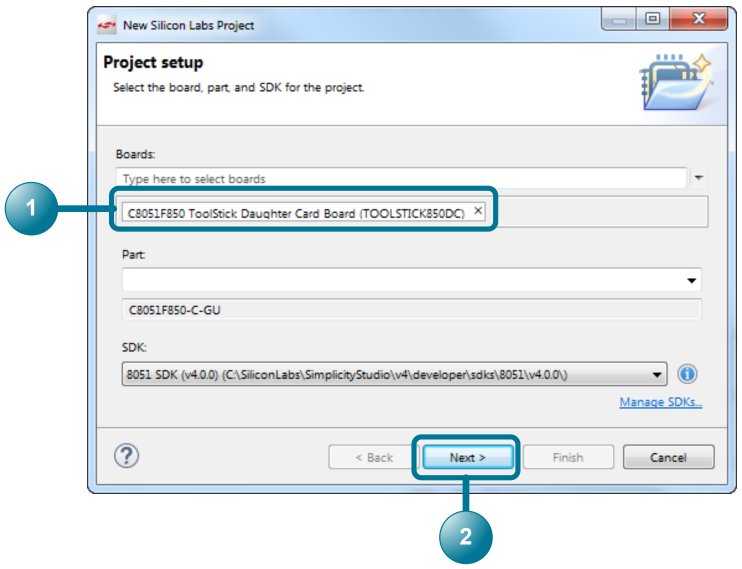
3.1 Step 1—Create a New Project

Create a new project by clicking [**New Project**] from the main landing page.



3.2 Step 2—Complete the New Project Wizard

1. In the Project setup step, select [**C8051F850 ToolStick Daughter Card Board (TOOLSTICK850DC)**] from the combo box. This will automatically select the C8051F850 part in the Part combo box.
2. Click the [**Next**] button.
3. Select the [**Simplicity Configurator Program**] radio button from the Project Type list. Simplicity Studio Configurator generates hardware initialization code using a graphical interface.
4. Click the [**Next**] button.
5. In the Project Configuration step, enter your project name in the [**Project Name**] text box.
6. Click the [**Finish**] button.


New Silicon Labs Project

Project setup

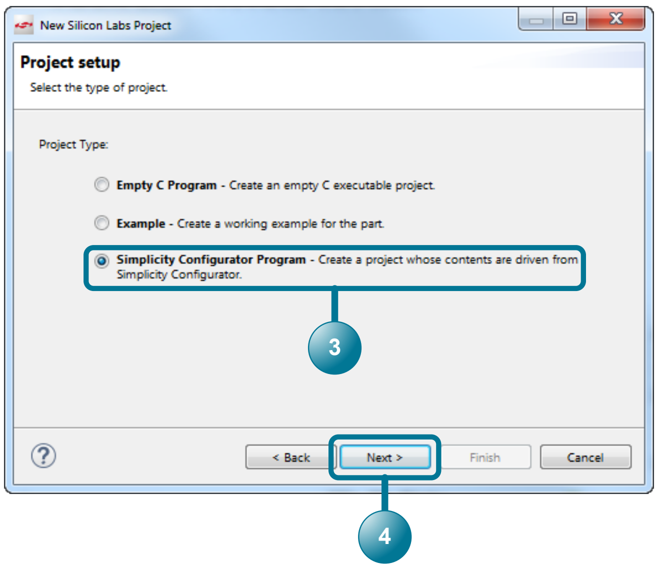
Select the board, part, and SDK for the project.

Boards: Type here to select boards

C8051F850 ToolStick Daughter Card Board (TOOLSTICK850DC)

Part: C8051F850-C-GU

SDK: 8051 SDK (v4.0.0) (C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\8051\v4.0.0\)

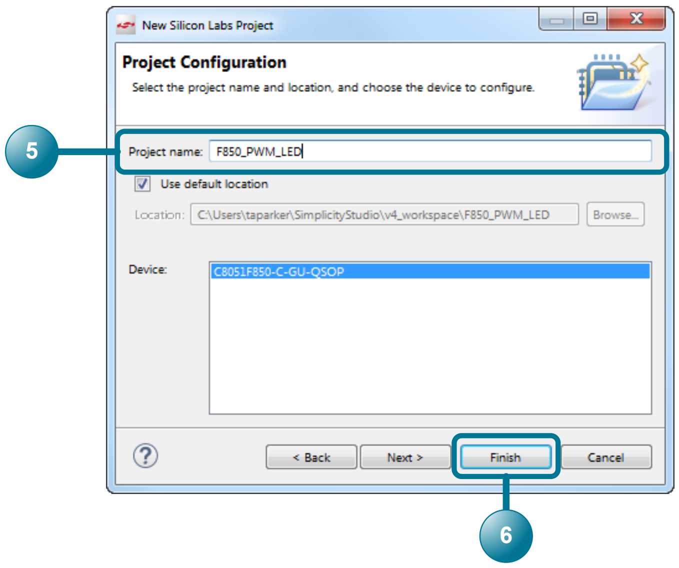

New Silicon Labs Project

Project setup

Select the type of project.

Project Type:

- **Empty C Program** - Create an empty C executable project.
- **Example** - Create a working example for the part.
- [✓ **Simplicity Configurator Program** - Create a project whose contents are driven from Simplicity Configurator.]


New Silicon Labs Project

Project Configuration

Select the project name and location, and choose the device to configure.

Project name: F850_PWM_LED

[✓] Use default location

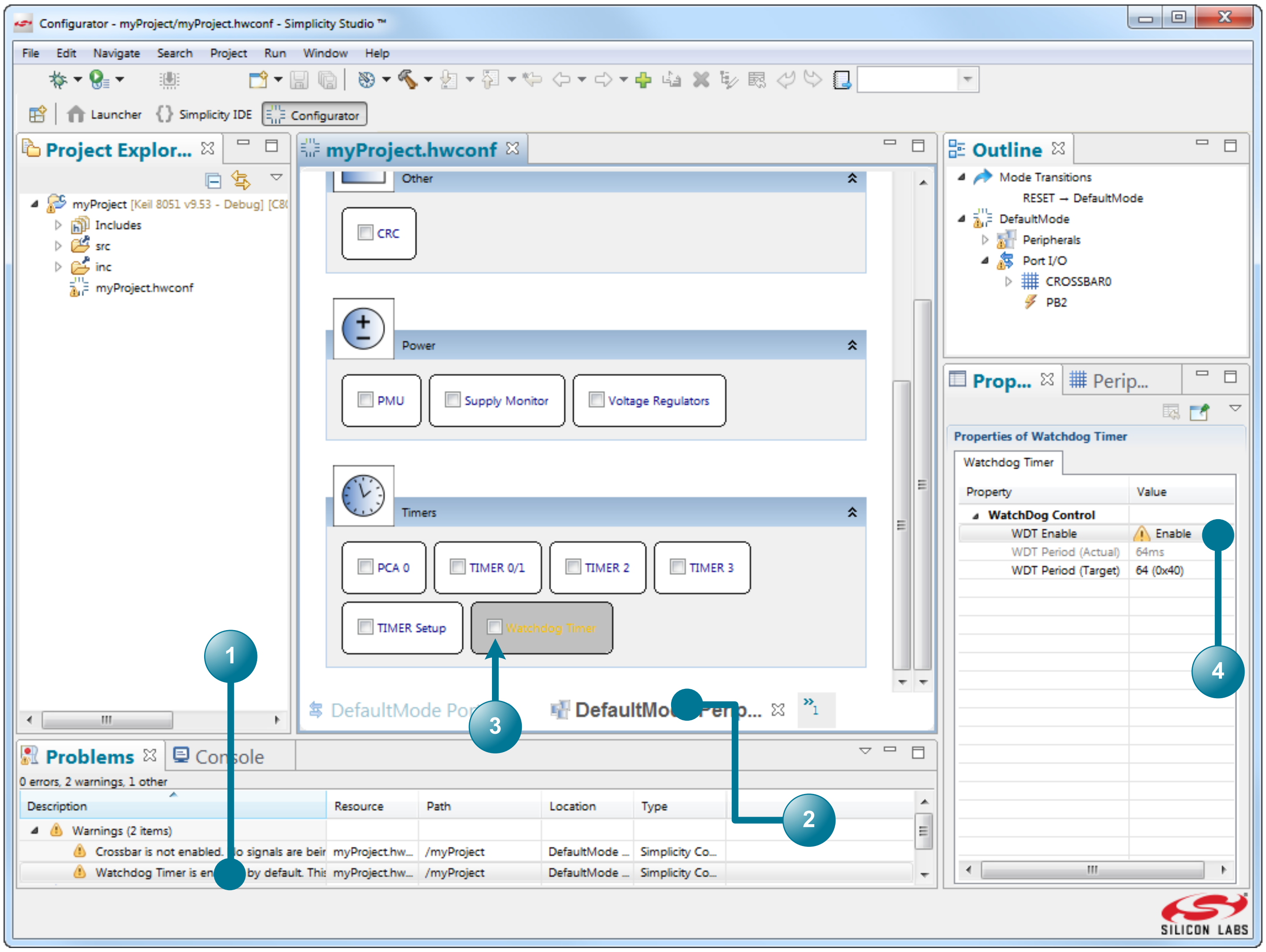
Location: C:\Users\taparker\SimplicityStudio\v4_workspace\F850_PWM_LED

Device: C8051F850-C-GU-QSOP

3.3 Step 3—Disabling the Watchdog

Studio will automatically validate register configuration values and display any errors or warnings in the [**Problems**] view. Disable the watchdog timer to fix the warning:

1. Double clicking on a warning or error will automatically display the property that raised the problem. Double click on the Watchdog timer warning to automatically switch to the [**Peripherals**] editor and display the [**Watchdog Timer**] properties in the [**Properties**] view.
2. The [**Peripherals**] tab shows all of the hardware peripherals that Configurator can generate initialization code for. Peripherals are grouped by categories such as Analog, Communications, Core, and Timers.
3. Check the check box next to a peripheral to indicate that Configurator should generate code for the checked peripheral module. Selecting a peripheral will display the peripheral properties for the selected peripheral in the [**Properties**] view.
4. Disable the watchdog timer to prevent unwanted device resets from occurring. Select the [**WDT Enable**] property in the [**Properties**] view and select [**Disable**] from the combo box. Press enter or deselect the property to confirm the new property value.
5. The watchdog timer warning in the [**Problems**] view should now disappear, indicating that the problem has been solved.



Properties of Watchdog Timer

Watchdog Timer

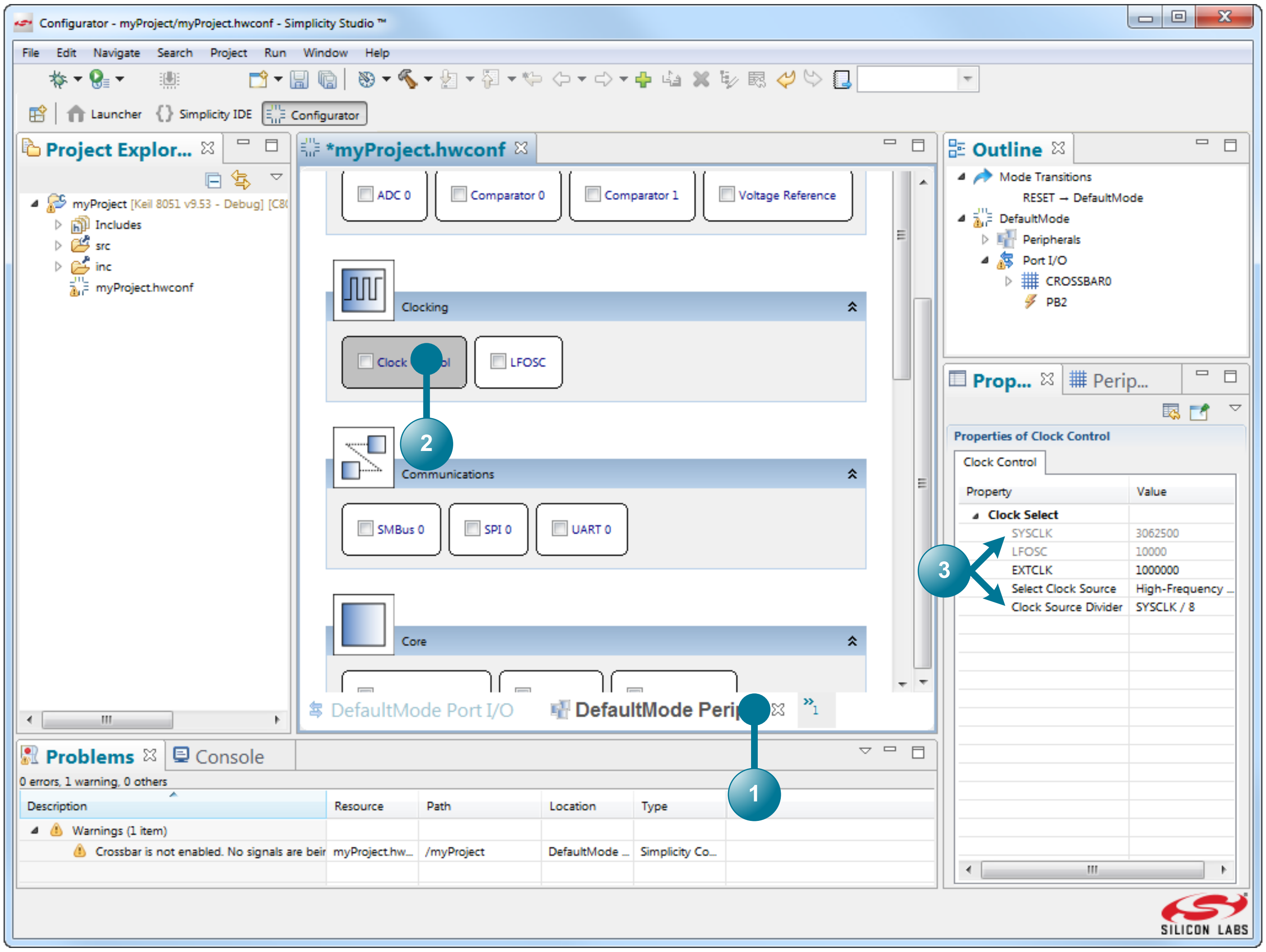
Property	Value
WatchDog Control	
WDT Enable	Enable
WDT Period (Actual)	64ms
WDT Period (Target)	64 (0x40)

0 errors, 2 warnings, 1 other

Description	Resource	Path	Location	Type
Warnings (2 items)				
Crossbar is not enabled. lo signals are beir	myProject.hw...	/myProject	DefaultMode ...	Simplicity Co...
Watchdog Timer is en by default. Thi	myProject.hw...	/myProject	DefaultMode ...	Simplicity Co...

3.4 Step 4—Confirm that the MCU system clock is configured as SYSCLK / 8 or 3.0625 MHz

1. Select the [**Peripherals**] tab, if it's not already selected.
2. Select the [**Clock Control**] box from the [**Clocking**] group to display the properties in the [**Properties**] view.
3. Verify that the [**Clock Source Divider**] property is set to [**SYSCLK / 8**] and [**SYSCLK**] reports a frequency of [**3062500**].



