PRECISION32™ IDE AND APPBUILDER DETAILED TUTORIAL AND WALKTHROUGH

1. Introduction

This document provides a step-by-step tutorial walkthrough for the Precision32 Development Tools (IDE and AppBuilder) using the **SiM3U1xx-B-DK** hardware and **Silicon Labs SDK version 1.0.1**. The first series in the walkthrough imports a pre-existing example into the IDE and discusses the IDE debugging basics. The second series guides the creation of a TIMER0 Blinky example using AppBuilder and adds some additional code editing and debugging tips for the IDE.

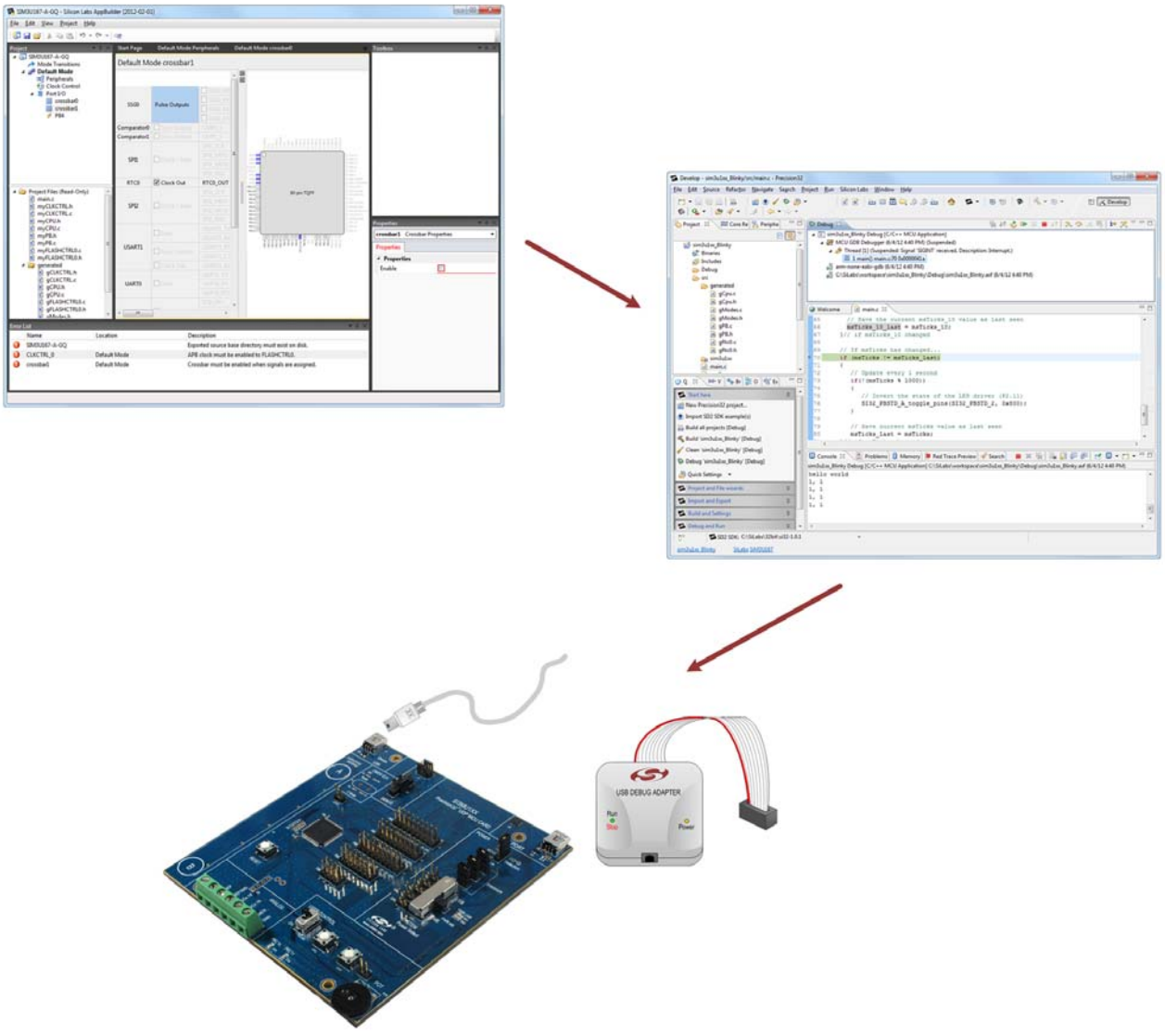


Figure 1. Precision32 IDE and AppBuilder Walkthrough Overview

2. Relevant Documentation

Precision32 Application Notes are listed on the following website: www.silabs.com/32bit-appnotes.

- **AN667: Getting Started with the Silicon Labs Precision32™ IDE**—provides a description of the IDE features and environment.
- **AN670: Getting Started with the Silicon Labs Precision32™ AppBuilder**—provides a description of the AppBuilder features.

3. Hardware Setup

To set up the hardware for this walkthrough:

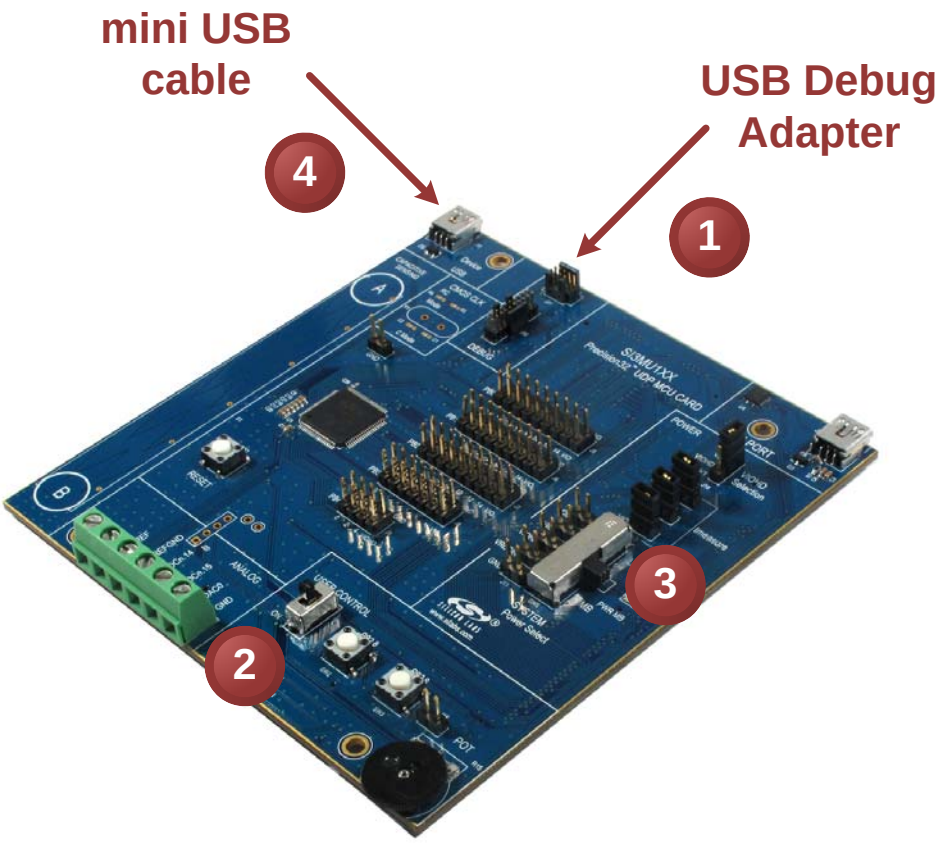


Figure 2. Hardware Setup Diagram

1. Connect the USB Debug Adapter to J31 and connect the Debug Adapter to the PC with a USB cable.
2. Move the **USER CONTROL** MCU card switch (SW4) to ON.
3. Move the **SYSTEM Power Select** switch (SW5) to the USB position.
4. Connect a mini USB cable to the Device USB connector (J13). Connect the cable to the PC to power the board.

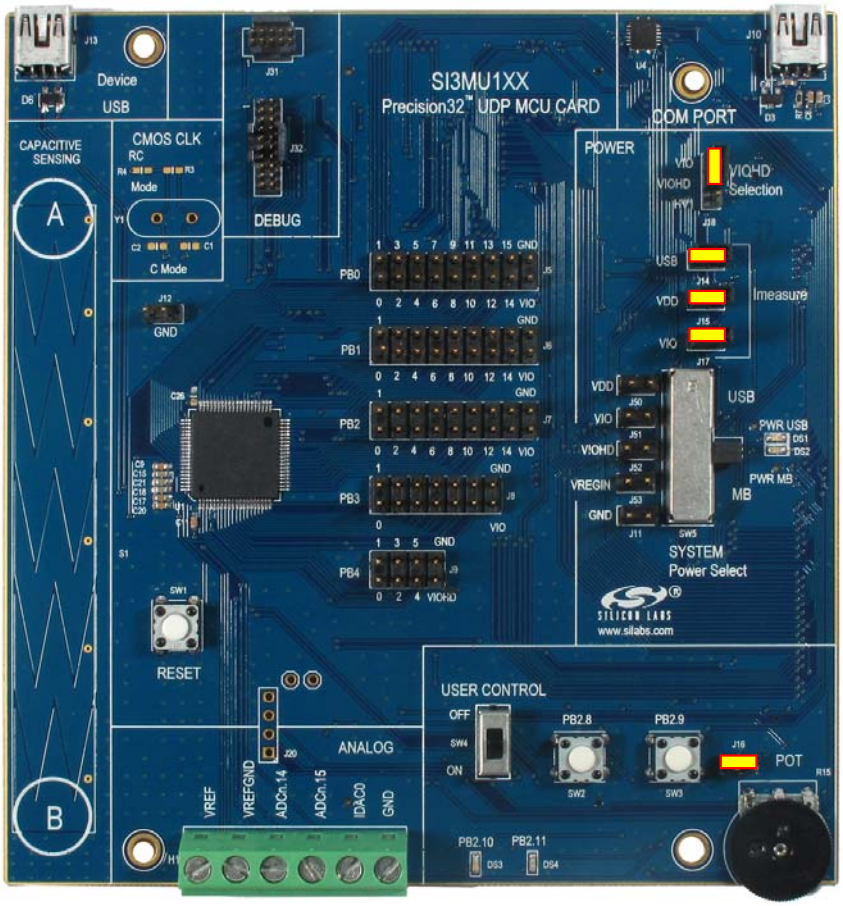


Figure 3. Shorting Block Settings

5. Verify the shorting blocks are installed as shown in Figure 3:
   a. J18 top two pins
   b. J14
   c. J15
   d. J17
   e. J16

4. Lab Software Setup

1. Install the Precision32 package from www.silabs.com/32bit-software.
2. Open the Precision32 IDE and activate it, if it's not activated already. Instructions on how to do this are available on the Welcome page in the IDE.