Property	Value
Clock Select	
SYSCLK	3062500
LFOSC	10000
EXTCLK	1000000
Select Clock Source	High-Frequency ...
Clock Source Divider	SYSCLK / 8

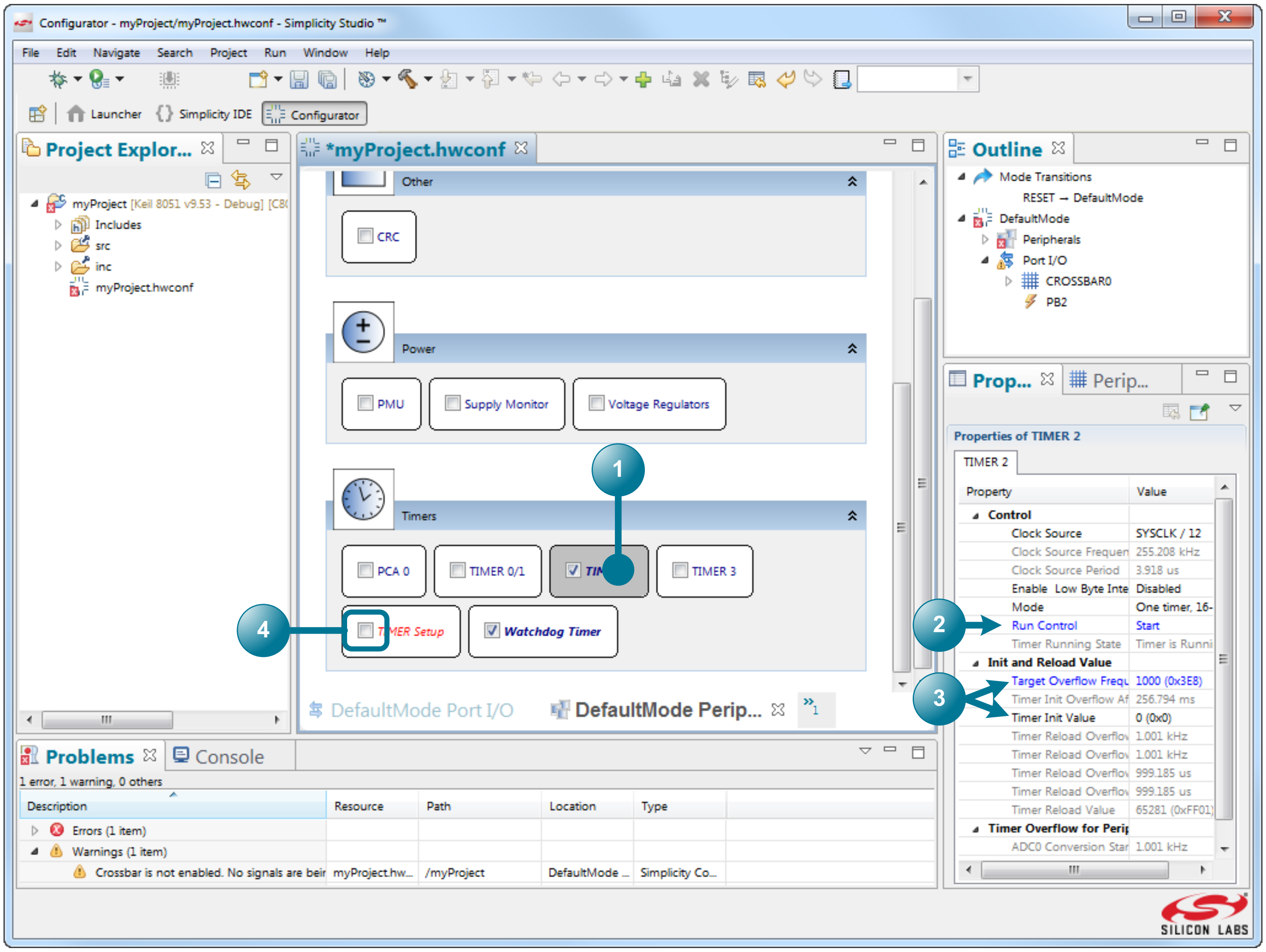
0 errors, 1 warning, 0 others

Description	Resource	Path	Location	Type
Warnings (1 item)				
Crossbar is not enabled. No signals are beir	myProject.hw...	/myProject	DefaultMode ...	Simplicity Co...

3.5 Step 5—Configuring the ADC Start-of-Conversion Source

Next, configure Timer 2 to overflow at 1 kHz. This timer will serve as the ADC start-of-conversion source.

1. Select [**TIMER 2**] from the [**Timers**] group.
2. Set the [**Run Control**] property to [**Start**].
3. Set the [**Timer Overflow Frequency**] property to [**1000**] for 1 kHz. This will change the read-only [**Timer Reload Overflow Frequency**] to ~1 kHz.
4. Check the [**TIMER Setup**] check box to enable code generation and remove the error.
5. Timer 2 is now clocked from the 3.0625 MHz system clock divided by 12 and the 16-bit Timer 2 counter will overflow approximately every 1 ms (1 kHz).



Properties of TIMER 2

TIMER 2

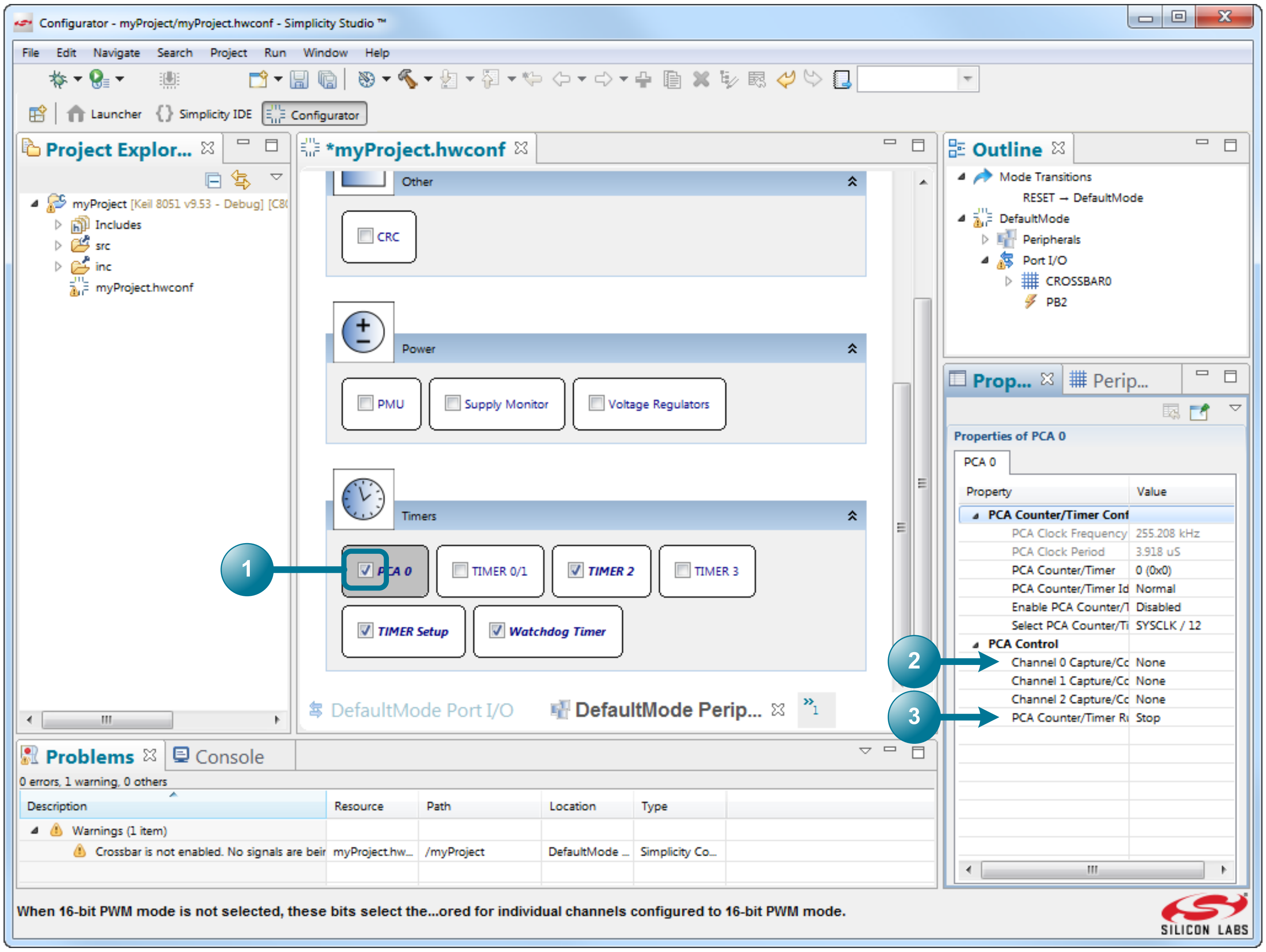
Property	Value
Control	
Clock Source	SYSCLK / 12
Clock Source Frequen	255.208 kHz
Clock Source Period	3.918 us
Enable Low Byte Inte	Disabled
Mode	One timer, 16-
Run Control	Start
Timer Running State	Timer is Runni
Init and Reload Value	
Target Overflow Frequ	1000 (0x3E8)
Timer Init Overflow Af	256.794 ms
Timer Init Value	0 (0x0)
Timer Reload Overflov	1.001 kHz
Timer Reload Overflov	1.001 kHz
Timer Reload Overflov	999.185 us
Timer Reload Overflov	999.185 us
Timer Reload Value	65281 (0xFF01)
Timer Overflow for Perip	
ADC0 Conversion Star	1.001 kHz

1 error, 1 warning, 0 others

Description	Resource	Path	Location	Type
Errors (1 item)				
Warnings (1 item)				
Crossbar is not enabled. No signals are beir	myProject.hw...	/myProject	DefaultMode ...	Simplicity Co...

3.6 Step 6—Configure PCA 0 to output an 8-bit PWM signal on P1.0 at 1 kHz

1. Select [**PCA 0**] from the [**Timers**] group.
2. Set the [**Channel 0 Capture/Compare Mode**] property to [**Predefined 8~11-bit pulse modulator**].
3. Set the [**PCA Counter/Timer Run Control**] property to [**Start**].



Properties of PCA 0

PCA 0

Property	Value
PCA Counter/Timer Conf	
PCA Clock Frequency	255.208 kHz
PCA Clock Period	3.918 uS
PCA Counter/Timer	0 (0x0)
PCA Counter/Timer Id	Normal
Enable PCA Counter/T	Disabled
Select PCA Counter/Ti	SYSCLK / 12
PCA Control	
Channel 0 Capture/Cc	None
Channel 1 Capture/Cc	None
Channel 2 Capture/Cc	None
PCA Counter/Timer Ru	Stop

0 errors, 1 warning, 0 others

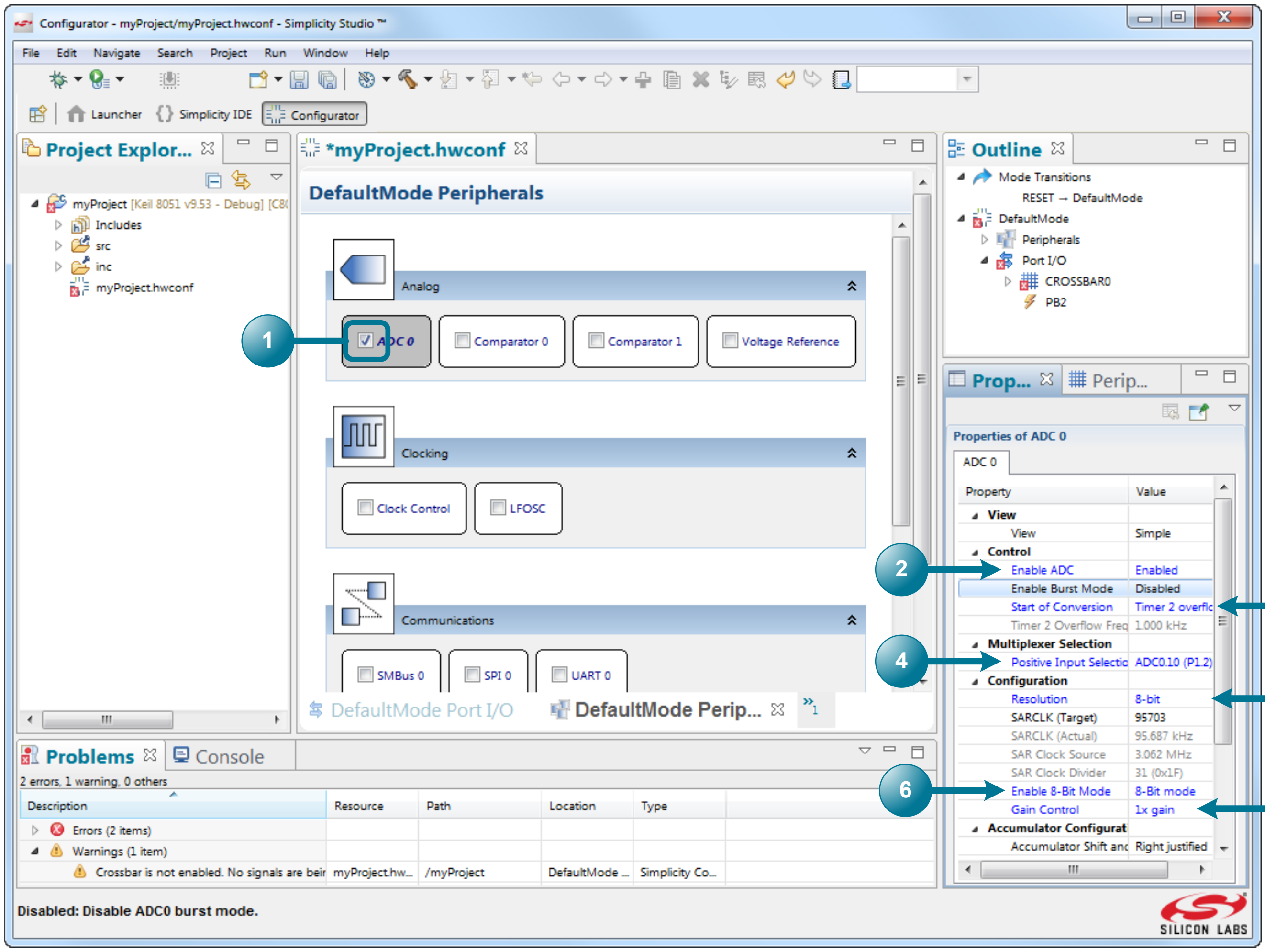
Description	Resource	Path	Location	Type
Warnings (1 item)				
Crossbar is not enabled. No signals are beir	myProject.hw...	/myProject	DefaultMode ...	Simplicity Co...

When 16-bit PWM mode is not selected, these bits select the...ored for individual channels configured to 16-bit PWM mode.

3.7 Step 7—Configuring the ADC

Next, configure ADC 0 to acquire 8-bit samples on P1.2, the potentiometer pin, on Timer 2 overflow:

1. Select [**ADC 0**] from the [**Analog**] group.
2. Set the [**Enable ADC**] property to [**Enabled**].
3. Set the [**Start of Conversion**] property to [**Timer 2 overflow**].
4. Set the [**Positive Input Selection**] property to [**ADC0.10 (P1.2)**].
5. Set the [**Resolution**] property to [**8-bit**].
6. Set the [**Enable 8-bit Mode**] property to [**8-bit mode**].
7. Set the [**Gain Control**] property to [**1x gain**].



Property	Value
View	
View	Simple
Control	
Enable ADC	Enabled
Enable Burst Mode	Disabled
Start of Conversion	Timer 2 overflc
Timer 2 Overflow Freq	1.000 kHz
Multiplexer Selection	
Positive Input Selectic	ADC0.10 (P1.2)
Configuration	
Resolution	8-bit
SARCLK (Target)	95703
SARCLK (Actual)	95.687 kHz
SAR Clock Source	3.062 MHz
SAR Clock Divider	31 (0x1F)
Enable 8-Bit Mode	8-Bit mode
Gain Control	1x gain
Accumulator Configurat	
Accumulator Shift anc	Right justified

Problems | Console

2 errors, 1 warning, 0 others

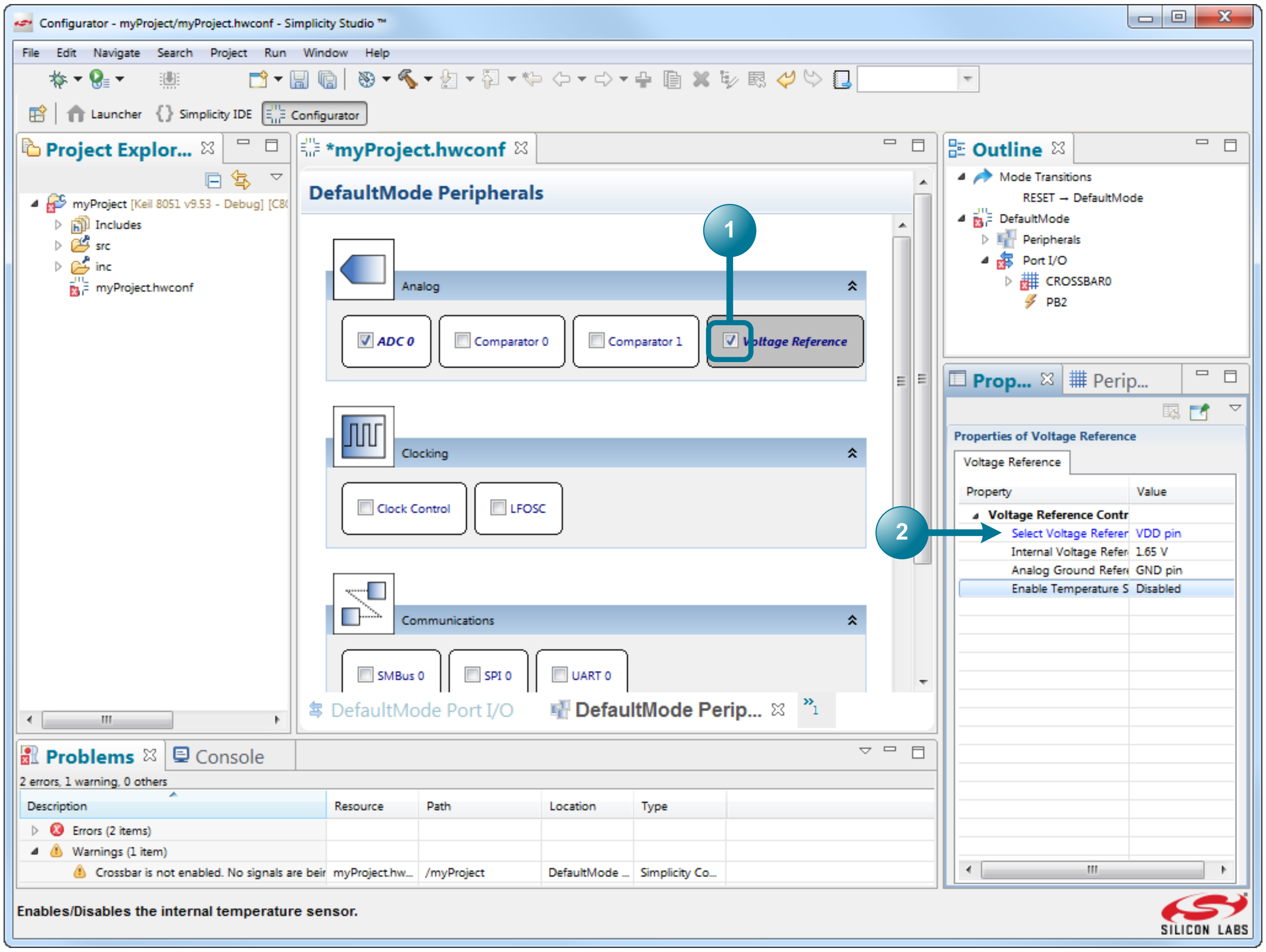
Description	Resource	Path	Location	Type
Errors (2 items)				
Warnings (1 item)				
Crossbar is not enabled. No signals are beir	myProject.hw...	/myProject	DefaultMode ...	Simplicity Co...