5. Walkthrough Series #1: Running Blinky in the Precision32 IDE

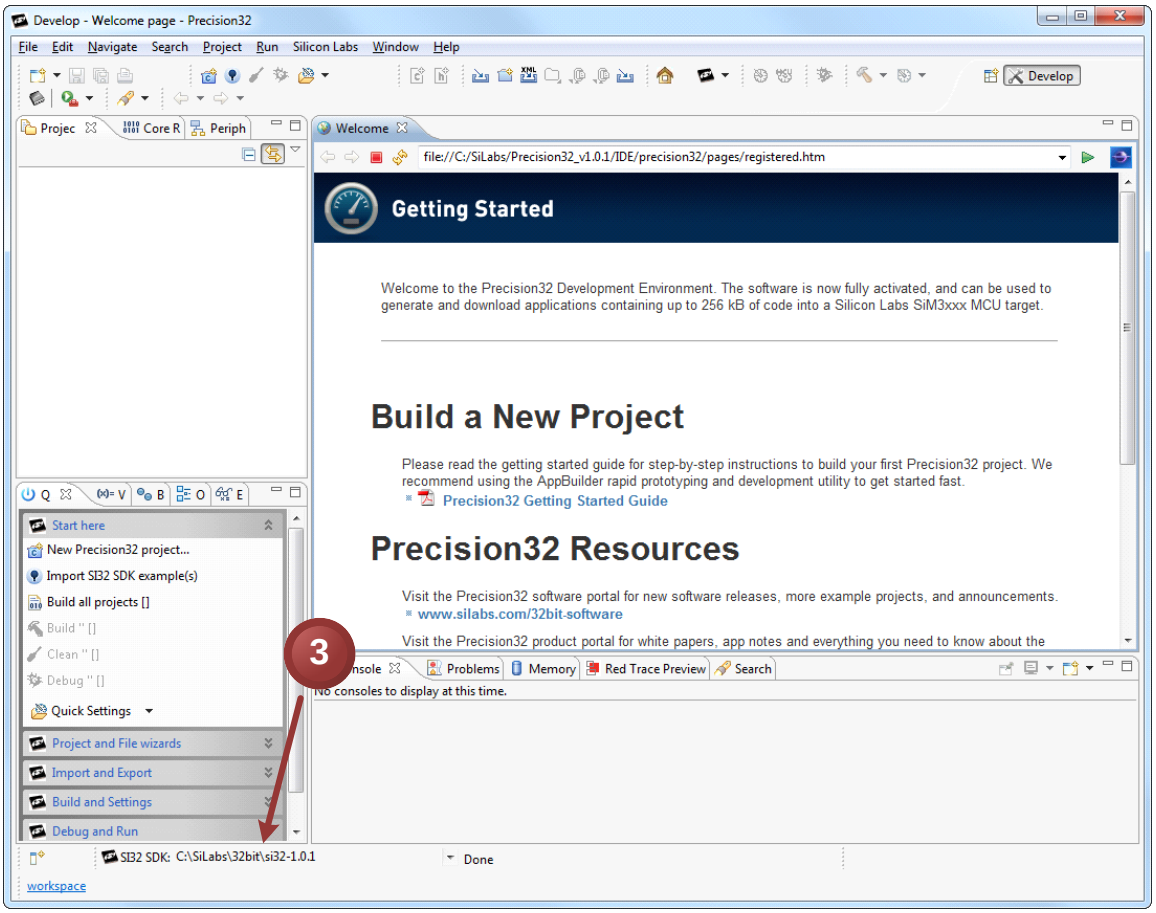
5.1. Objectives

The objectives of series are:

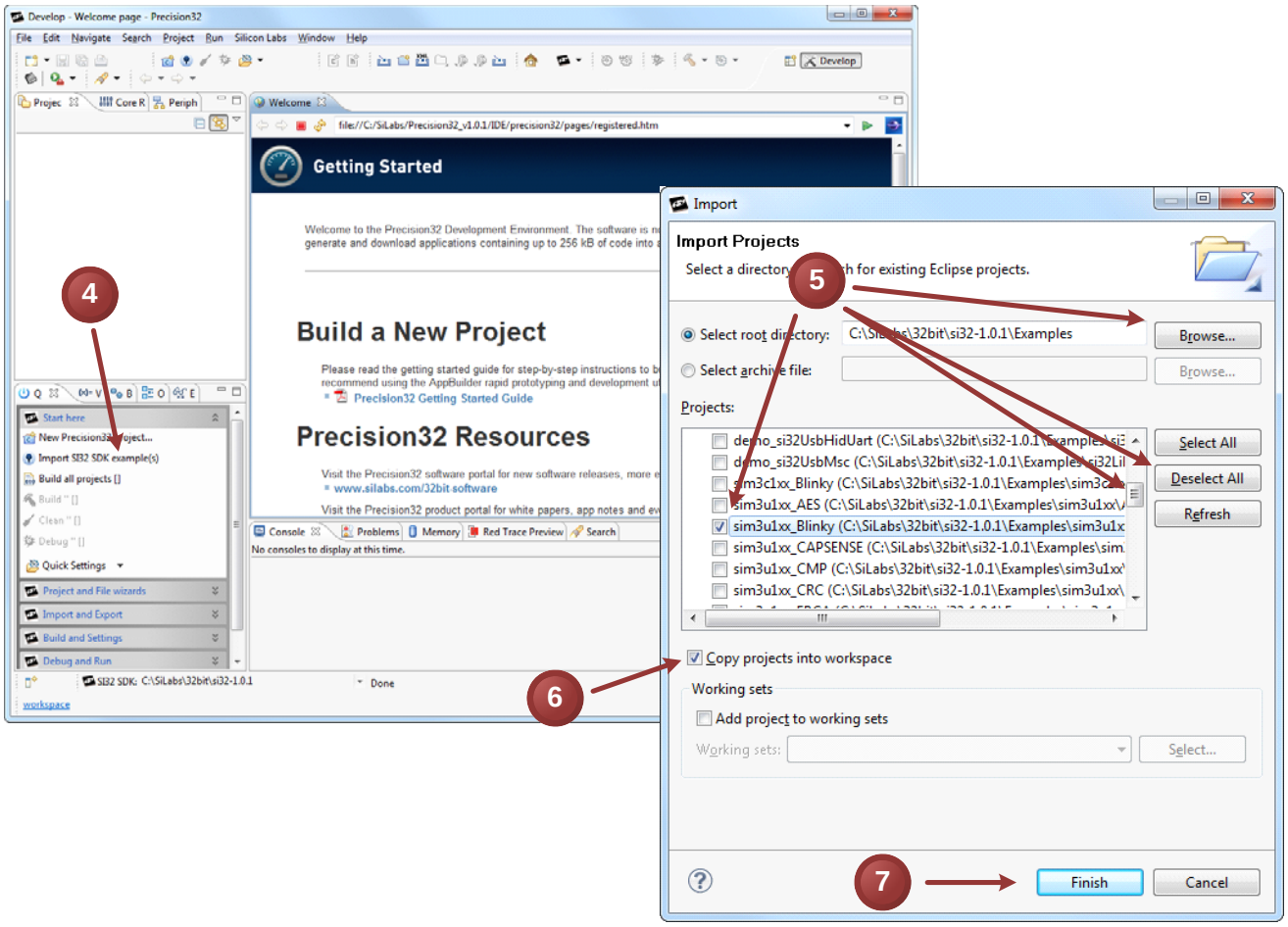
- Learn how to import an SDK example into the Precision32 IDE.
- Learn how to download a program to a device.
- Explore some of the debugging features of the IDE.

5.2. Instructions

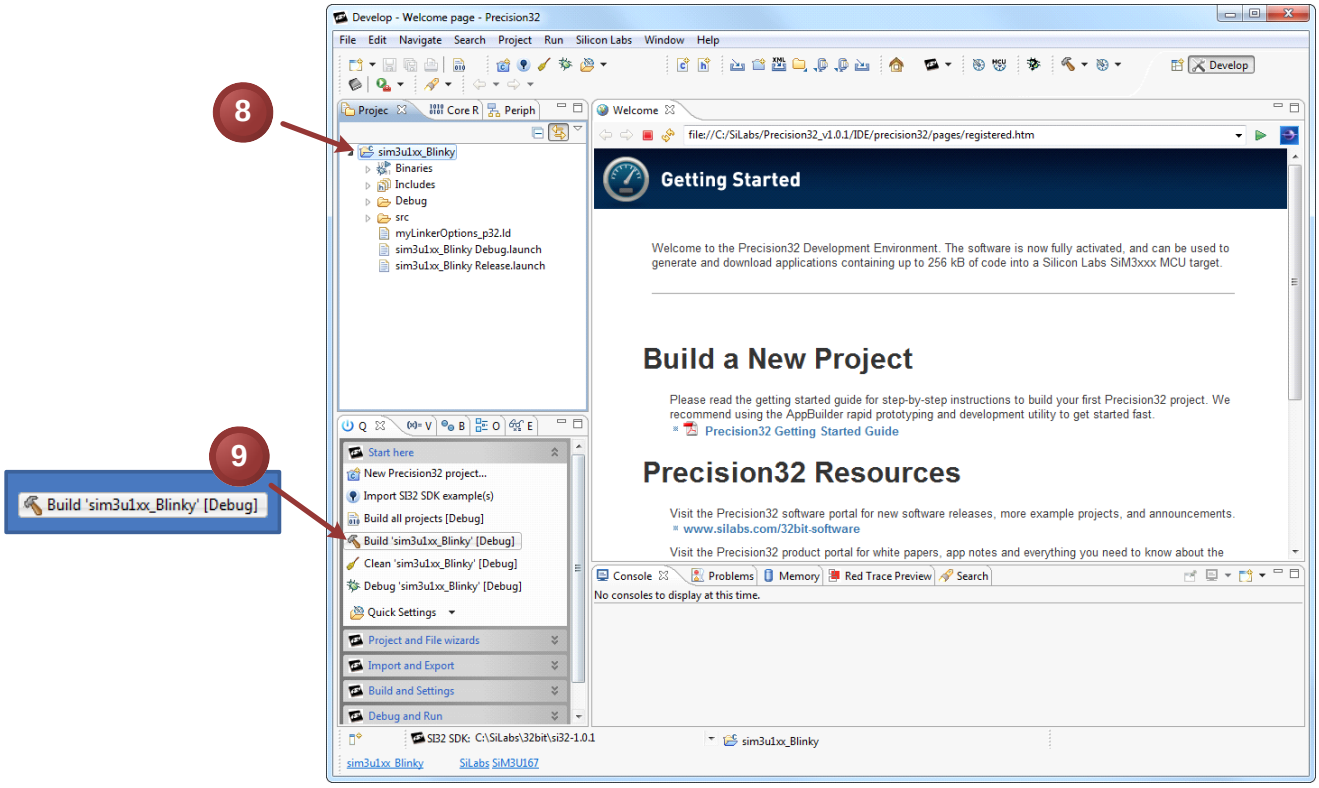
1. Launch the **Precision32 IDE 1.0.1** (default path: C:\SiLabs\Precision32_v1.0.1\IDE) and select a workspace (**C:\SiLabs\workspace**).
2. Activate the IDE using the instructions on the landing page, if the IDE isn't already activated.
3. Set the Silicon Labs SDK path to **C:\SiLabs\32bit\si32-1.0.1**.



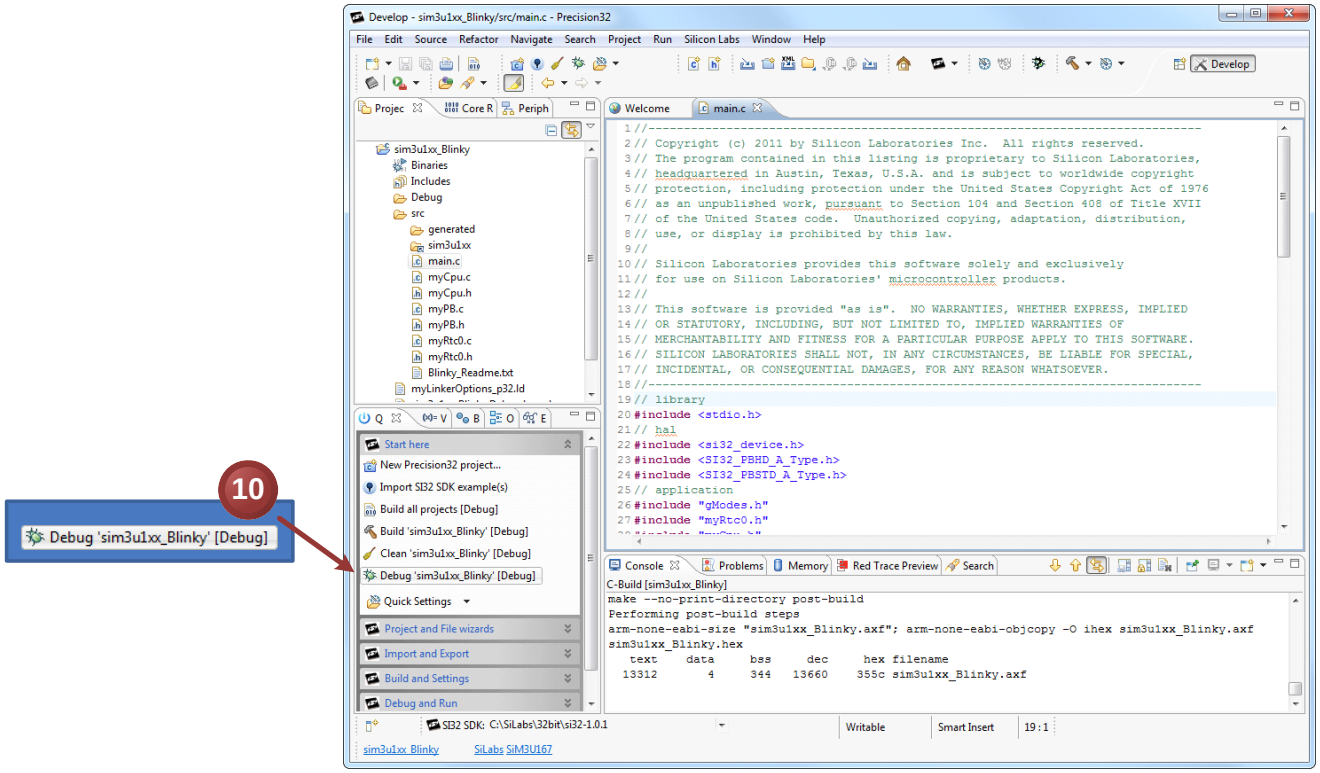
4. Click the **Import SI32 SDK example(s)** link in the **Quickstart** view.
5. Scroll down and check just the **sim3u1xx_Blinky** example. The default path for importing projects is C:\SiLabs\32bit\si32-x.y.z\Examples, where x.y.z is the version of the SDK selected in the previous step, but the **Browse...** button can be used to navigate to a specific example directory. The **Deselect All** button is also useful for deselecting a long set of examples.
6. Ensure the **Copy projects into workspace** checkbox is enabled.
7. Click **Finish.**



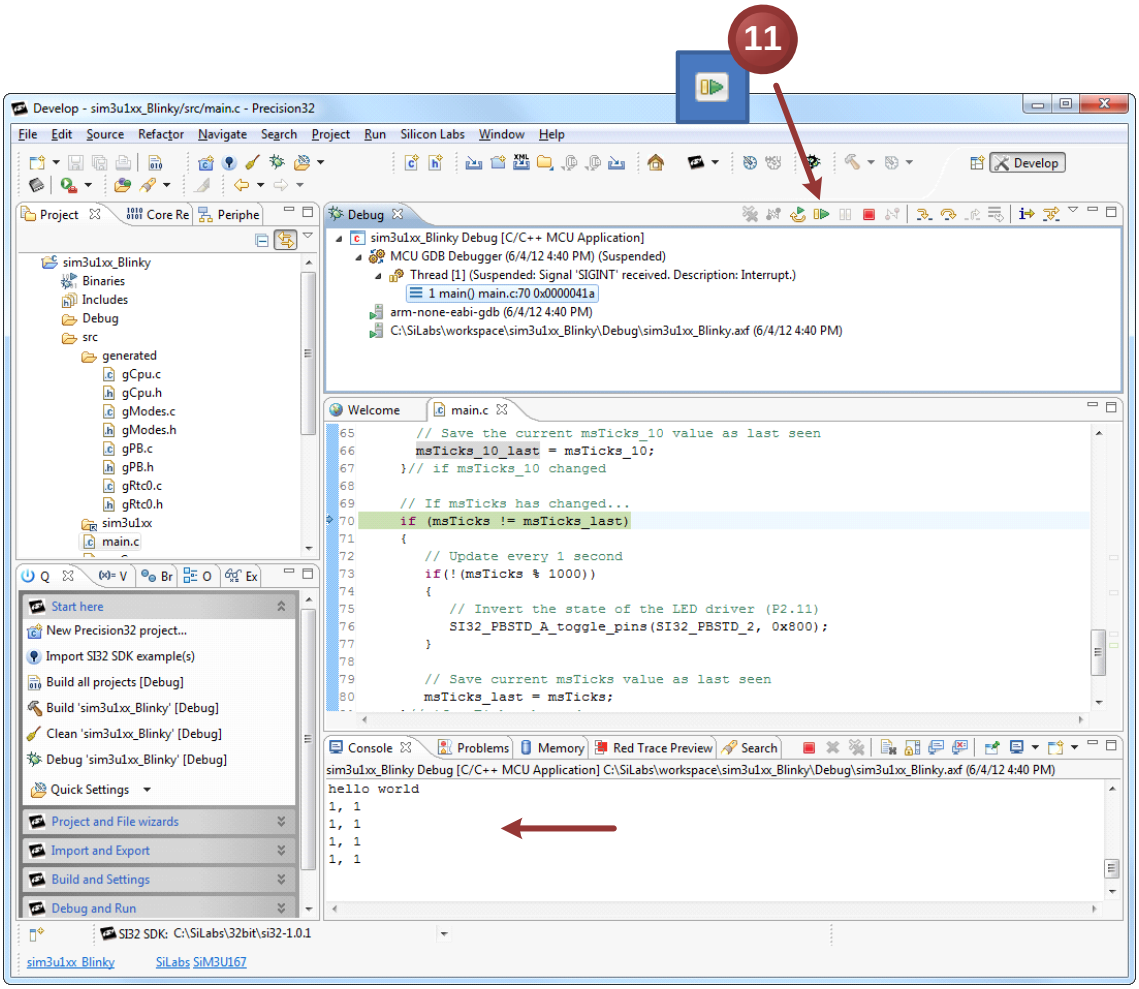
8. Click the **sim3u1xx_Blinky** project name.
9. Click the **Build 'sim3u1xx_Blinky' [Debug]** link.