Disabled: Disable ADC0 burst mode.

3.8 Step 8—Configure the Voltage Reference

Configure the voltage reference for the ADC to use VDD:

1. Select [**Voltage Reference**] from the [**Analog**] group.
2. Set the [**Select Voltage Reference**] property to [**VDD pin**].



Property	Value
Voltage Reference Contr	
Select Voltage Referer	VDD pin
Internal Voltage Refer	1.65 V
Analog Ground Refer	GND pin
Enable Temperature S	Disabled

2 errors, 1 warning, 0 others

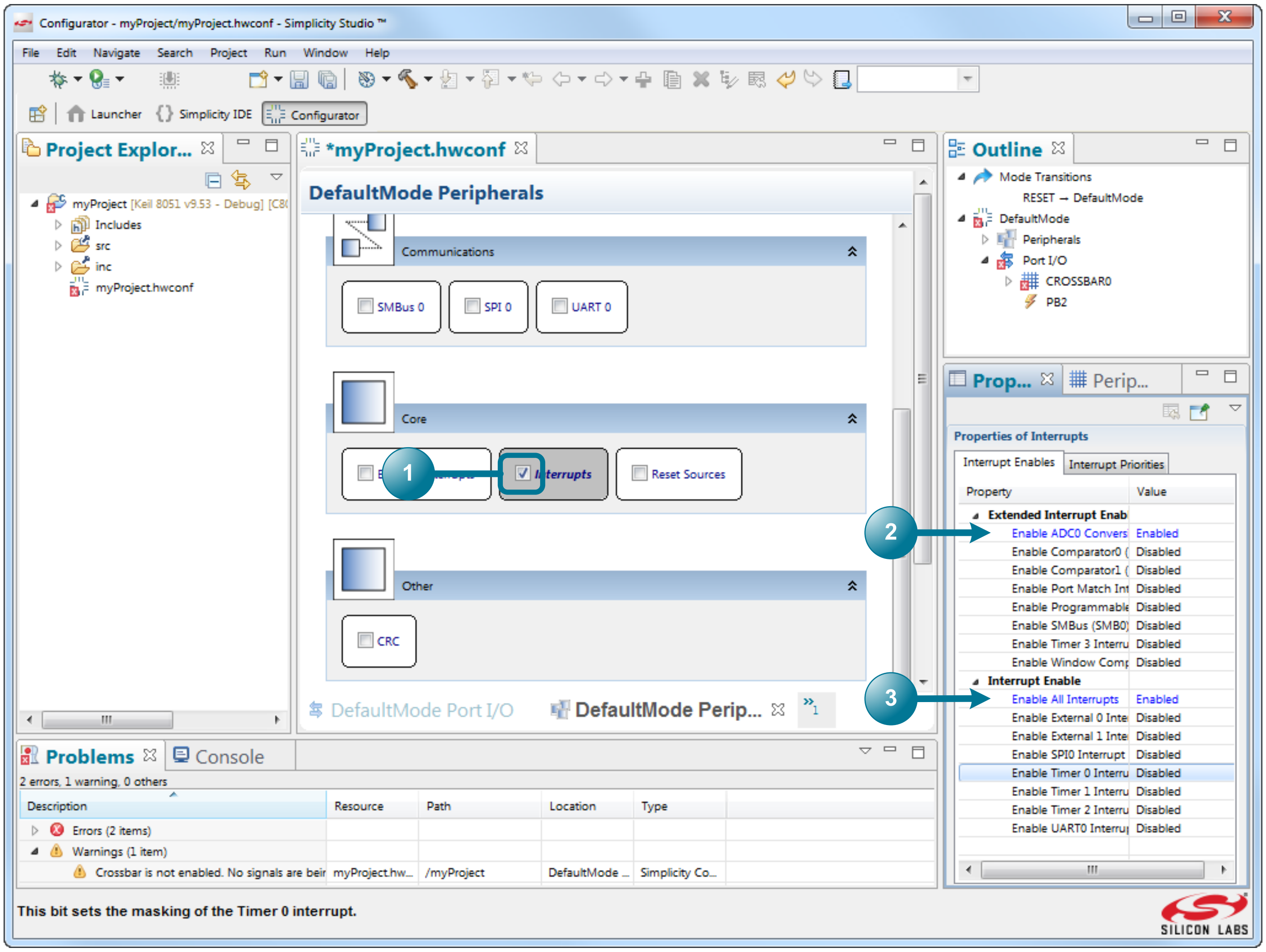
Description	Resource	Path	Location	Type
Errors (2 items)				
Warnings (1 item)				
Crossbar is not enabled. No signals are beir	myProject.hw...	/myProject	DefaultMode ...	Simplicity Co...

Enables/Disables the internal temperature sensor.

3.9 Step 9—Enable ADC Interrupts

Enable ADC 0 conversion complete interrupts and enable the global interrupt enable flag:

1. Select [**Interrupts**] from the [**Core**] group.
2. Set the [**Enable ADC0 Conversion Complete Interrupt**] property to [**Enabled**].
3. Set the [**Enable All Interrupts**] property to [**Enabled**].



Properties of Interrupts

Interrupt Enables | Interrupt Priorities

Property	Value
Extended Interrupt Enab	
Enable ADC0 Convers	Enabled
Enable Comparator0 (	Disabled
Enable Comparator1 (	Disabled
Enable Port Match Int	Disabled
Enable Programmable	Disabled
Enable SMBus (SMB0)	Disabled
Enable Timer 3 Interru	Disabled
Enable Window Comp	Disabled
Interrupt Enable	
Enable All Interrupts	Enabled
Enable External 0 Inte	Disabled
Enable External 1 Inte	Disabled
Enable SPI0 Interrupt	Disabled
Enable Timer 0 Interru	Disabled
Enable Timer 1 Interru	Disabled
Enable Timer 2 Interru	Disabled
Enable UART0 Interru	Disabled

2 errors, 1 warning, 0 others

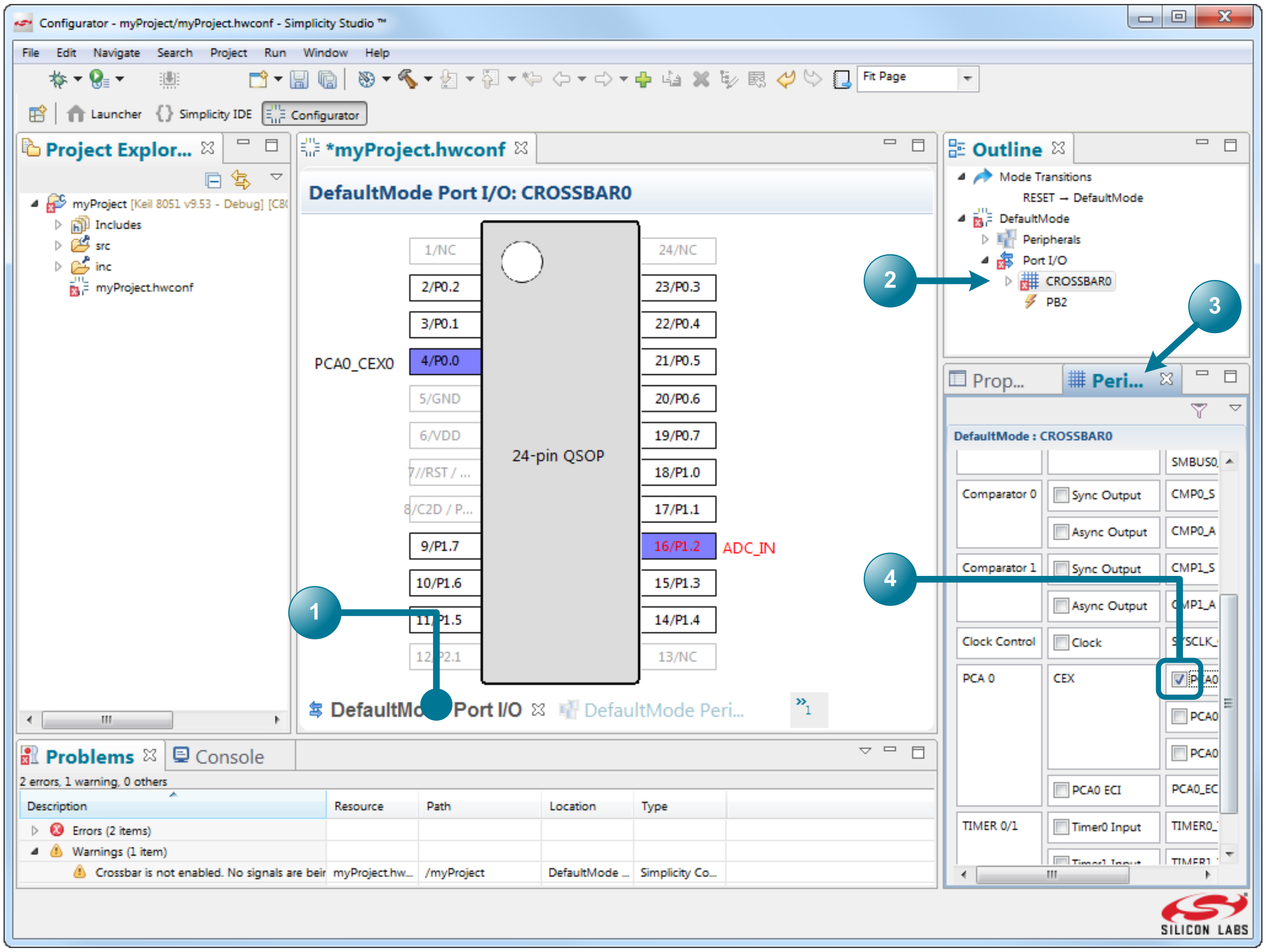
Description	Resource	Path	Location	Type
Errors (2 items)				
Warnings (1 item)				
Crossbar is not enabled. No signals are beir	myProject.hw...	/myProject	DefaultMode ...	Simplicity Co...

This bit sets the masking of the Timer 0 interrupt.

3.10 Step 10—Enable the LED Pin

Enable the PCA0_CEX0 pin on the crossbar:

1. Switch to the [**Port I/O**] tab.
2. Click [**CROSSBAR0**] from the [**Outline**] view.
3. Switch to the [**Peripheral Mapping**] view, if it's not already open.
4. Check only the [**PCA0_CEX0**] check box.



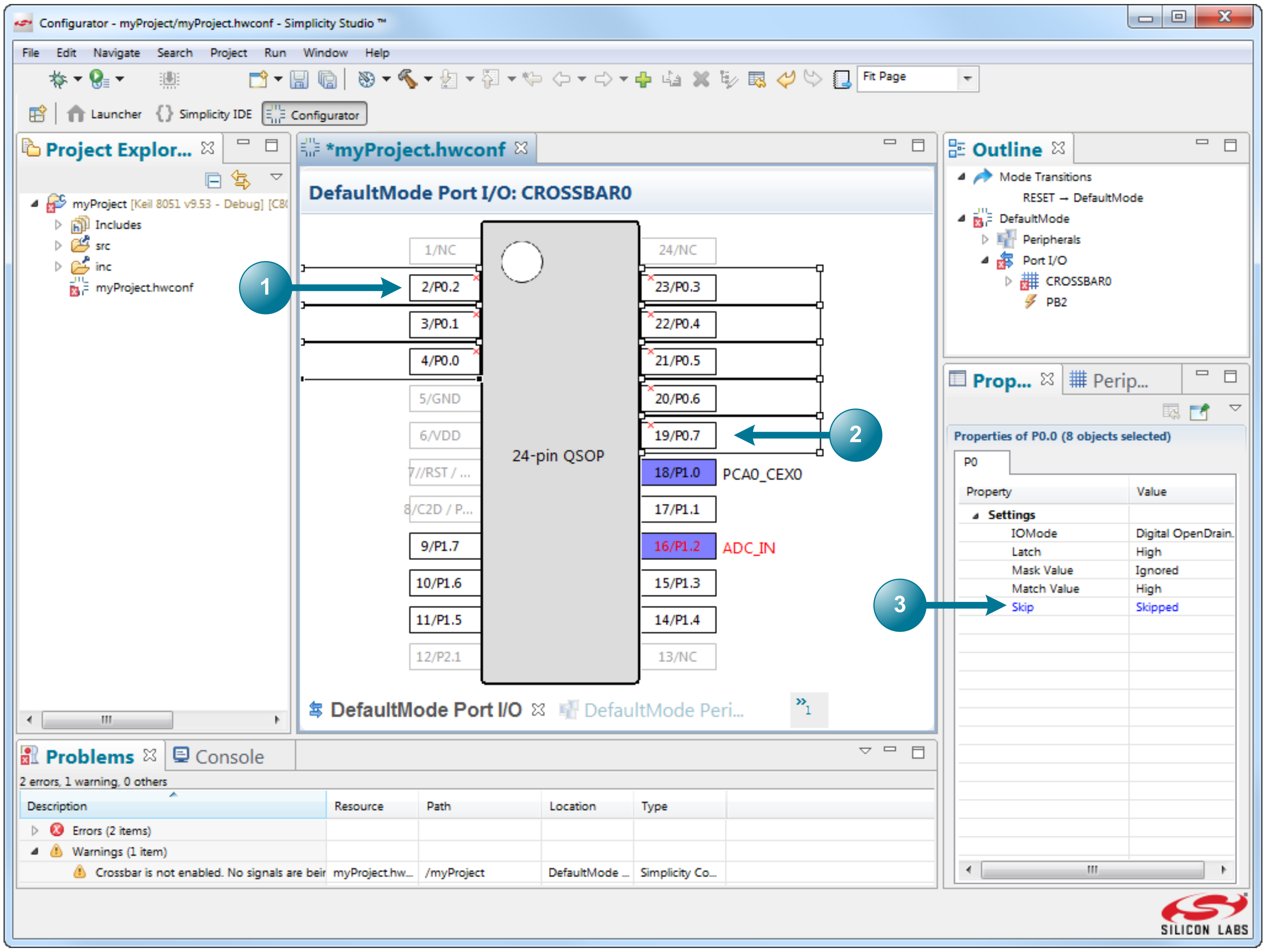
2 errors, 1 warning, 0 others

Description	Resource	Path	Location	Type
Errors (2 items)				
Warnings (1 item)				
Crossbar is not enabled. No signals are beir	myProject.hw...	/myProject	DefaultMode ...	Simplicity Co...

3.11 Step 11—Move the LED Pin

Place PCA0_CEX0 on the P1.0 LED pin by skipping P0.0 through P0.7 on the crossbar:

1. Click [**P0.2**] in the [**Port I/O**] editor.
2. Holding the [**Ctrl**] key, click each pin from [**P0.0**] through [**P0.7**] to select all 8 pins.
3. Set the [**Skip**] property to [**Skipped**] or right-click on the pins and select [**Skip Selected Pins**].