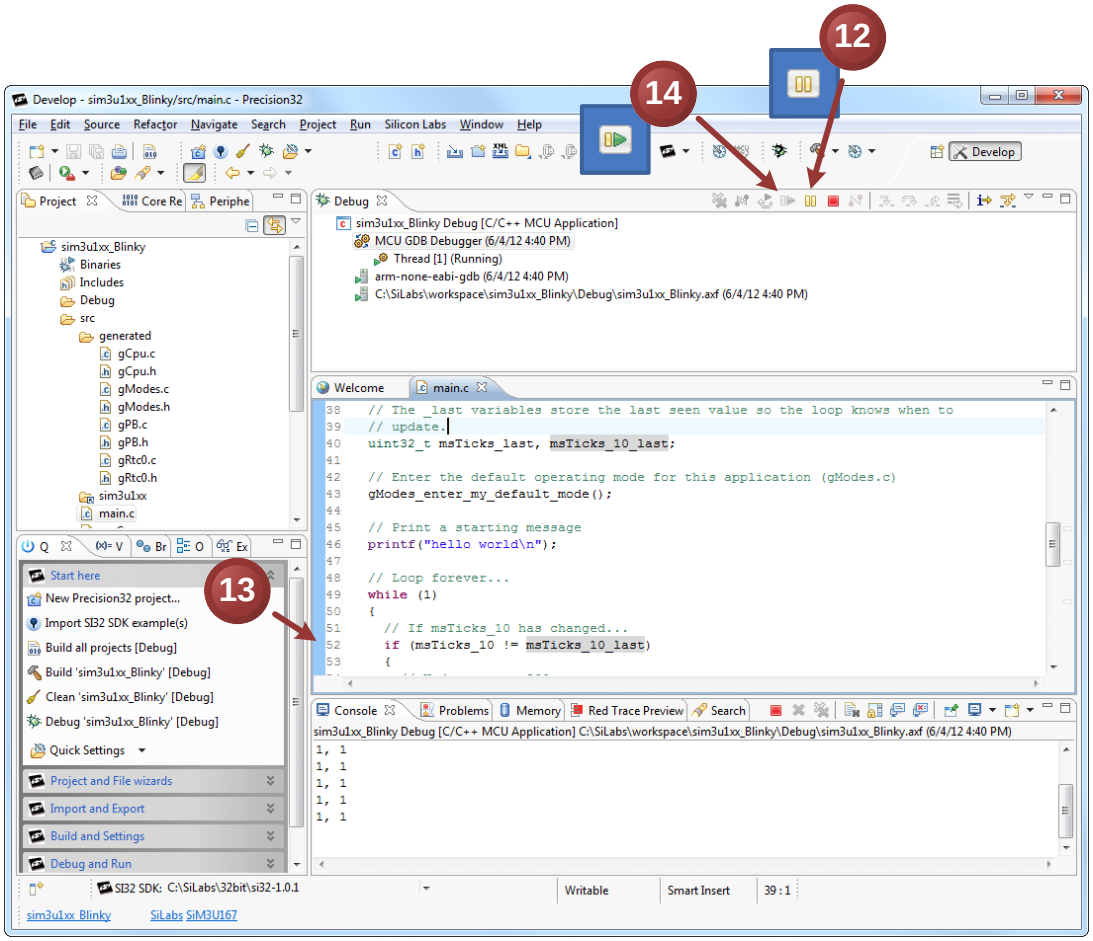
10. Click the **Debug 'sim3u1xx_Blinky' [Debug]** link. The **Debug** view will open with trace information.



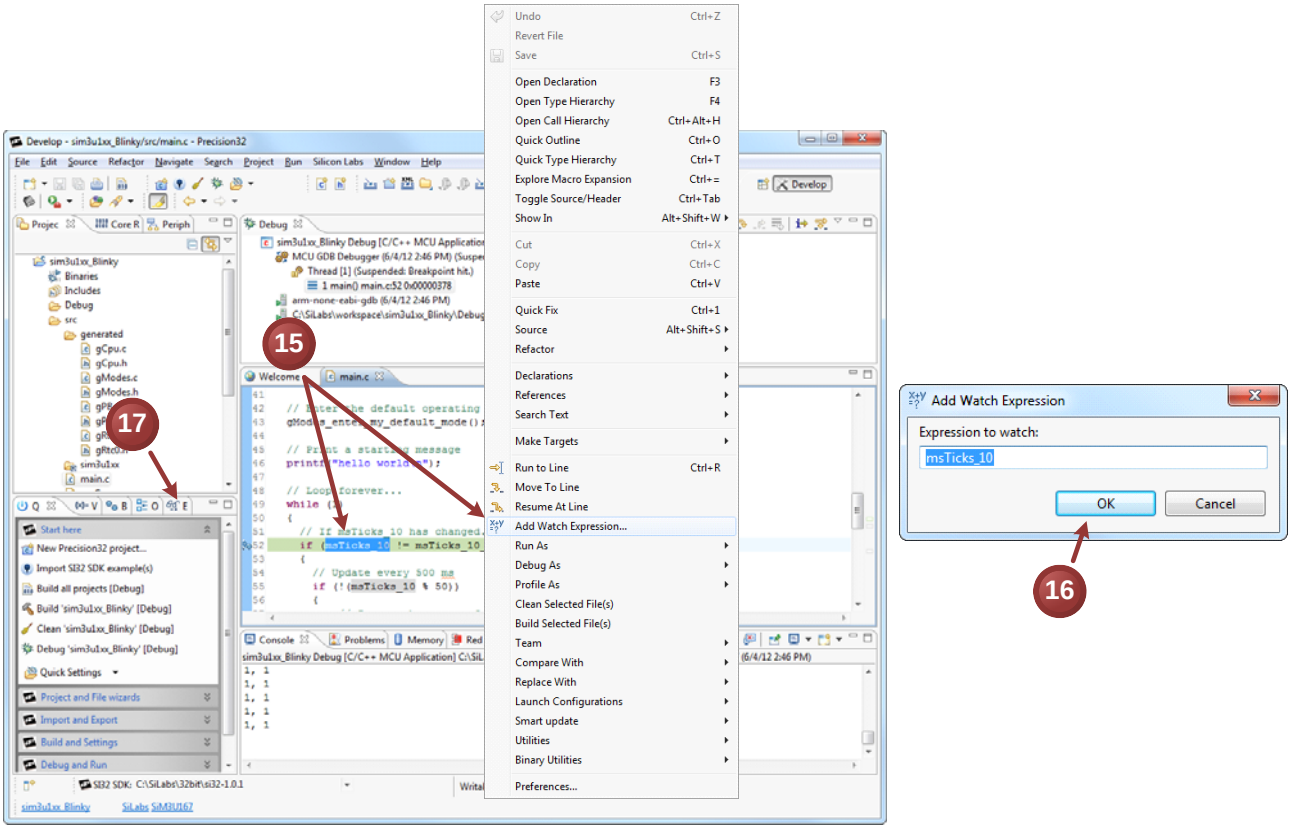
11. Start the program. PB2.10 and PB2.11 will blink after a small delay. The printf() output from the example will appear in the **Console** view.



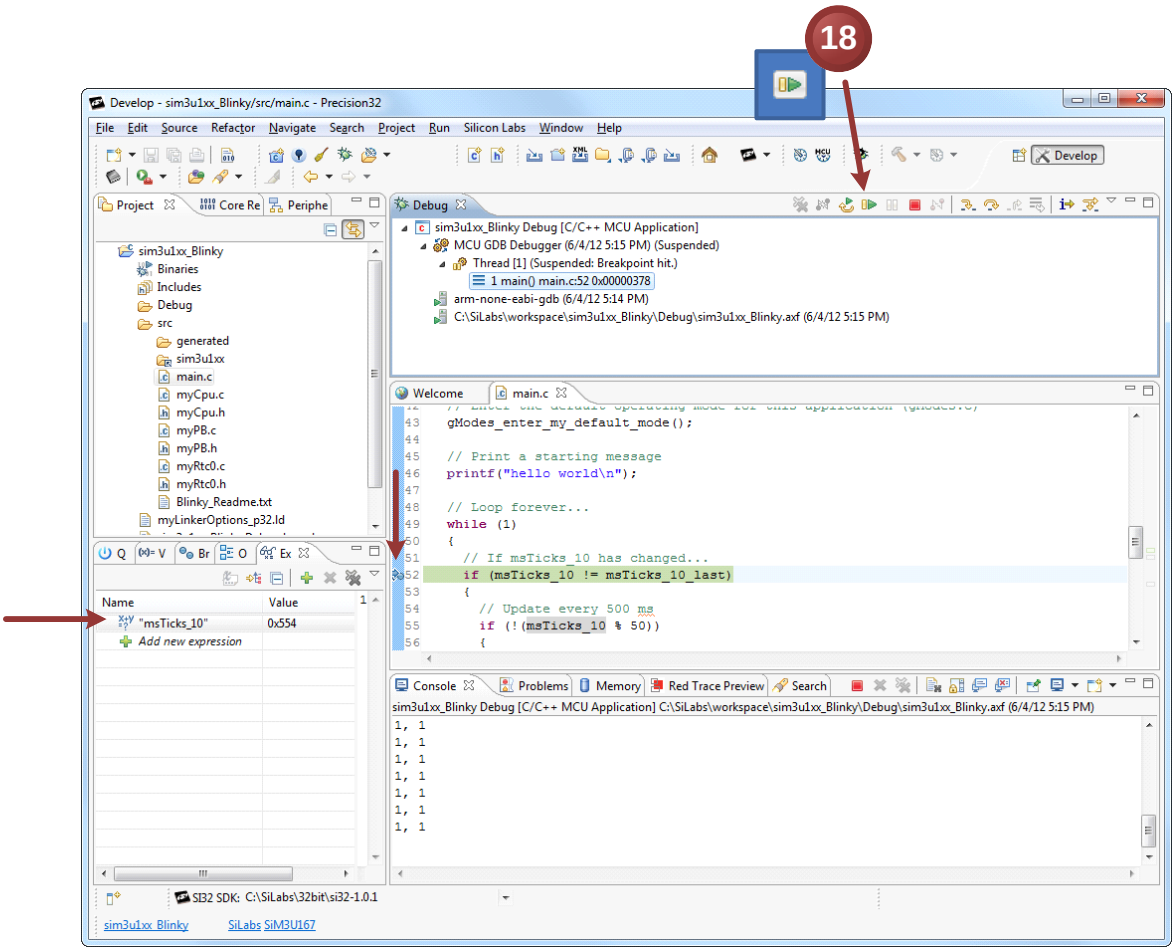
12. Pause the program.
13. Double-click the blue area next to a line to add a breakpoint.
14. Start the program again.