Properties of P0.0 (8 objects selected)

P0

Property	Value
Settings	
IOMode	Digital OpenDrain.
Latch	High
Mask Value	Ignored
Match Value	High
Skip	Skipped

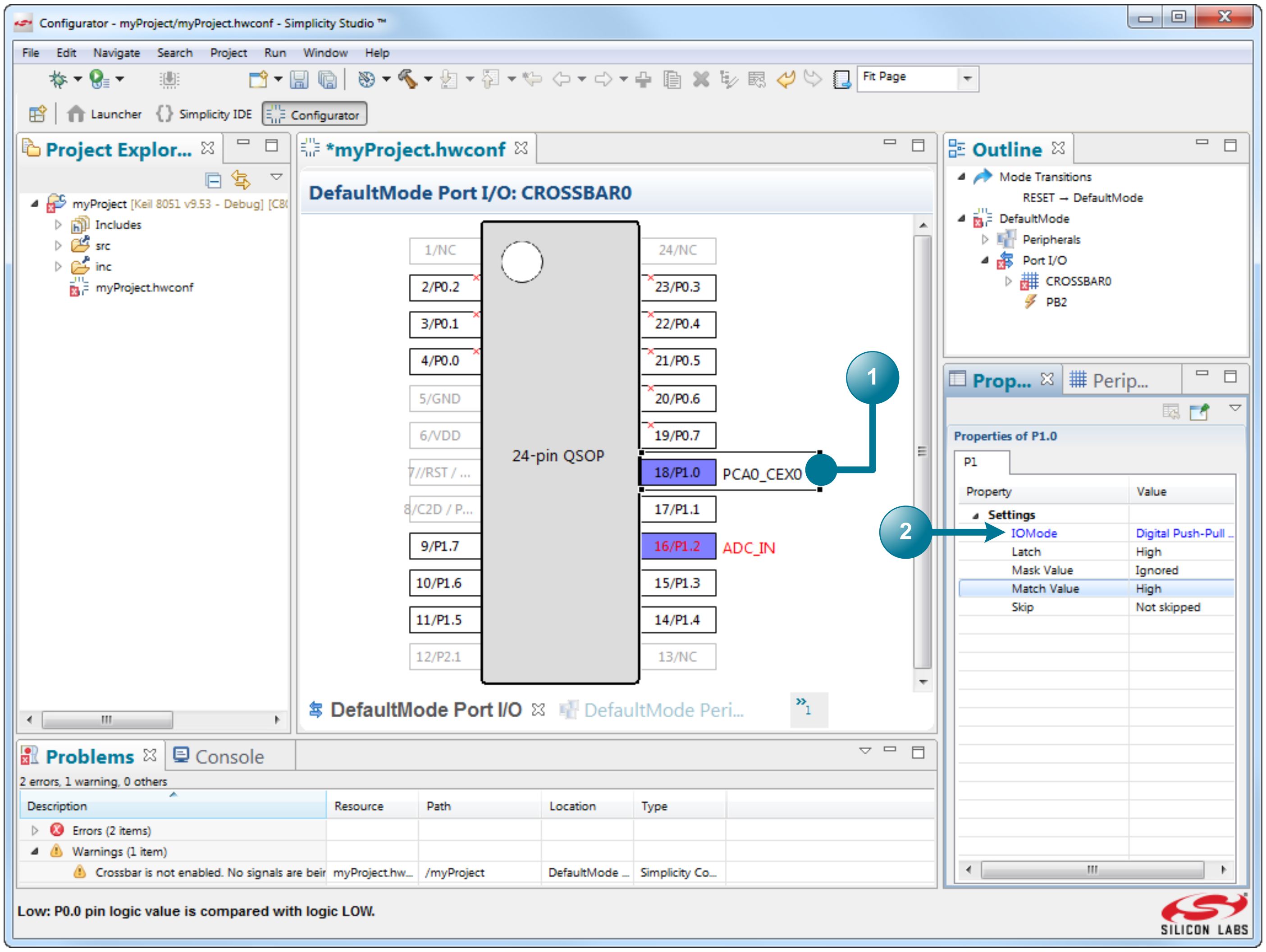
2 errors, 1 warning, 0 others

Description	Resource	Path	Location	Type
Errors (2 items)				
Warnings (1 item)				
Crossbar is not enabled. No signals are beir	myProject.hw...	/myProject	DefaultMode ...	Simplicity Co...

3.12 Step 12—Configure the LED Pin Output

Configure PCA_CEX0 / P1.0 (LED) for push-pull output mode:

1. Select [**PCA0_CEX0**] / P1.0 or pin 18.
2. Set the [**IOMode**] property to [**Digital Push-Pull Output**].



Property	Value
Settings	
IOMode	Digital Push-Pull ...
Latch	High
Mask Value	Ignored
Match Value	High
Skip	Not skipped

2 errors, 1 warning, 0 others

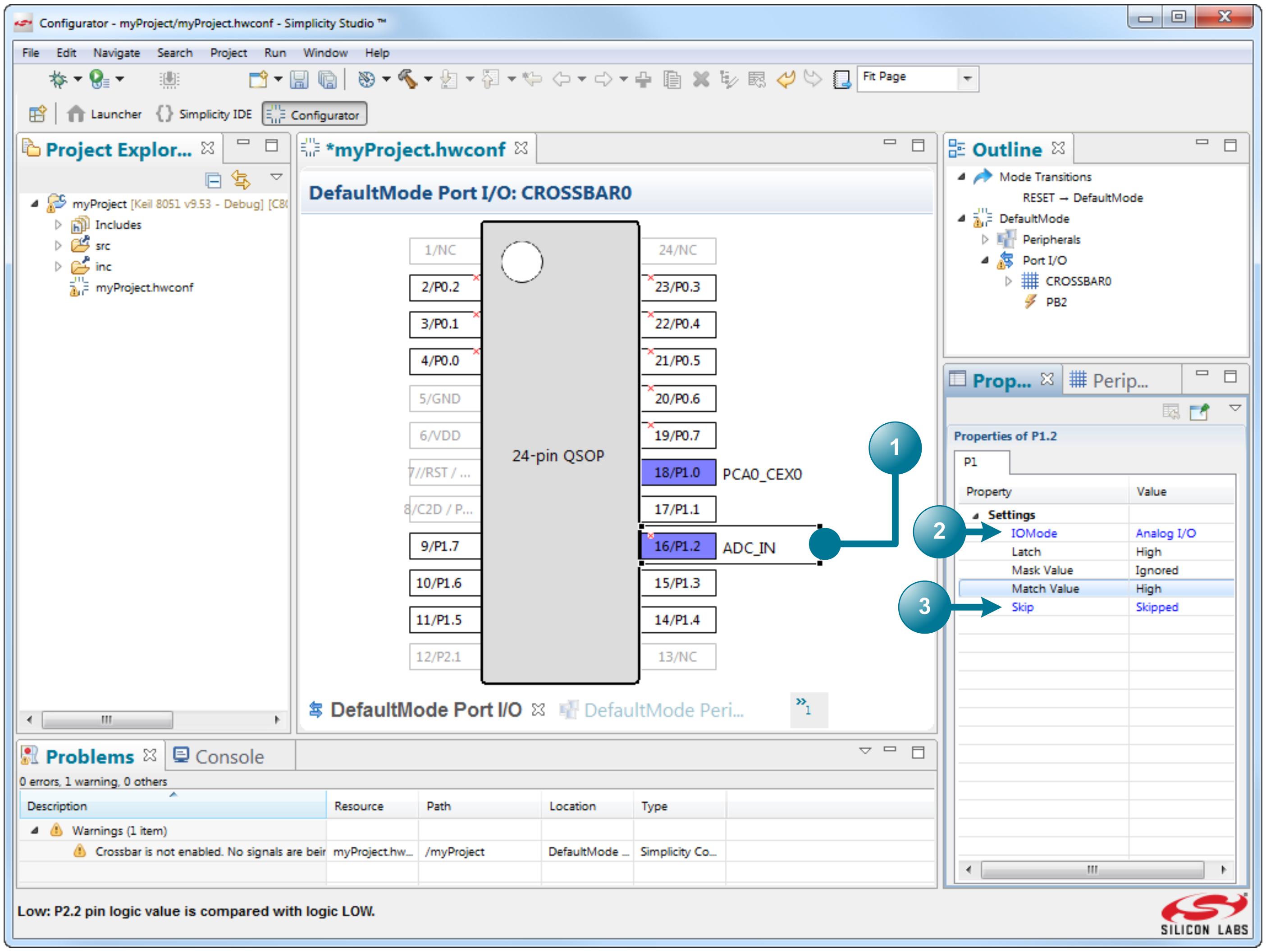
Description	Resource	Path	Location	Type
Errors (2 items)				
Warnings (1 item)				
Crossbar is not enabled. No signals are beir	myProject.hw...	/myProject	DefaultMode ...	Simplicity Co...

Low: P0.0 pin logic value is compared with logic LOW.

3.13 Step 13—Configure the Potentiometer Input

Configure ADC0.10 / P1.2 (Potentiometer) for analog mode and skipped on the crossbar:

1. Select [**ADC_IN**] / P1.2 or pin 16.
2. Set the [**IOMode**] property to [**Analog I/O**].
3. Set the [**Skip**] property to [**Skipped**].



Properties of P1.2

P1

Property	Value
Settings	
IOMode	Analog I/O
Latch	High
Mask Value	Ignored
Match Value	High
Skip	Skipped

0 errors, 1 warning, 0 others

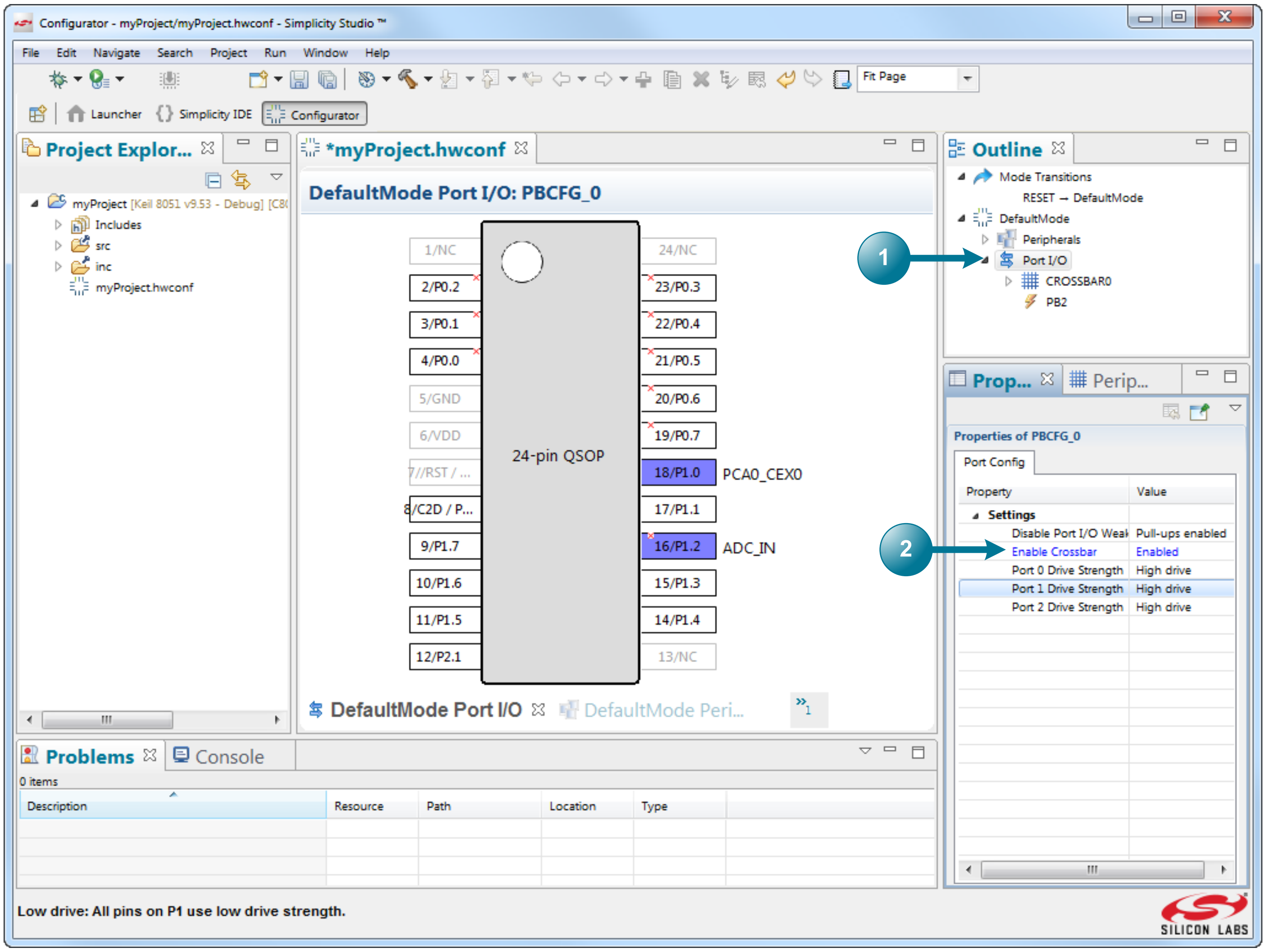
Description	Resource	Path	Location	Type
Warnings (1 item)				
Crossbar is not enabled. No signals are beir	myProject.hw...	/myProject	DefaultMode ...	Simplicity Co...

Low: P2.2 pin logic value is compared with logic LOW.

3.14 Step 14—Enabling the Crossbar

Enable the crossbar to enable the port pin drivers and weak pullup resistors:

1. Select [**Port I/O**] from the [**Outline**] view to display the Port Config properties.
2. Set the [**Enable Crossbar**] property to [**Enabled**].



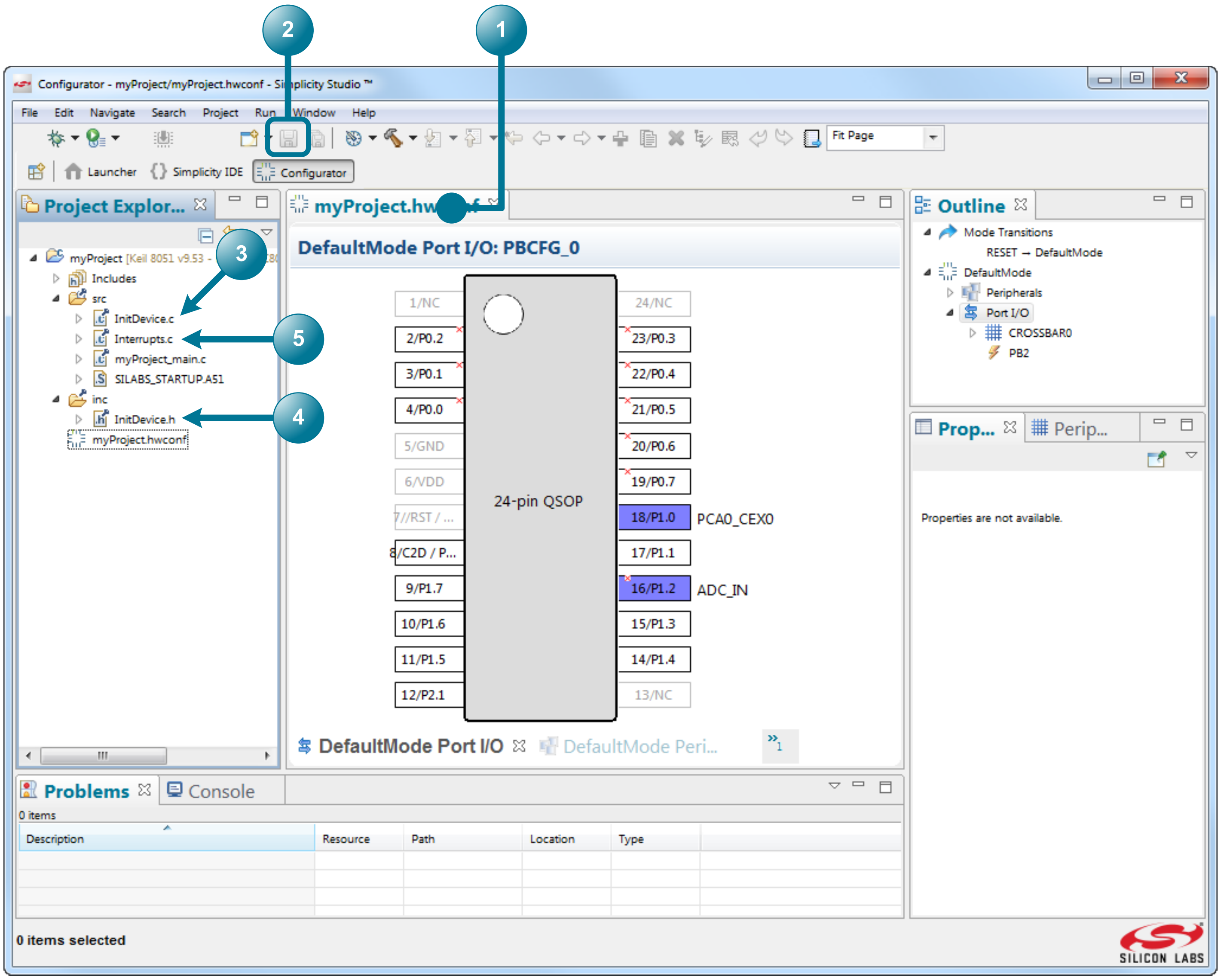
Property	Value
Settings	
Disable Port I/O Weak	Pull-ups enabled
Enable Crossbar	Enabled
Port 0 Drive Strength	High drive
Port 1 Drive Strength	High drive
Port 2 Drive Strength	High drive

Low drive: All pins on P1 use low drive strength.

3.15 Step 15—Save to Generate the Source Code

After configuring peripheral properties, save the Configurator file (`*.hwconf`) to generate `InitDevice.c`, `InitDevice.h`, and `Interrupts.c`:

1. Select the `F85x_PWM_LED.hwconf` editor tab (assuming the initial project was called F85x_PWM_LED).
2. Click the **[Save]** button from the tool bar. The save file operation is context sensitive and will only save the currently active document in the editor.
3. Configurator generates `InitDevice.c`, which contains source code to initialize the peripherals as configured in the GUI.
4. Configurator also generates `InitDevice.h`, which contains function prototypes for initialization routines.
5. Finally, Configurator generates `Interrupts.c`, which contains the interrupt service routines.

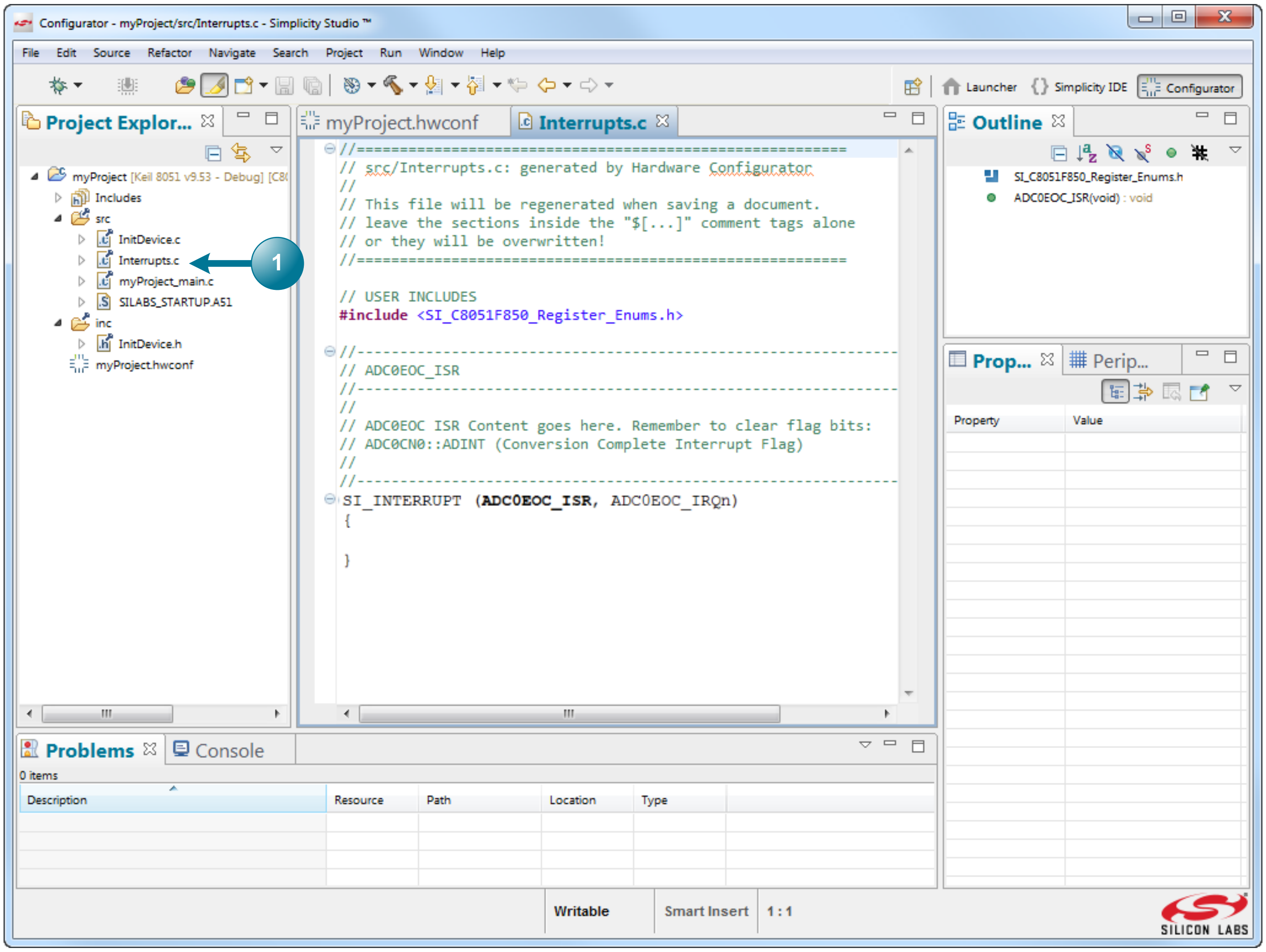


3.16 Step 16—Opening `Interrupts.c`

After generating hardware initialization functions using Simplicity Configurator, edit the source code to copy the ADC reading into the PCA PWM output. The Eclipse framework provides a very powerful editor with rich features, such as C++ syntax error highlighting, code completion, and source browse information to quickly locate symbol declarations.

Edit the project source code to change the PWM duty cycle based on the ADC sample to change the LED brightness:

1. Double-click on the `Interrupts.c` file in the Project Explorer.

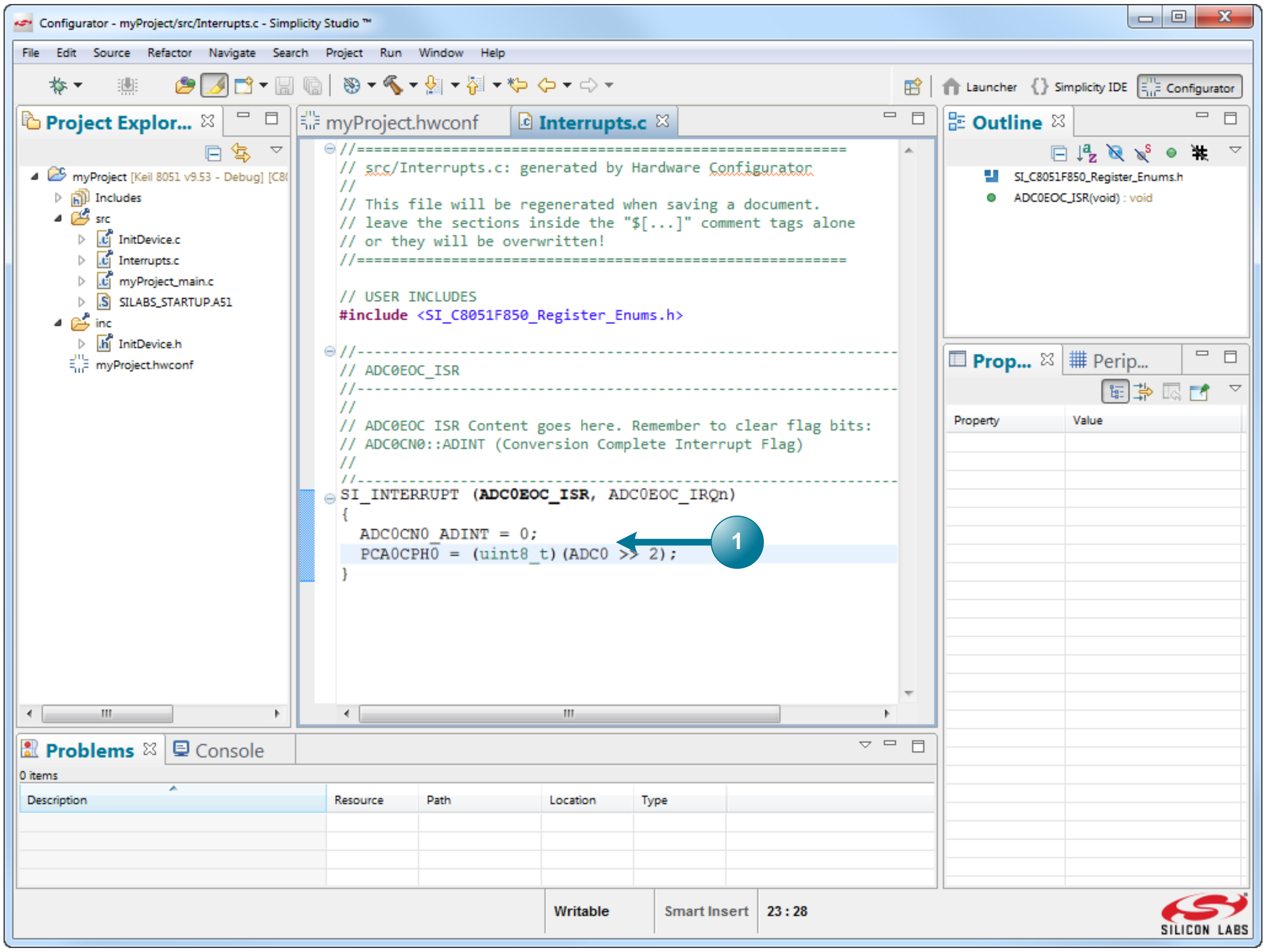


```
//=========================================================
// src/Interrupts.c: generated by Hardware Configurator
//
// This file will be regenerated when saving a document.
// leave the sections inside the "$[...]" comment tags alone
// or they will be overwritten!
//=========================================================

// USER INCLUDES
#include <SI_C8051F850_Register_Enums.h>

//-----------------------------------------------------------------
// ADC0EOC_ISR
//-----------------------------------------------------------------
//
// ADC0EOC ISR Content goes here. Remember to clear flag bits:
// ADC0CN0::ADINT (Conversion Complete Interrupt Flag)
//
//-----------------------------------------------------------------
SI_INTERRUPT (ADC0EOC_ISR, ADC0EOC_IRQn)
{

}
```

3.17 Step 17—Updating `Interrupts.c`

Enter the following code into the ADC ISR and save the document:

```
ADC0CN0_ADINT = 0;
PCA0CPH0 = (uint8_t)(ADC0 >> 2);
```

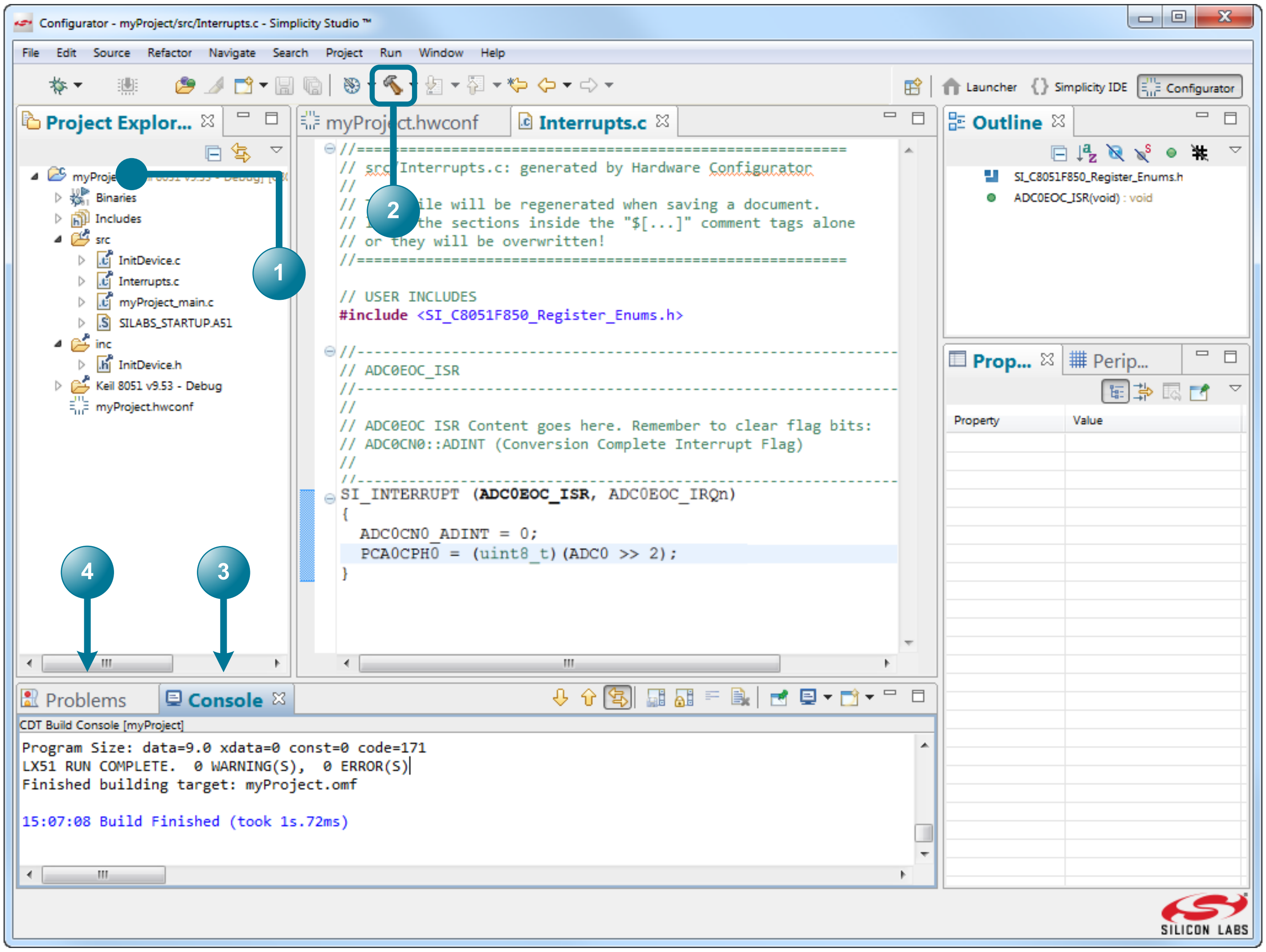


```
//=========================================================
// src/Interrupts.c: generated by Hardware Configurator
//
// This file will be regenerated when saving a document.
// leave the sections inside the "$[...]" comment tags alone
// or they will be overwritten!
//=========================================================

// USER INCLUDES
#include <SI_C8051F850_Register_Enums.h>

//-----------------------------------------------------------------
// ADC0EOC_ISR
//-----------------------------------------------------------------
//
// ADC0EOC ISR Content goes here. Remember to clear flag bits:
// ADC0CN0::ADINT (Conversion Complete Interrupt Flag)
//
//-----------------------------------------------------------------
SI_INTERRUPT (ADC0EOC_ISR, ADC0EOC_IRQn)
{
  ADC0CN0_ADINT = 0;
  PCA0CPH0 = (uint8_t)(ADC0 >> 2);
}
```

3.18 Step 18—Building the Project

Build the project and make sure that there are no errors or warnings:

1. Select the project in the [**Project Explorer**].
2. Select [**Project**]>[**Build Project**] from the menu bar or use the [**Build**] button on the tool bar. The Project menu is context sensitive, which means that an active project must be selected before building the project.
3. The compiler and linker build output will be displayed in the [**Console**] view.
4. Check the [**Problems**] tab to see if there are any errors or warnings.
5. Correct any problems before continuing.



```
//=========================================================
// src/Interrupts.c: generated by Hardware Configurator
//
// This file will be regenerated when saving a document.
// leave the sections inside the "$[...]" comment tags alone
// or they will be overwritten!
//=========================================================

// USER INCLUDES
#include <SI_C8051F850_Register_Enums.h>

//-----------------------------------------------------------
// ADC0EOC_ISR
//-----------------------------------------------------------
//
// ADC0EOC ISR Content goes here. Remember to clear flag bits:
// ADC0CN0::ADINT (Conversion Complete Interrupt Flag)
//
//-----------------------------------------------------------
SI_INTERRUPT (ADC0EOC_ISR, ADC0EOC_IRQn)
{
  ADC0CN0_ADINT = 0;
  PCA0CPH0 = (uint8_t)(ADC0 >> 2);
}
```