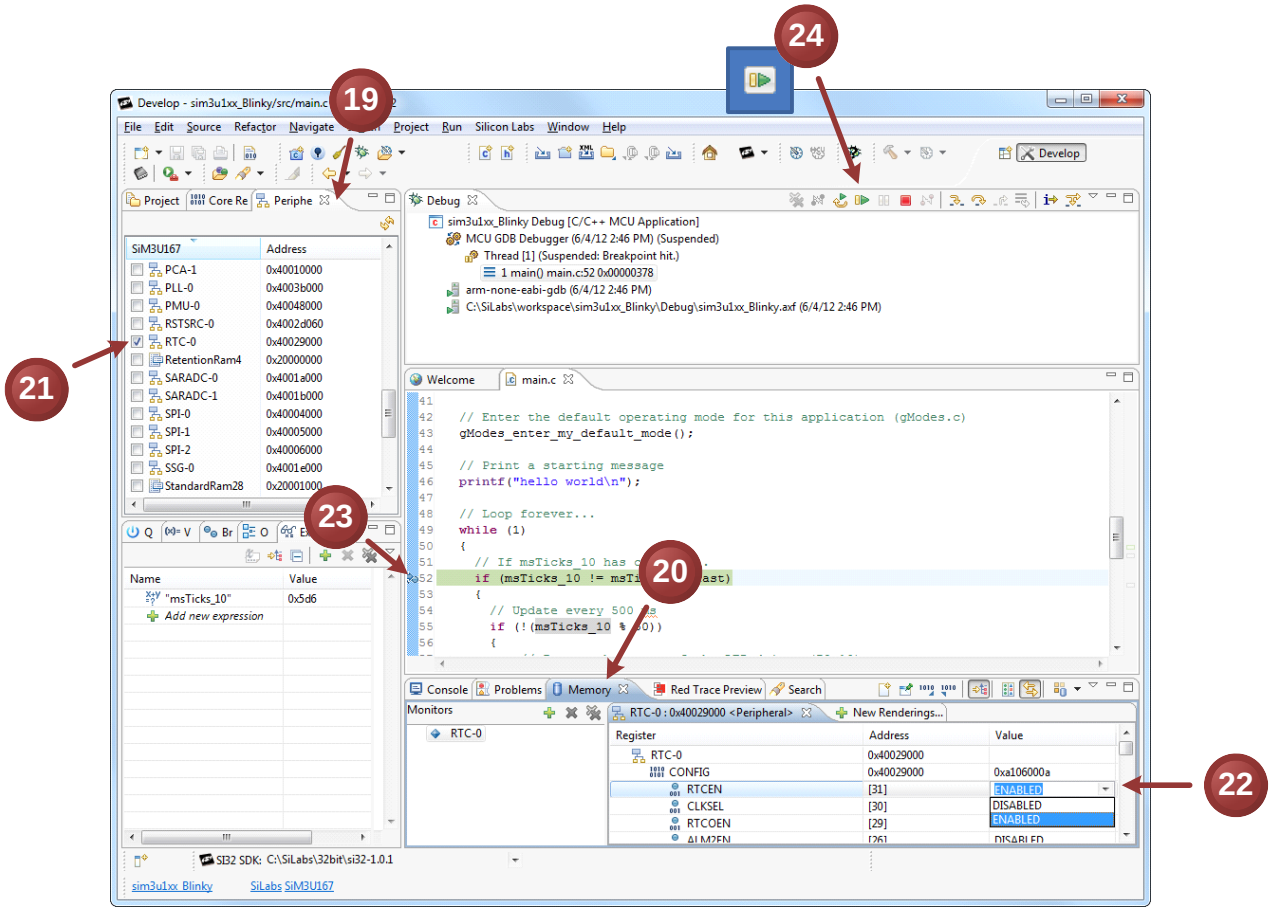
15. Highlight the **msTicks_10** variable in the **main.c** file, right click, and select **Add Watch Expression...**.
16. Press OK in the **Add Watch Expression** dialog.
17. Select the **Expression** view.



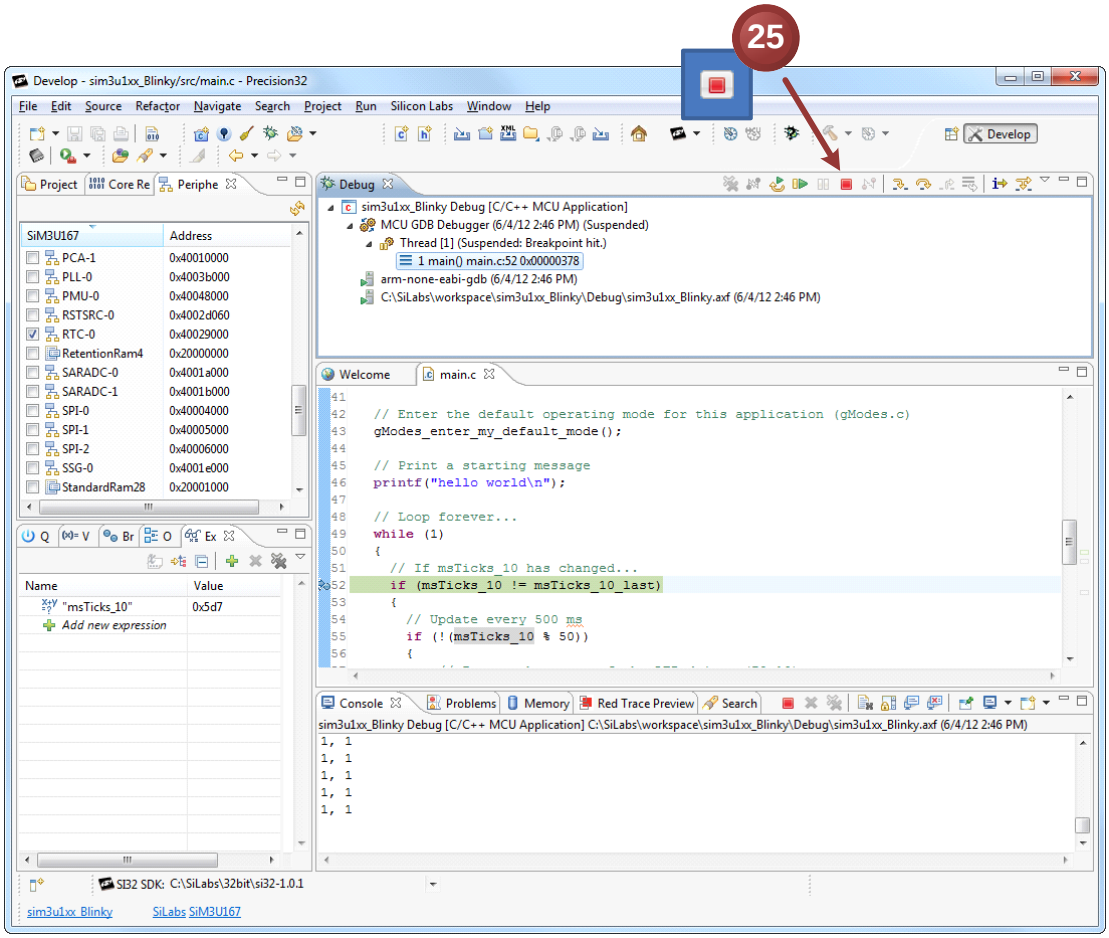
18. Click the **Resume** button a few times. The program will stop each time at the breakpoint and update the value of **msTicks_10**.



19. Select the **Peripheral** view.
20. Select the **Memory** view.
21. Enable the **RTC-0** checkbox.
22. Disable the RTC by selecting **DISABLED** for the **RTCEN** bit.
23. Double-click on the breakpoint to remove it.
24. Start the program again. The PB2.10 LED will stop blinking since the RTC is no longer running. PB2.11 still blinks because it's running from the SysTick timer.



25. Click the **Terminate** button to end the debugging session.



6. Walkthrough Series #2: Creating a Timer Blinky

6.1. Objectives

The objectives of this series are:

- Use AppBuilder to configure the TIMER0 high 16-bit timer to generate a 12 Hz waveform on PB2.10 (red LED).
- Use the information learned in Lab 1 to download and debug the generated AppBuilder project.

6.2. Lab Preparation

The default SiM3U1xx oscillator is the Low Power Oscillator at 20 MHz, but this doesn't create a 12 Hz waveform cleanly:

$$\frac{20000000}{12} = 1666666.67$$

Instead, we can use the 48 MHz USB0 oscillator. The actual overflow rate desired for the timer is 12 Hz multiplied by two, since the timer must overflow for each rising and falling edge of the waveform on the pin. This means our TIMER0 effective overflow rate is:

$$\frac{48000000}{12 \times 2} = 2000000$$

The TIMER0 module has two ways to meet this overflow rate. The module can clock from a divided version of APB. This prescaler (CLKDIV and CLKDIVRL) is an integer value and can be arbitrarily chosen between 0 and 255 (effectively creating APB divided by 256 up to APB divided by 1). In addition, the Timer count interval is 16 bits in split mode, so it can be anywhere between 0 and 65535 (setting the timer count to 0xFFFF causes the timer to overflow in one clock). We can choose these values in any way such that they equal the required count of 2000000.

If we choose to set the TIMER0 APB prescaler to 40 (for a TIMER0 clock of 1.2 MHz):

$$\frac{2000000}{40} = 50000$$

Prescaler (CLKDIVRL) value from the reference manual:

$$F_{prescale} = \frac{F_{APB}}{256 - CLKDIVRL}$$

$$256 - CLKDIVRL = 256 - 40 = 216$$

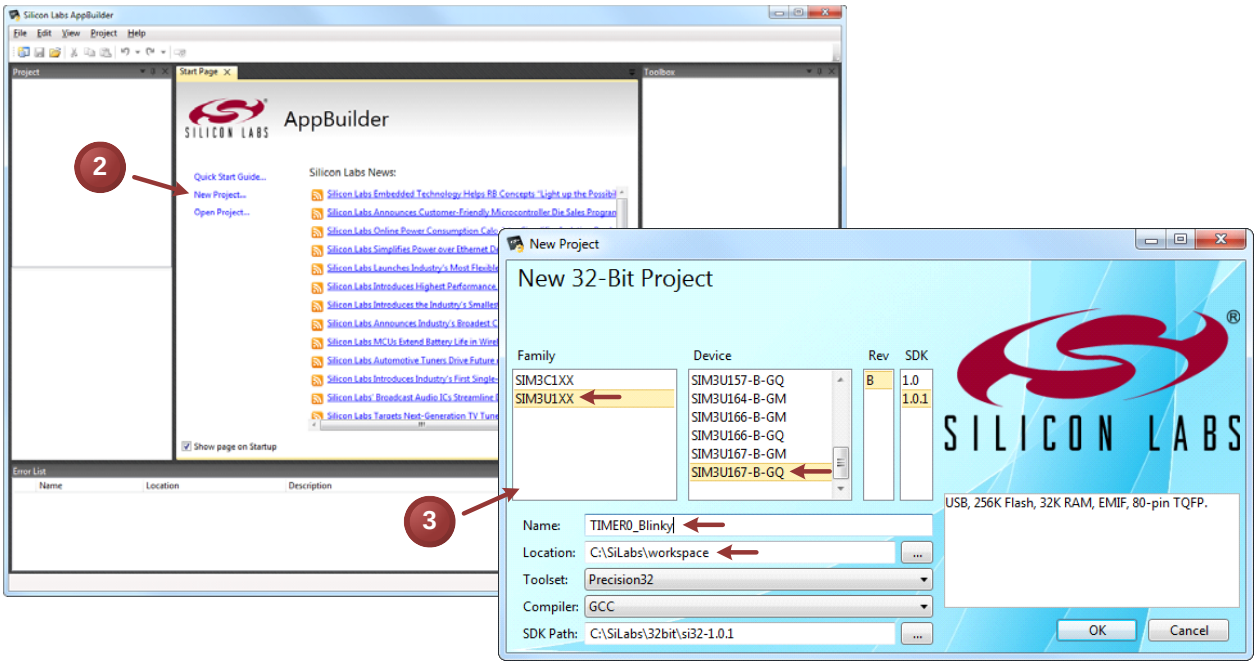
16-bit reload value (HCCR) from the reference manual:

$$\text{Overflow Rate} = \frac{F_{TIMER}}{65536 - HCCR}$$

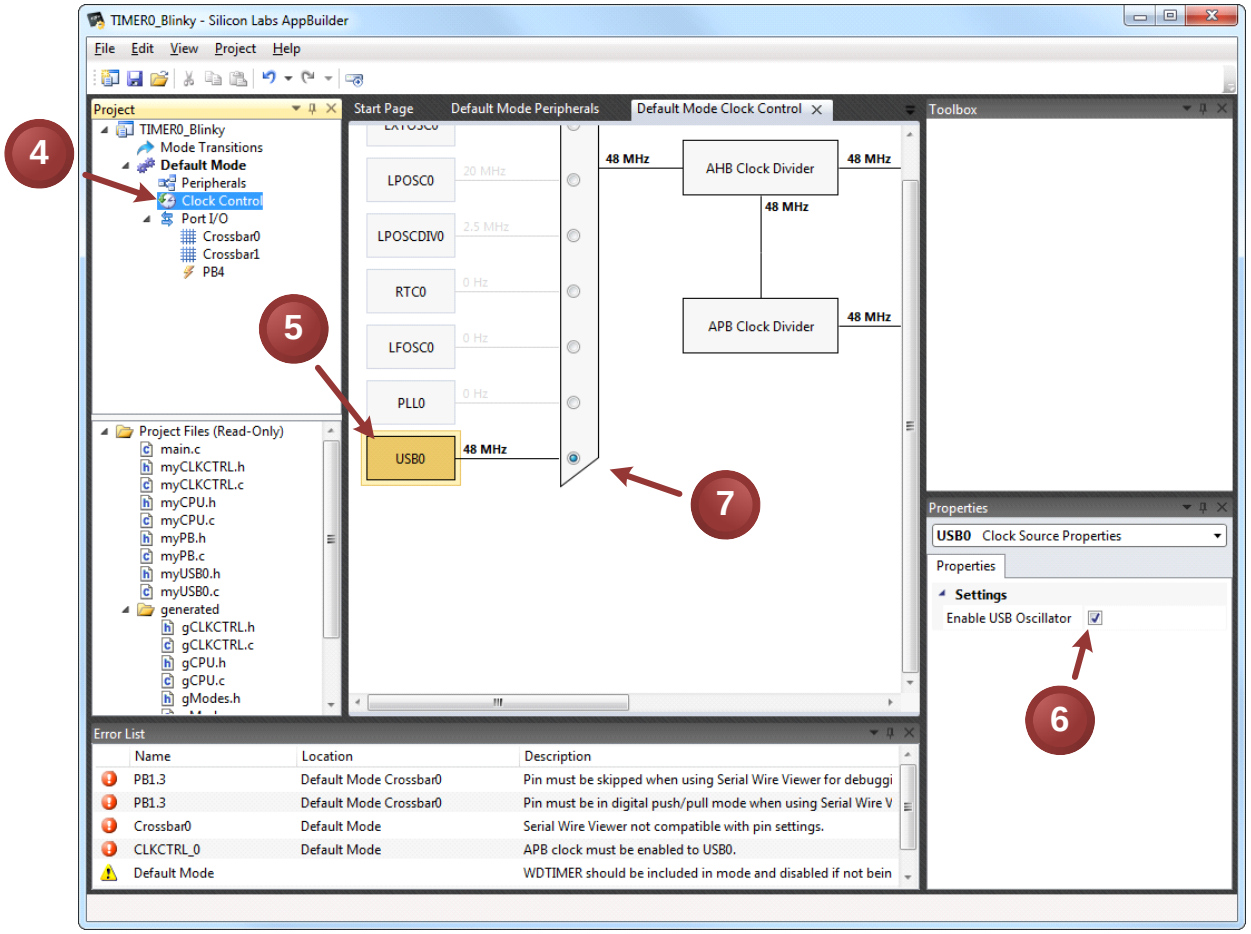
$$65536 - HCCR = 65536 - 50000 = 15536$$