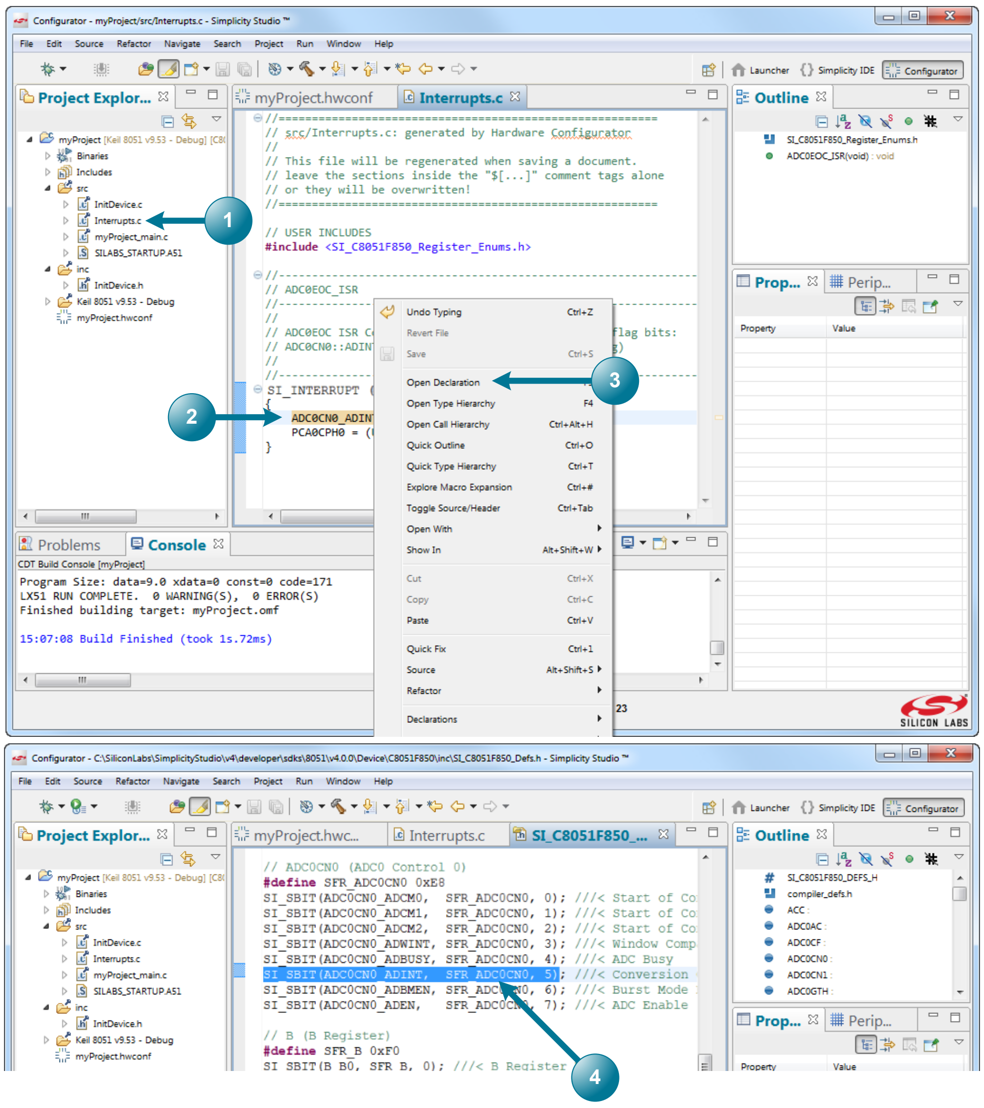
```
Program Size: data=9.0 xdata=0 const=0 code=171
LX51 RUN COMPLETE.  0 WARNING(S),  0 ERROR(S)
Finished building target: myProject.omf

15:07:08 Build Finished (took 1s.72ms)
```

3.19 (Optional) Step 19—Using Open Declaration

Simplicity IDE supports many modern code editing features. For example, the IDE will automatically index all code within the project to support symbol lookup. The code does not have to build completely for the indexer to work. However certain features may not be available if for example, the main() routine is not declared.

Next, use the [**Open Declaration**] feature to quickly find symbol declarations:

1. If not already open, open the `Interrupts.c` document by double clicking `Interrupts.c` in the [**Project Explorer**].
2. Right click on [**ADC0CN0_ADINT**] to display the context menu.
3. Click [**Open Declaration**] to quickly navigate to the definition of ADC0CN0_ADINT.
4. Studio will automatically open `SI_C8051F850_Defs.h` and highlight the line containing the declaration of ADC0CN0_ADINT.



```
//============================================================
// src/Interrupts.c: generated by Hardware Configurator
//
// This file will be regenerated when saving a document.
// leave the sections inside the "$[...]" comment tags alone
// or they will be overwritten!
//============================================================

// USER INCLUDES
#include <SI_C8051F850_Register_Enums.h>

//-----------------------------------------------------------------
// ADC0EOC_ISR
//-----------------------------------------------------------------
//
// ADC0EOC ISR C                                  flag bits:
// ADC0CN0::ADINT                                 g)
//
//-----------------------------------------------------------------
SI_INTERRUPT (
{
    ADC0CN0_ADINT
    PCA0CPH0 = (
}
```

```
Program Size: data=9.0 xdata=0 const=0 code=171
LX51 RUN COMPLETE.  0 WARNING(S),  0 ERROR(S)
Finished building target: myProject.omf

15:07:08 Build Finished (took 1s.72ms)
```

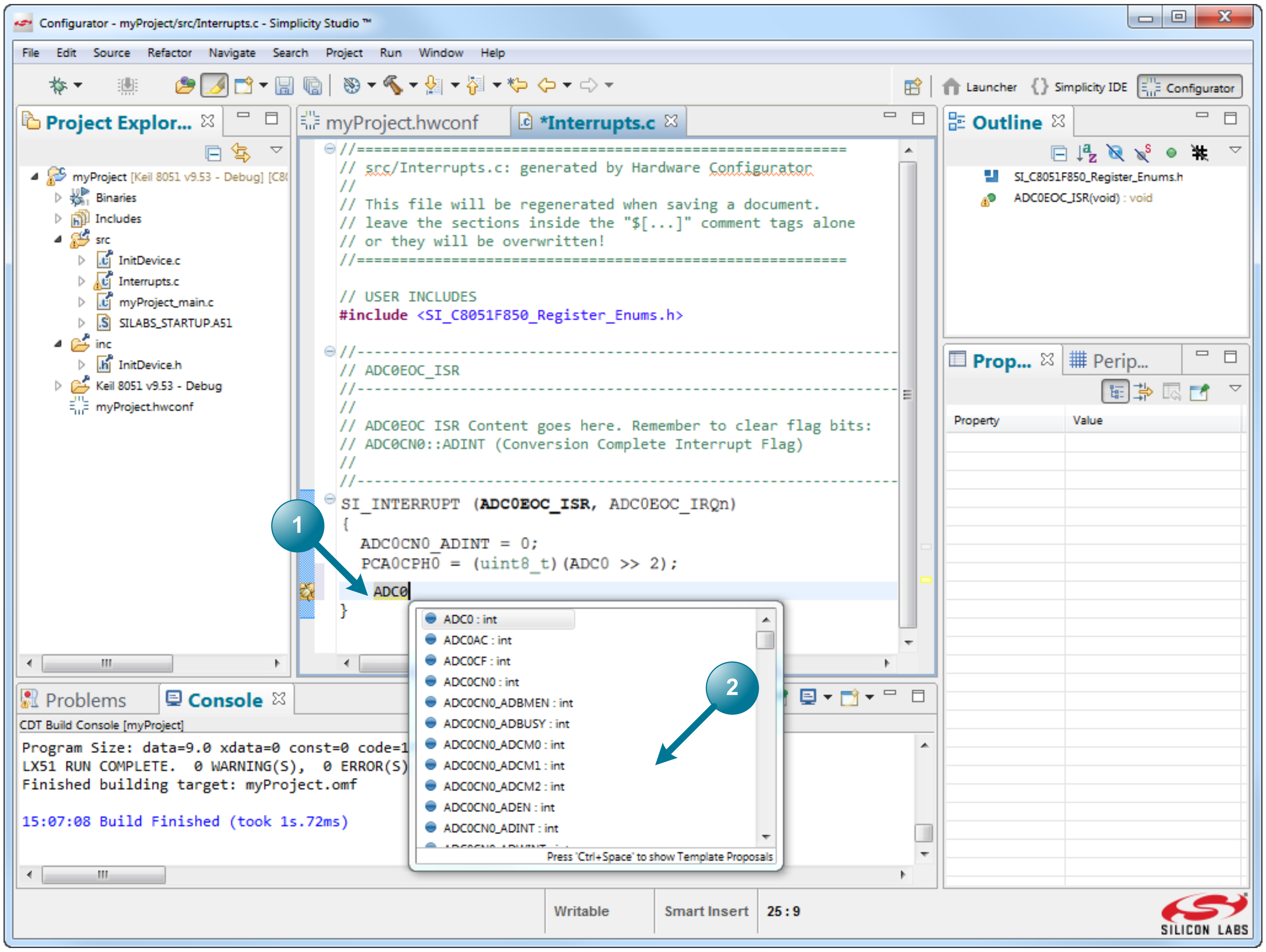
```
// ADC0CN0 (ADC0 Control 0)
#define SFR_ADC0CN0 0xE8
SI_SBIT(ADC0CN0_ADCM0,  SFR_ADC0CN0, 0); ///< Start of Co
SI_SBIT(ADC0CN0_ADCM1,  SFR_ADC0CN0, 1); ///< Start of Co
SI_SBIT(ADC0CN0_ADCM2,  SFR_ADC0CN0, 2); ///< Start of Co
SI_SBIT(ADC0CN0_ADWINT, SFR_ADC0CN0, 3); ///< Window Comp
SI_SBIT(ADC0CN0_ADBUSY, SFR_ADC0CN0, 4); ///< ADC Busy
SI_SBIT(ADC0CN0_ADINT,  SFR_ADC0CN0, 5); ///< Conversion
SI_SBIT(ADC0CN0_ADBMEN, SFR_ADC0CN0, 6); ///< Burst Mode
SI_SBIT(ADC0CN0_ADEN,   SFR_ADC0CN0, 7); ///< ADC Enable

// B (B Register)
#define SFR_B 0xF0
SI_SBIT(B_B0, SFR_B, 0); ///< B Register
```

3.20 (Optional) Step 20—Using Content Assist

Simplicity IDE also supports code completion, a feature called Content Assist. The user can type the first few letters of a symbol or include file and press [**Ctrl+Space**] to display a list of symbols that match the letters types.

Next, use Content Assist to display a list of symbols starting with the characters P1:

1. In `Interrupts.c`, type [**ADC0**] at the end of the ADC0_ISR function.
2. Press [**Ctrl+Space**] to display the Content Assist list.
3. Use the arrow keys or page up and down keys to look through the list of matching symbols.
4. Pressing [**Enter**] will replace the typed characters with the selected symbol.
5. Once familiar with the Content Assist feature, delete any lines inserted in this step.



```
//==========================================================
// src/Interrupts.c: generated by Hardware Configurator
//
// This file will be regenerated when saving a document.
// leave the sections inside the "$[...]" comment tags alone
// or they will be overwritten!
//==========================================================

// USER INCLUDES
#include <SI_C8051F850_Register_Enums.h>

//-----------------------------------------------------------------
// ADC0EOC_ISR
//-----------------------------------------------------------------
//
// ADC0EOC ISR Content goes here. Remember to clear flag bits:
// ADC0CN0::ADINT (Conversion Complete Interrupt Flag)
//
//-----------------------------------------------------------------
SI_INTERRUPT (ADC0EOC_ISR, ADC0EOC_IRQn)
{
  ADC0CN0_ADINT = 0;
  PCA0CPH0 = (uint8_t)(ADC0 >> 2);
   ADC0
}
```

```
ADC0 : int
ADC0AC : int
ADC0CF : int
ADC0CN0 : int
ADC0CN0_ADBMEN : int
ADC0CN0_ADBUSY : int
ADC0CN0_ADCM0 : int
ADC0CN0_ADCM1 : int
ADC0CN0_ADCM2 : int
ADC0CN0_ADEN : int
ADC0CN0_ADINT : int
Press 'Ctrl+Space' to show Template Proposals
```

CDT Build Console [myProject]

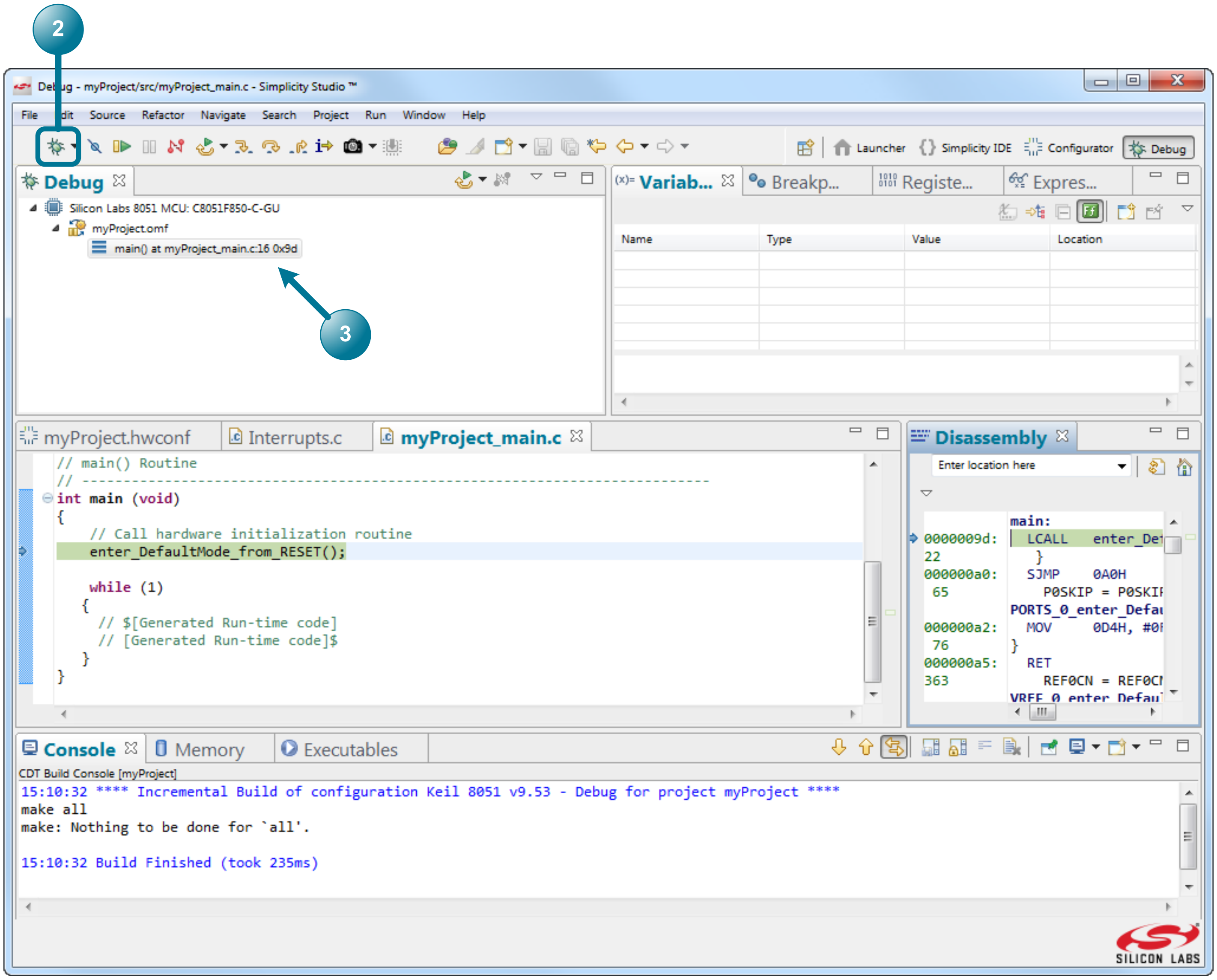
```
Program Size: data=9.0 xdata=0 const=0 code=1
LX51 RUN COMPLETE.  0 WARNING(S),  0 ERROR(S)
Finished building target: myProject.omf

15:07:08 Build Finished (took 1s.72ms)
```

3.21 Step 21—Start a Debug Session

Once an MCU project builds, start a debug session. If a single debug adapter is available, the IDE will automatically download the code to the MCU. If more than one debug adapter is available, the IDE will prompt the user to select a debugger. The Silicon Labs products support non-intrusive, on-chip debug that provides hardware breakpoints, single stepping, and register/memory view.