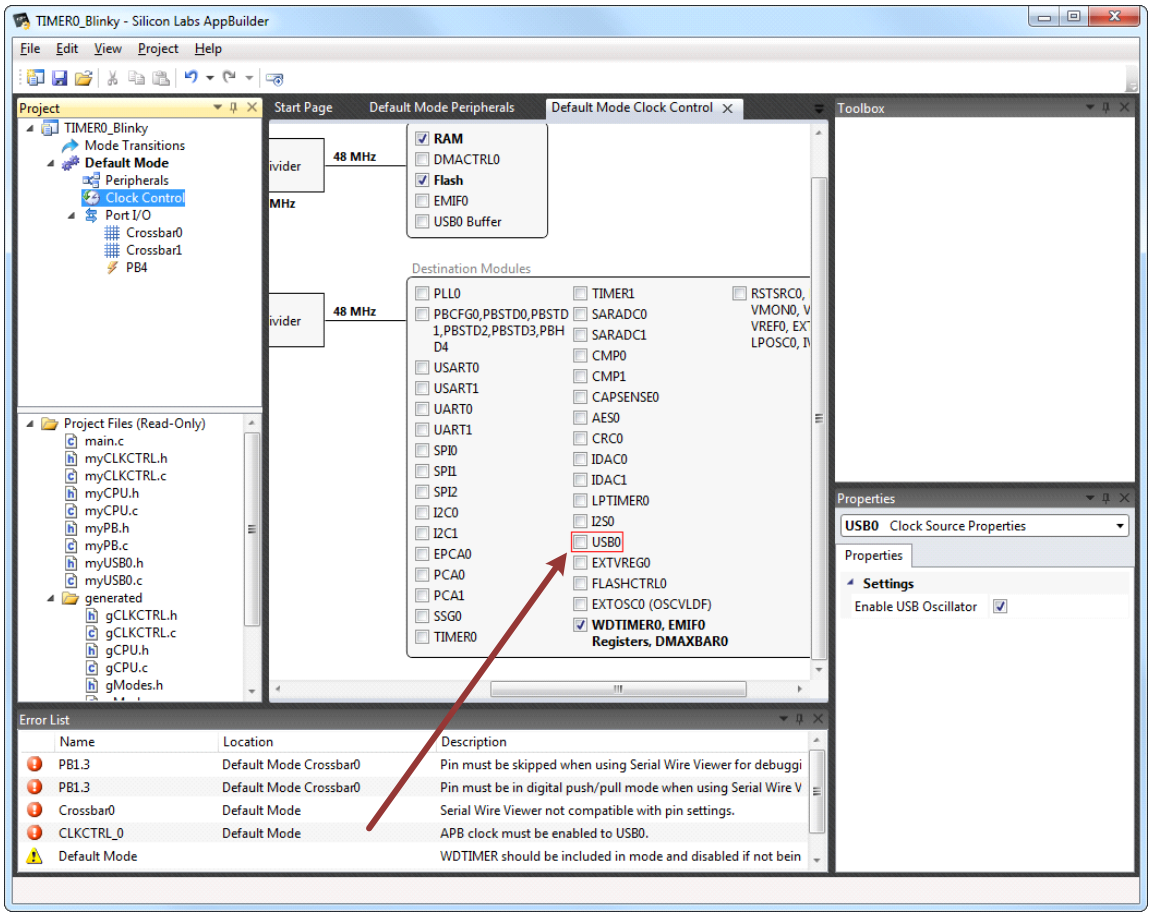
6.3. Instructions

1. Launch **AppBuilder 1.0.1** (default path: C:\SiLabs\Precision32_v1.0.1\AppBuilder).
2. Select **New Project...**.
3. Select the project settings. **SiM3U1xx→SiM3U167-B-GQ→B→1.0.1**. Update the **Name** of the project and **Location** to the workspace location. Press **OK**.

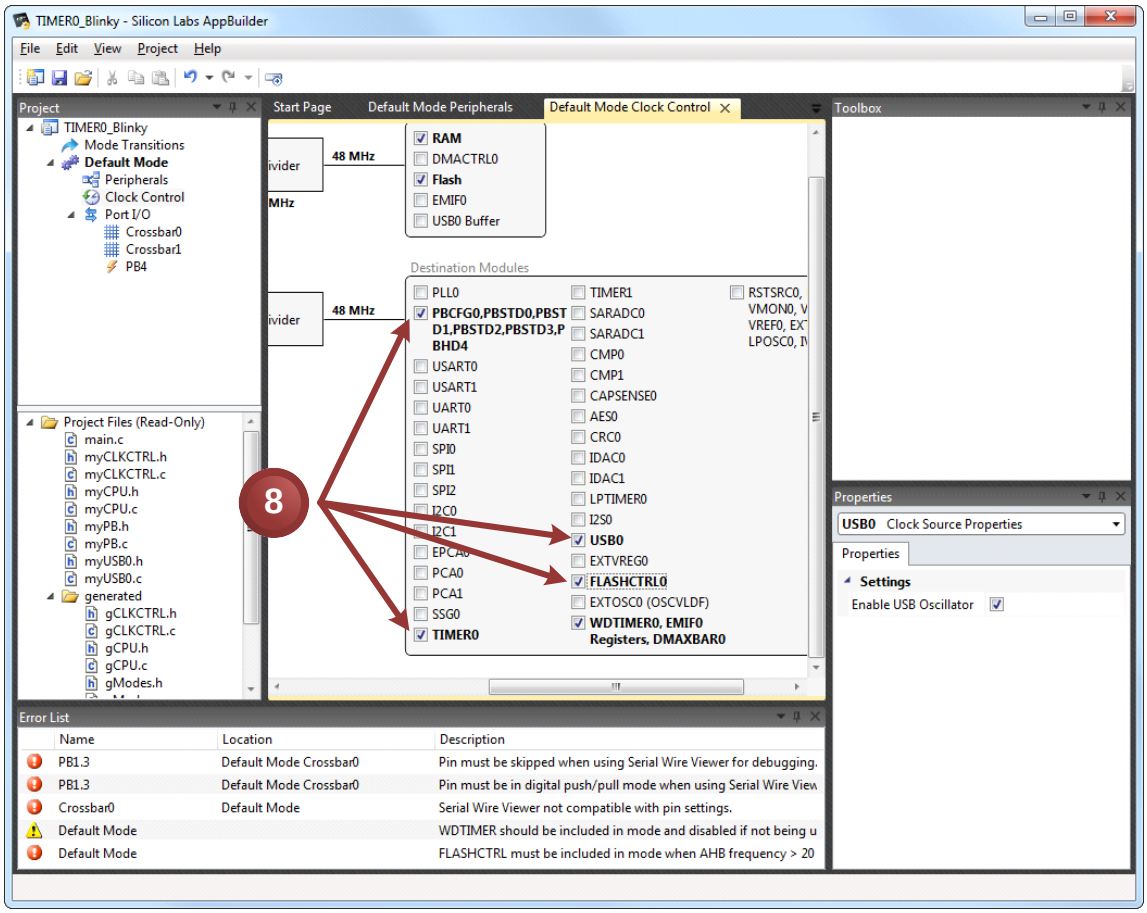


4. Double-click on **Clock Control**.
5. Select the **USB0** Oscillator.
6. Enable the USB0 Oscillator in the **Properties** window.
7. Click on the MUX radial button to select the USB0 oscillator as the AHB clock source. The diagram updates to show the current AHB and APB clock frequencies. After selecting the USB0 clock as the AHB source, an error appears indicating that the USB0 APB clock must be enabled.