Start a new debug session:

1. Select the project name from the [**Project Explorer**].
2. Click the [**Debug**] button or go to [**Run**]>[**Debug**]. The IDE will automatically switch to the Debug perspective. This perspective can be used to set breakpoints, run/halt the CPU, single step, view/modify register values, and display the disassembly viewer.
3. The Debug view in the top-left corner displays any active debug sessions. A debug adapter can only support a single debug session at a time. An active debug session must be disconnected before code can be recompiled and a new debug session started.



```
Silicon Labs 8051 MCU: C8051F850-C-GU
  myProject.omf
    main() at myProject_main.c:16 0x9d
```

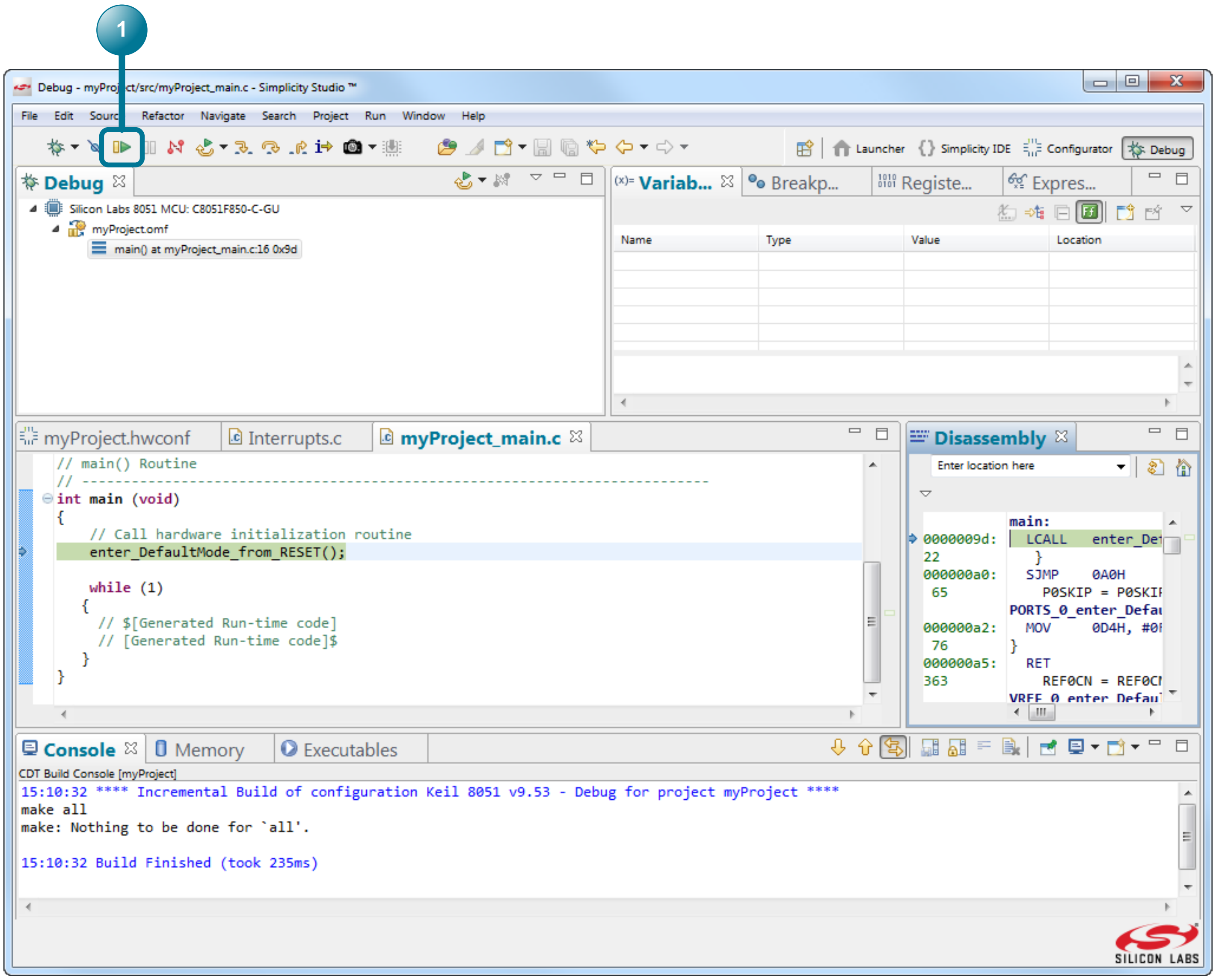
```
// main() Routine
// -----------------------------------------------------------------------------
int main (void)
{
    // Call hardware initialization routine
    enter_DefaultMode_from_RESET();

    while (1)
   {
     // $[Generated Run-time code]
     // [Generated Run-time code]$
   }
}
```

```
                main:
0000009d:       LCALL    enter_Def
22                }
000000a0:       SJMP     0A0H
 65                P0SKIP = P0SKIF
                PORTS_0_enter_Defau
000000a2:       MOV      0D4H, #0F
 76             }
000000a5:       RET
363                REF0CN = REF0CN
                VREF_0_enter_Defaul
```

```
15:10:32 **** Incremental Build of configuration Keil 8051 v9.53 - Debug for project myProject ****
make all
make: Nothing to be done for `all'.

15:10:32 Build Finished (took 235ms)
```

3.22 Step 22—Run the Code

Press the Resume button in the IDE. This will start the code running, and rotating the potentiometer will brighten and dim the P1.0 LED.



```
// main() Routine
// ---------------------------------------------------------------------------
int main (void)
{
    // Call hardware initialization routine
    enter_DefaultMode_from_RESET();

    while (1)
    {
      // $[Generated Run-time code]
      // [Generated Run-time code]$
    }
}
```

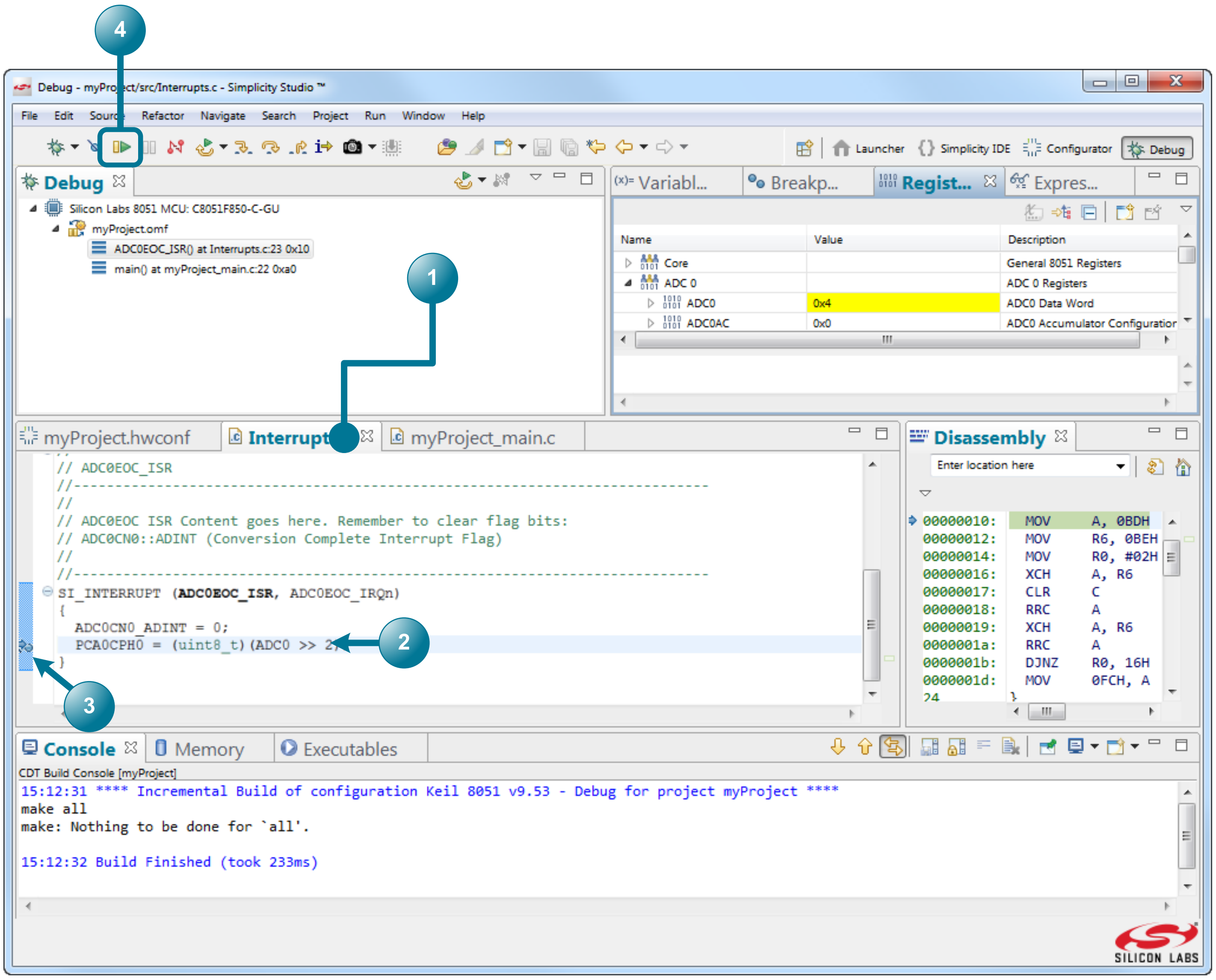
```
            main:
0000009d:   LCALL    enter_De
22          }
000000a0:   SJMP     0A0H
65            P0SKIP = P0SKI
            PORTS_0_enter_Defa
000000a2:   MOV      0D4H, #0
76          }
000000a5:   RET
363           REF0CN = REF0C
            VREF_0_enter_Defau
```

```
15:10:32 **** Incremental Build of configuration Keil 8051 v9.53 - Debug for project myProject ****
make all
make: Nothing to be done for `all'.

15:10:32 Build Finished (took 235ms)
```

3.23 (Optional) Step 23—Using the Debugging Features

Use a breakpoint to halt the CPU just prior to executing the statement to read the ADC sample and update the PWM duty cycle:

1. Open `Interrupts.c` in the editor. If `Interrupts.c` is not shown in the tabbed list of files, then quickly switch to the Development perspective and use the Project Explorer to open the Interrupts.c file. Next, switch back to the Debug perspective. The debug session is still active when switching perspectives. Files shown in the editor will be visible in both the Development and Debug perspectives.
2. Scroll down to the following line in `Interrupts.c`: `PCA0CPH0 = (uint8_t)(ADC0 >> 2);`.
3. Move the mouse cursor to the left-hand margin of the editor in the highlighted blue region. Double-click the margin to add or remove a breakpoint at the corresponding source code line.
4. The core will halt on the breakpoint. Clicking the [**Resume**] button will run the CPU until the ADC conversion complete interrupt fires again. Then the CPU will halt and all of the debug windows will update to reflect the new CPU state.


4
1
2
3
Debug - myProject/src/Interrupts.c - Simplicity Studio ™
File Edit Source Refactor Navigate Search Project Run Window Help
Launcher
Simplicity IDE
Configurator
Debug
Debug
Silicon Labs 8051 MCU: C8051F850-C-GU
myProject.omf
ADC0EOC_ISR() at Interrupts.c:23 0x10
main() at myProject_main.c:22 0xa0
Variabl...
Breakp...
Regist...
Expres...
myProject.hwconf
Interrupt
myProject_main.c
Disassembly
Enter location here
Console
Memory
Executables
CDT Build Console [myProject]
SILICON LABS

Name	Value	Description
Core		General 8051 Registers
ADC 0		ADC 0 Registers
ADC0	0x4	ADC0 Data Word
ADC0AC	0x0	ADC0 Accumulator Configuratior

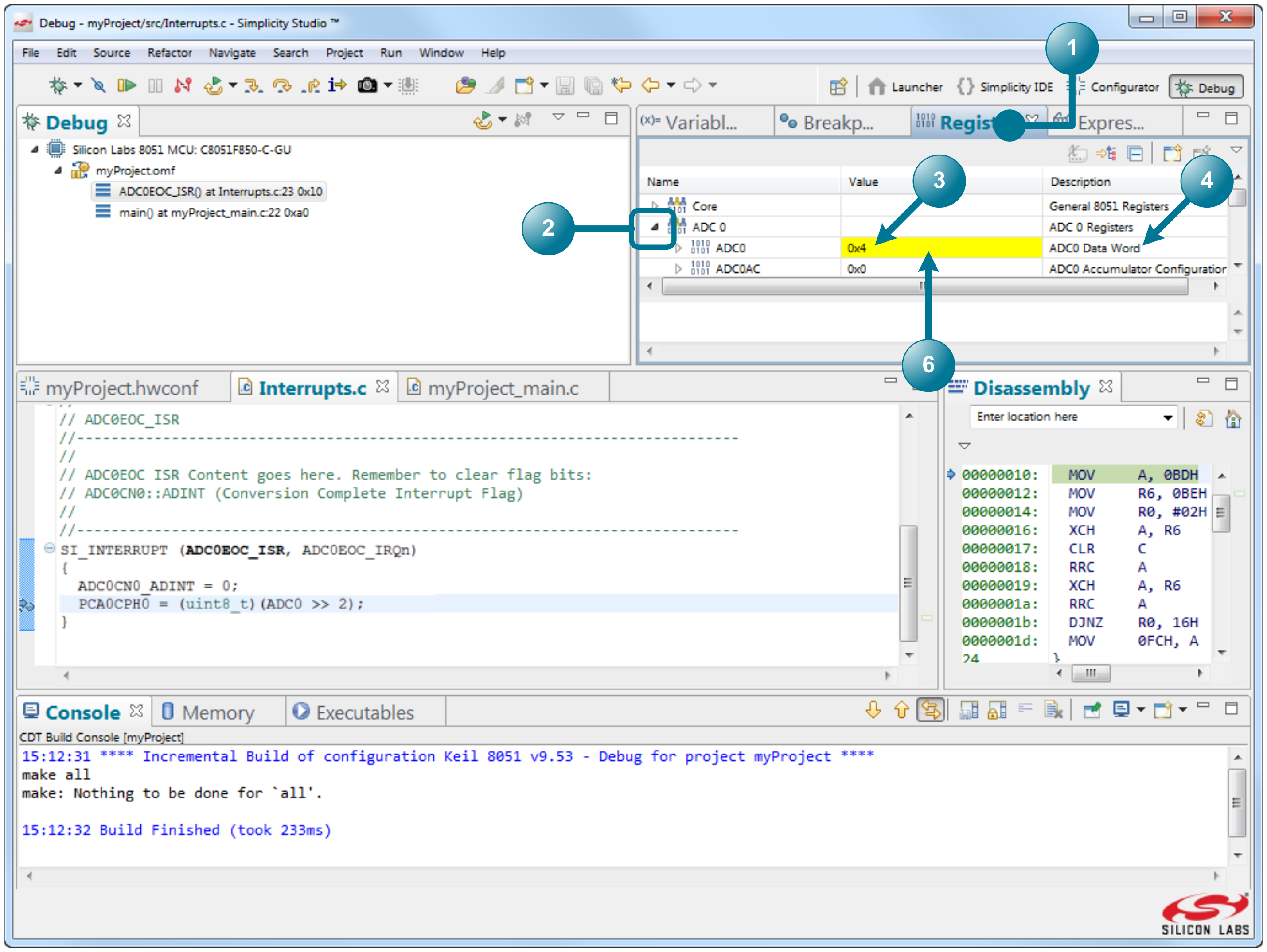
```
// ADC0EOC_ISR
//-----------------------------------------------------------------------------
//
// ADC0EOC ISR Content goes here. Remember to clear flag bits:
// ADC0CN0::ADINT (Conversion Complete Interrupt Flag)
//
//-----------------------------------------------------------------------------
SI_INTERRUPT (ADC0EOC_ISR, ADC0EOC_IRQn)
{
  ADC0CN0_ADINT = 0;
  PCA0CPH0 = (uint8_t)(ADC0 >> 2);
}
```

```
00000010:   MOV    A, 0BDH
00000012:   MOV    R6, 0BEH
00000014:   MOV    R0, #02H
00000016:   XCH    A, R6
00000017:   CLR    C
00000018:   RRC    A
00000019:   XCH    A, R6
0000001a:   RRC    A
0000001b:   DJNZ   R0, 16H
0000001d:   MOV    0FCH, A
24          }
```

```
15:12:31 **** Incremental Build of configuration Keil 8051 v9.53 - Debug for project myProject ****
make all
make: Nothing to be done for `all'.

15:12:32 Build Finished (took 233ms)
```

Use the [**Registers**] view to view/modify register values:

1. Select the [**Registers**] tab.
2. Scroll down the register list and expand the [**ADC**] registers.
3. The ADC0 sample value is displayed in the [**ADC0**] row under the [**Value**] column.
4. The register description is displayed in the [**Description**] column.
5. Registers with bitfields will be split into individual fields with the enumeration value decoded.
6. Registers that have changed between breakpoints will be highlighted in yellow.


Debug - myProject/src/Interrupts.c - Simplicity Studio ™
File Edit Source Refactor Navigate Search Project Run Window Help
Launcher
Simplicity IDE
Configurator
Debug
Variabl...
Breakp...
Regist...
Expres...
Silicon Labs 8051 MCU: C8051F850-C-GU
myProject.omf
ADC0EOC_ISR() at Interrupts.c:23 0x10
main() at myProject_main.c:22 0xa0
1
2
3
4
6
myProject.hwconf
Interrupts.c
myProject_main.c
Disassembly
Enter location here
Console
Memory
Executables
CDT Build Console [myProject]
SILICON LABS

Name	Value	Description
Core		General 8051 Registers
ADC 0		ADC 0 Registers
ADC0	0x4	ADC0 Data Word
ADC0AC	0x0	ADC0 Accumulator Configuration

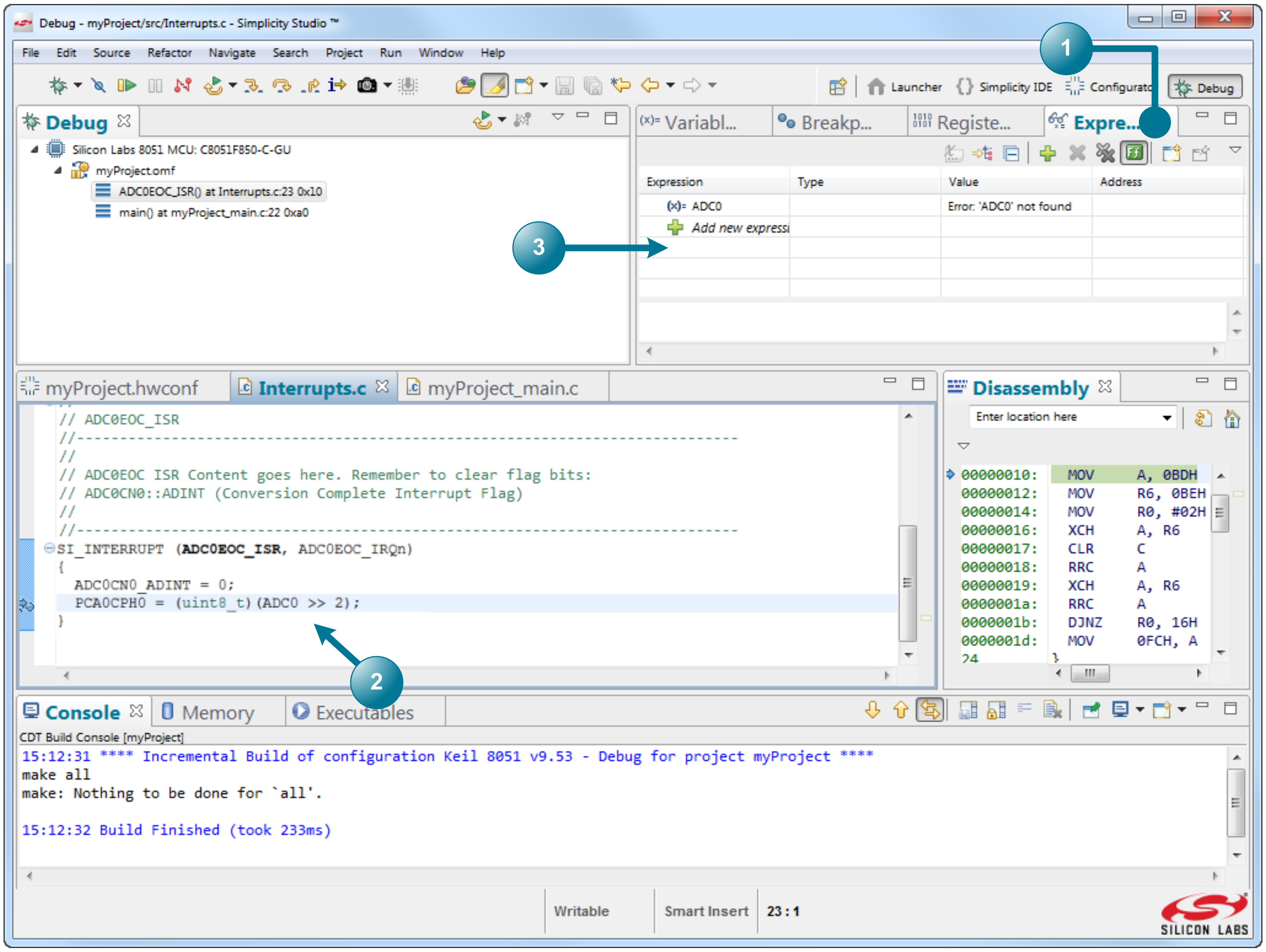
```
// ADC0EOC_ISR
//-----------------------------------------------------------------------------
//
// ADC0EOC ISR Content goes here. Remember to clear flag bits:
// ADC0CN0::ADINT (Conversion Complete Interrupt Flag)
//
//-----------------------------------------------------------------------------
SI_INTERRUPT (ADC0EOC_ISR, ADC0EOC_IRQn)
{
  ADC0CN0_ADINT = 0;
  PCA0CPH0 = (uint8_t)(ADC0 >> 2);
}
```

```
00000010:   MOV    A, 0BDH
00000012:   MOV    R6, 0BEH
00000014:   MOV    R0, #02H
00000016:   XCH    A, R6
00000017:   CLR    C
00000018:   RRC    A
00000019:   XCH    A, R6
0000001a:   RRC    A
0000001b:   DJNZ   R0, 16H
0000001d:   MOV    0FCH, A
24          }
```

```
15:12:31 **** Incremental Build of configuration Keil 8051 v9.53 - Debug for project myProject ****
make all
make: Nothing to be done for `all'.

15:12:32 Build Finished (took 233ms)
```

Use the Expressions view to display the current value of code expressions:

1. Select the [**Expressions**] tab.
2. Highlight the expression, [**ADC0**], in `Interrupts.c`.
3. Drag the highlighted expression to the line just below the [**Add new expression**] button in the [**Expressions**] view.
4. If possible, Simplicity Studio will evaluate the expression and display its current value when the MCU is halted. If an expression is out of scope, Simplicity Studio will not be able to evaluate the expression until the MCU is halted in a section of code where the expression is in scope.



Silicon Labs 8051 MCU: C8051F850-C-GU
myProject.omf
ADC0EOC_ISR() at Interrupts.c:23 0x10
main() at myProject_main.c:22 0xa0

Expression	Type	Value	Address
ADC0		Error: 'ADC0' not found	
Add new expressi			

```
// ADC0EOC_ISR
//-----------------------------------------------------------------------------
//
// ADC0EOC ISR Content goes here. Remember to clear flag bits:
// ADC0CN0::ADINT (Conversion Complete Interrupt Flag)
//
//-----------------------------------------------------------------------------
SI_INTERRUPT (ADC0EOC_ISR, ADC0EOC_IRQn)
{
  ADC0CN0_ADINT = 0;
  PCA0CPH0 = (uint8_t)(ADC0 >> 2);
}
```

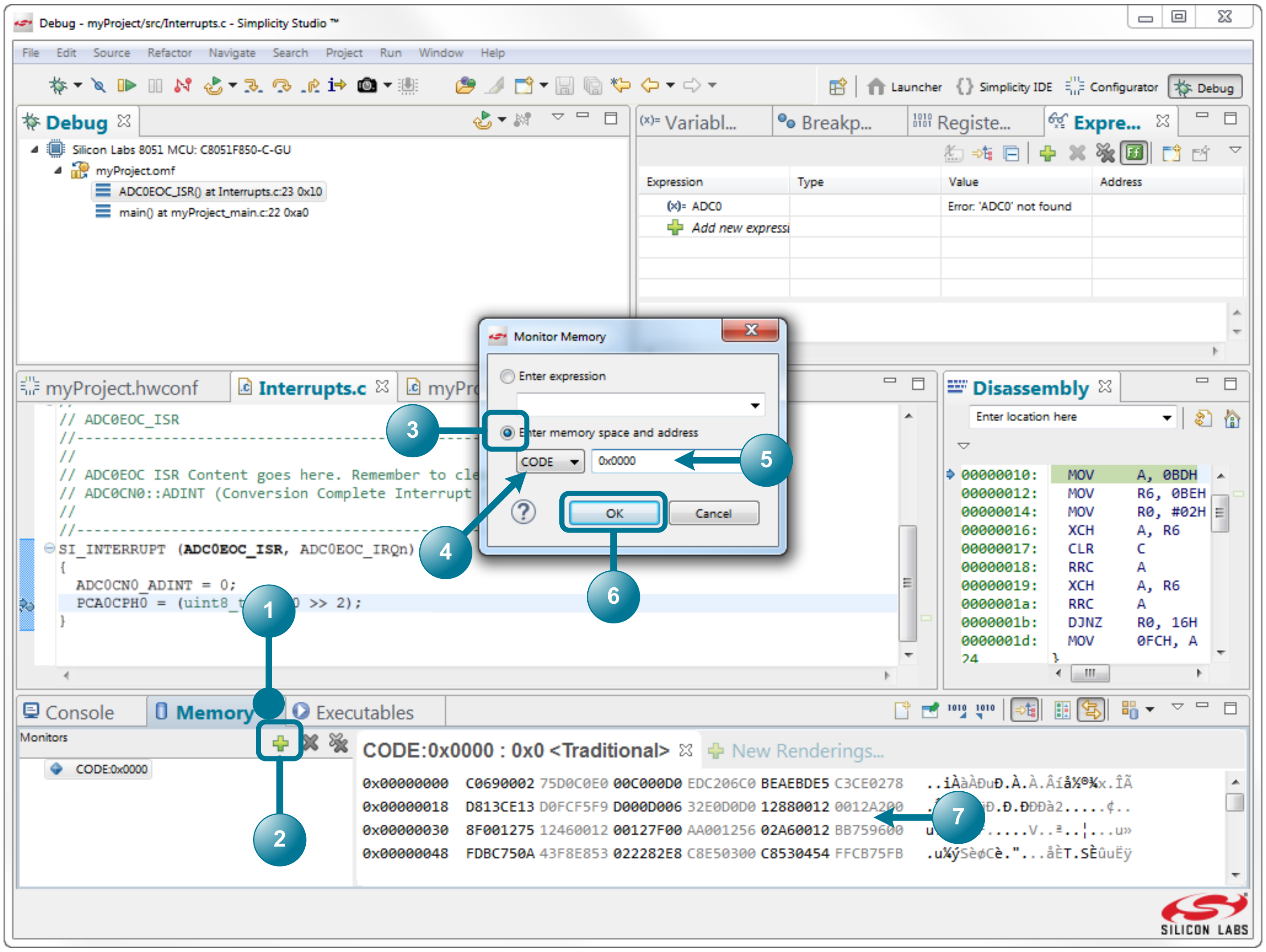
```
00000010:   MOV    A, 0BDH
00000012:   MOV    R6, 0BEH
00000014:   MOV    R0, #02H
00000016:   XCH    A, R6
00000017:   CLR    C
00000018:   RRC    A
00000019:   XCH    A, R6
0000001a:   RRC    A
0000001b:   DJNZ   R0, 16H
0000001d:   MOV    0FCH, A
```

CDT Build Console [myProject]

```
15:12:31 **** Incremental Build of configuration Keil 8051 v9.53 - Debug for project myProject ****
make all
make: Nothing to be done for `all'.

15:12:32 Build Finished (took 233ms)
```

Use the Memory view to display the contents of the CODE, RAM, and XRAM memory:

1. Select the [**Memory**] tab in the bottom-left hand.
2. Click the [**Add Memory Monitor**] button. The [**Monitor Memory**] dialog will appear.
3. Select the [**Enter memory space and address**] radio button.
4. Select the [**CODE**] memory space from the combo box.
5. Enter an address to view, such as [**0x0000**].
6. Press [**OK**].
7. Notice that a new memory monitor has been added to the [**Memory**] view that displays the contents of CODE space starting at 0x0000.


Debug - myProject/src/Interrupts.c - Simplicity Studio ™
File Edit Source Refactor Navigate Search Project Run Window Help
Launcher
Simplicity IDE
Configurator
Debug
Debug
Silicon Labs 8051 MCU: C8051F850-C-GU
myProject.omf
ADC0EOC_ISR() at Interrupts.c:23 0x10
main() at myProject_main.c:22 0xa0
Variabl...
Breakp...
Registe...
Expre...
myProject.hwconf
Interrupts.c
myPro
Disassembly
Enter location here
Console
Memory
Executables
Monitors
CODE:0x0000
CODE:0x0000 : 0x0 <Traditional>
New Renderings...
Monitor Memory
Enter expression
Enter memory space and address
CODE
0x0000
OK
Cancel
1
2
3
4
5
6
7
SILICON LABS

Expression	Type	Value	Address
ADC0		Error: 'ADC0' not found	
Add new express			

```
// ADC0EOC_ISR
//-----------------------------------
//
// ADC0EOC ISR Content goes here. Remember to cle
// ADC0CN0::ADINT (Conversion Complete Interrupt
//
//-----------------------------------
SI_INTERRUPT (ADC0EOC_ISR, ADC0EOC_IRQn)
{
  ADC0CN0_ADINT = 0;
  PCA0CPH0 = (uint8_t    0 >> 2);
}
```

```
00000010:   MOV     A, 0BDH
00000012:   MOV     R6, 0BEH
00000014:   MOV     R0, #02H
00000016:   XCH     A, R6
00000017:   CLR     C
00000018:   RRC     A
00000019:   XCH     A, R6
0000001a:   RRC     A
0000001b:   DJNZ    R0, 16H
0000001d:   MOV     0FCH, A
24          }
```

```
0x00000000  C0690002 75D0C0E0 00C000D0 EDC206C0 BEAEBDE5 C3CE0278   ..iÀàÀÐuÐ.À.À.Âíå½®¾x.ÎÃ
0x00000018  D813CE13 D0FCF5F9 D000D006 32E0D0D0 12880012 0012A200   .    Ð.Ð.ÐÐà2.....¢..
0x00000030  8F001275 12460012 00127F00 AA001256 02A60012 BB759600   u    .....V..ª..¦...u»
0x00000048  FDBC750A 43F8E853 022282E8 C8E50300 C8530454 FFCB75FB   .u¼ýSèøCè."...åÈT.SÈûuËÿ
```

4. Revision History

4.1 Revision 0.3

September, 2019

Updated screenshots and code references to use si_toolchain.h macros.

4.2 Revision 0.2

June 13th, 2016

Updated formatting.

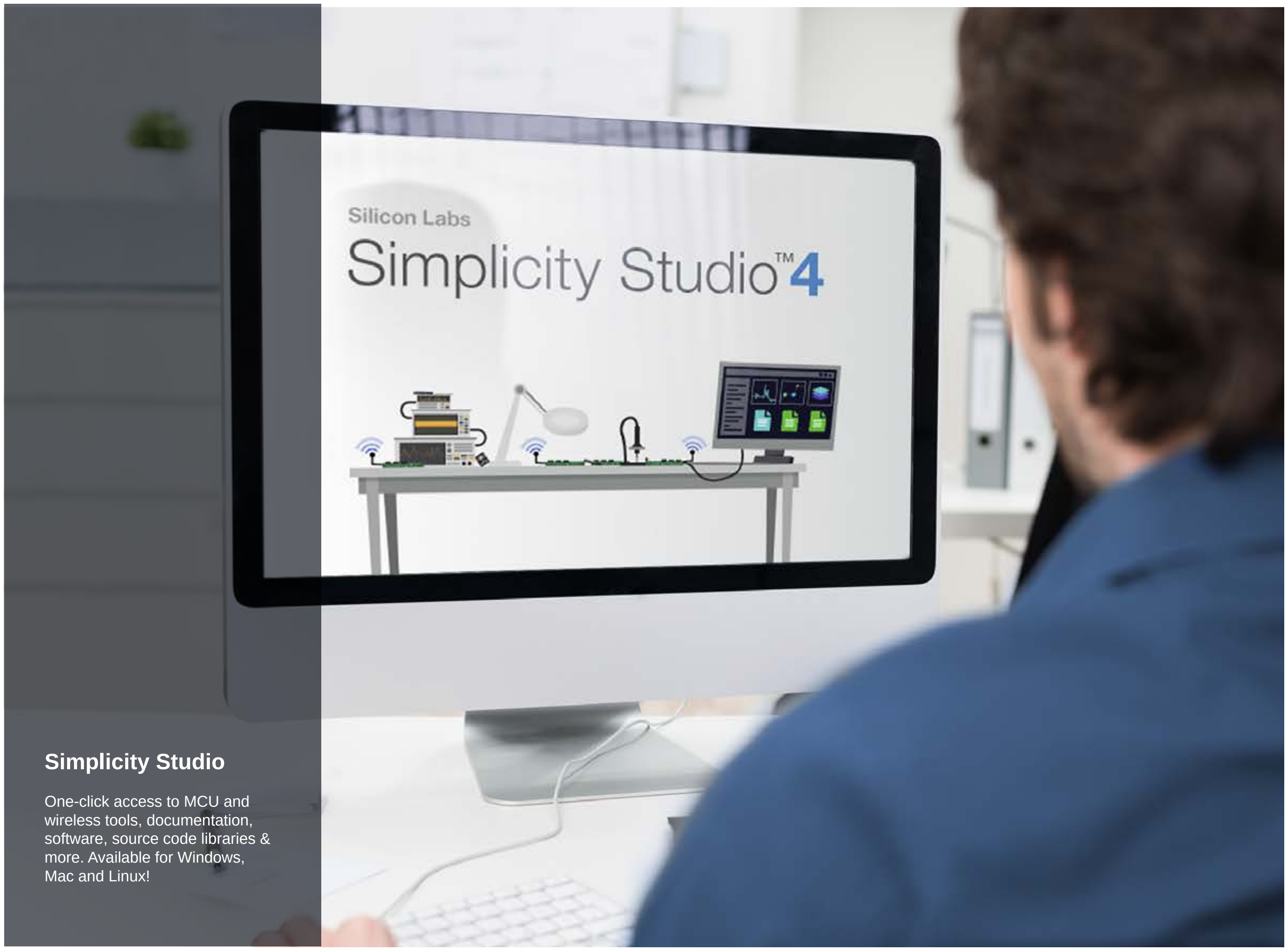
Updated screenshots for Simplicity Studio v4.

Reorganized initial setup chapters.

4.3 Revision 0.1

February, 2014

Initial revision.



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio *www.silabs.com/IoT*	**SW/HW** *www.silabs.com/simplicity*	**Quality** *www.silabs.com/quality*	**Support and Community** *community.silabs.com*

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com