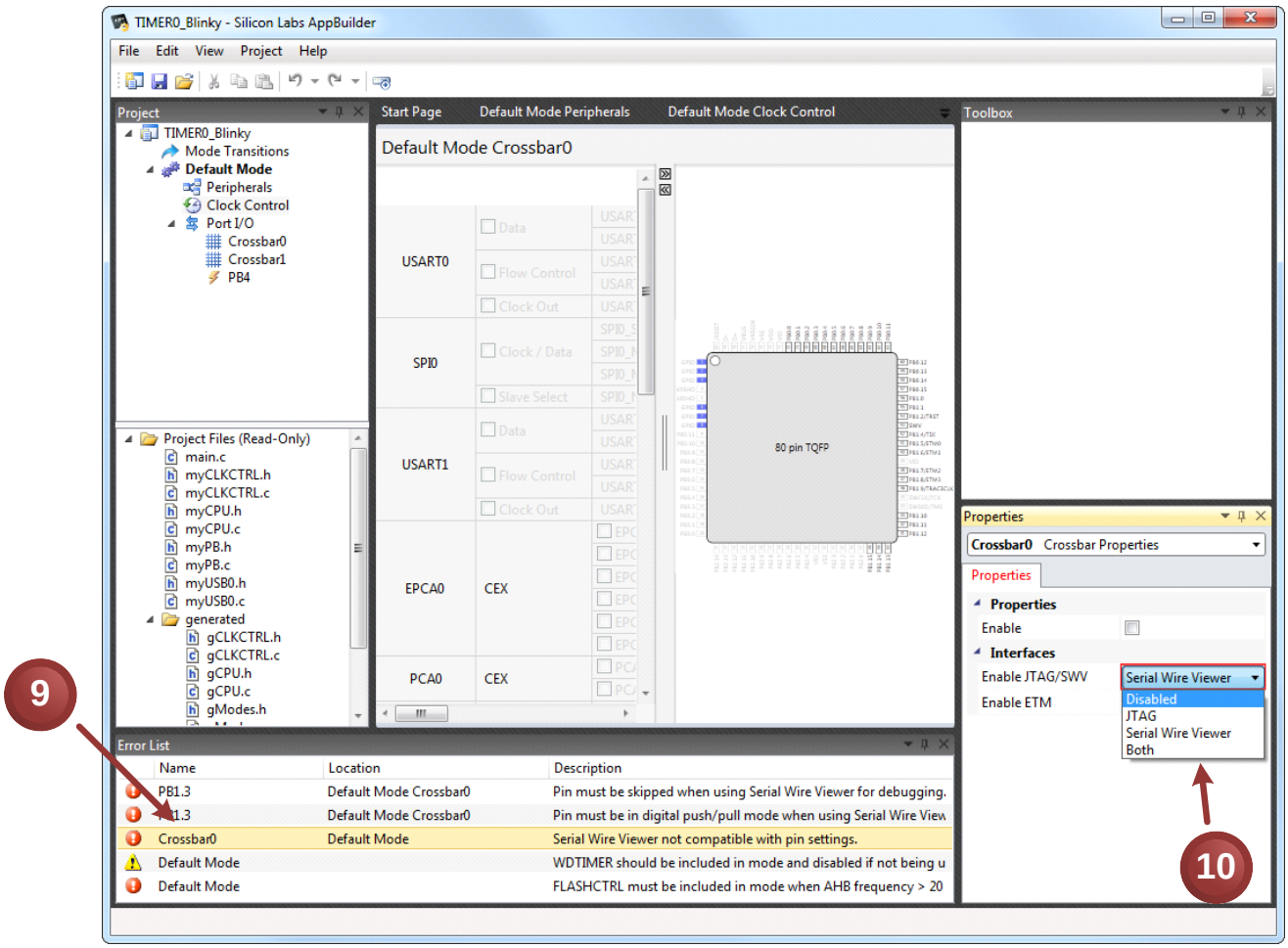



TIMER0_Blinky - Silicon Labs AppBuilder
File Edit View Project Help
Project
TIMER0_Blinky
Mode Transitions
Default Mode
Peripherals
Clock Control
Port I/O
Crossbar0
Crossbar1
PB4
Start Page
Default Mode Peripherals
Default Mode Clock Control
48 MHz
RAM
DMACTRL0
Flash
EMIF0
USB0 Buffer
Destination Modules
PLL0
PBCFG0,PBSTD0,PBSTD1,PBSTD2,PBSTD3,PBHD4
USART0
USART1
UART0
UART1
SPI0
SPI1
SPI2
I2C0
I2C1
EPCA0
PCA0
PCA1
SSG0
TIMER0
TIMER1
SARADC0
SARADC1
CMP0
CMP1
CAPSENSE0
AES0
CRC0
IDAC0
IDAC1
LPTIMER0
I2S0
USB0
EXTVREG0
FLASHCTRL0
EXTOSC0 (OSCVLDF)
WDTIMER0, EMIF0 Registers, DMAXBAR0
Toolbox
Properties
USB0 Clock Source Properties
Settings
Enable USB Oscillator
Project Files (Read-Only)
main.c
myCLKCTRL.h
myCLKCTRL.c
myCPU.h
myCPU.c
myPB.h
myPB.c
myUSB0.h
myUSB0.c
generated
gCLKCTRL.h
gCLKCTRL.c
gCPU.h
gCPU.c
gModes.h
Error List
Name
Location
Description
PB1.3
Default Mode Crossbar0
Pin must be skipped when using Serial Wire Viewer for debuggi
PB1.3
Default Mode Crossbar0
Pin must be in digital push/pull mode when using Serial Wire V
Crossbar0
Default Mode
Serial Wire Viewer not compatible with pin settings.
CLKCTRL_0
Default Mode
APB clock must be enabled to USB0.
Default Mode
WDTIMER should be included in mode and disabled if not bein

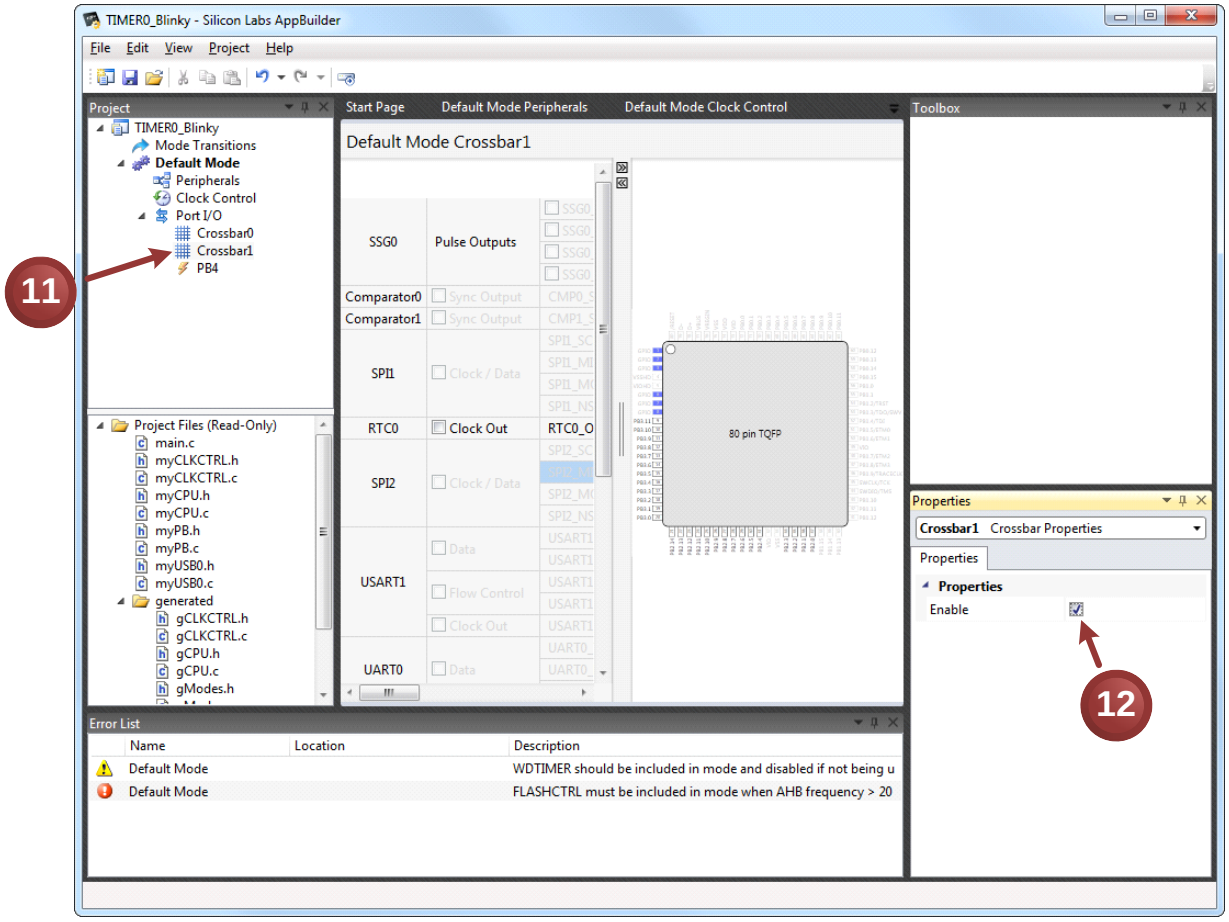
8. Enable the APB clocks to the **Ports (PBCFG0, PBSTD0, PBSTD1, PBSTD2, PBSTD3, PBHD4)**, **TIMER0**, **USB0**, and **FLASHCTRL0**. The error and red highlight around USB0 disappear after enabling the USB0 APB clock.



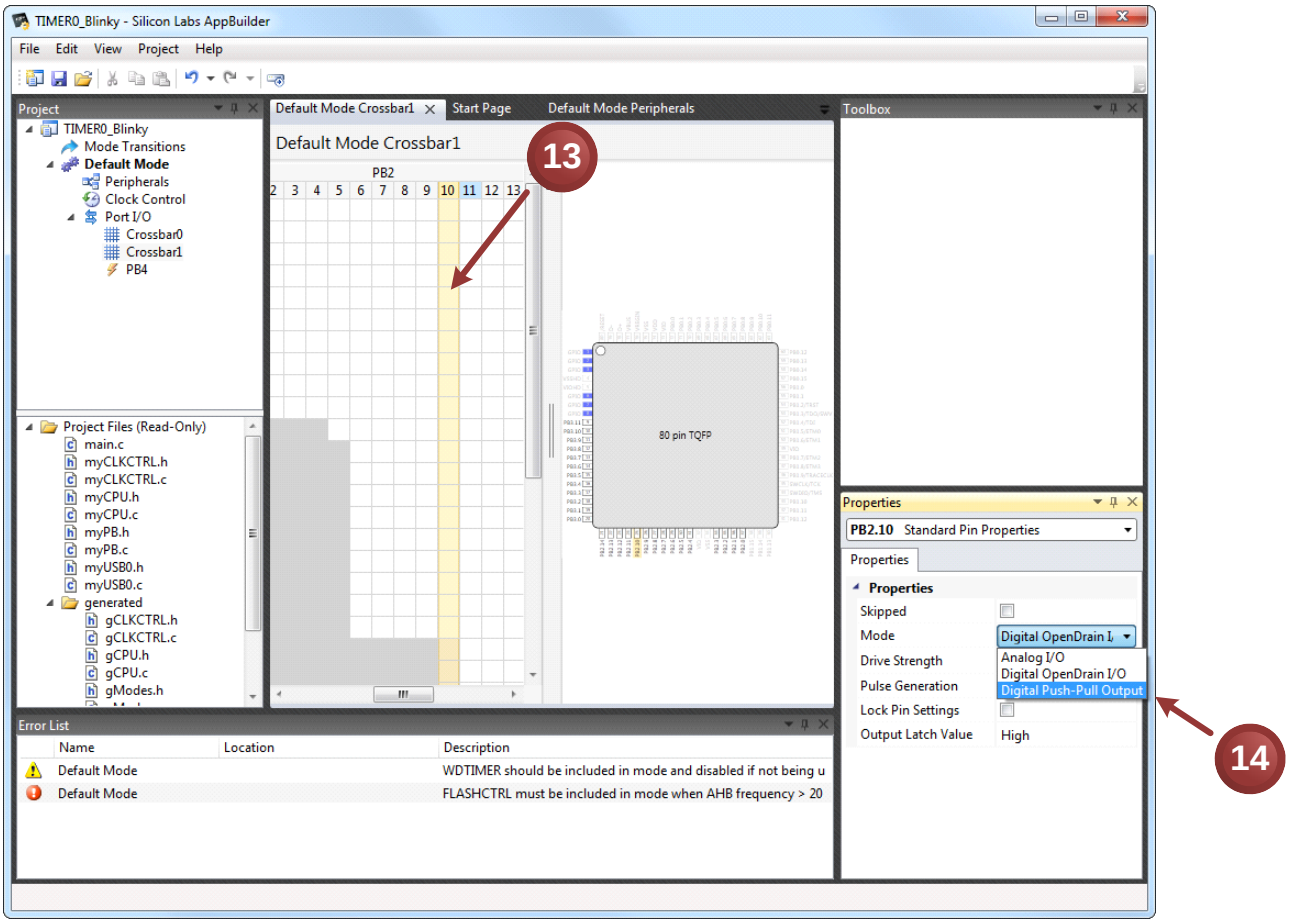
9. We can remove some errors and disable SWV since we're not using it for this application. Double-click on the crossbar 0 error in the **Error List** window.
10. Select **Disabled** in the **Enable JTAG/SWV** drop-down menu.



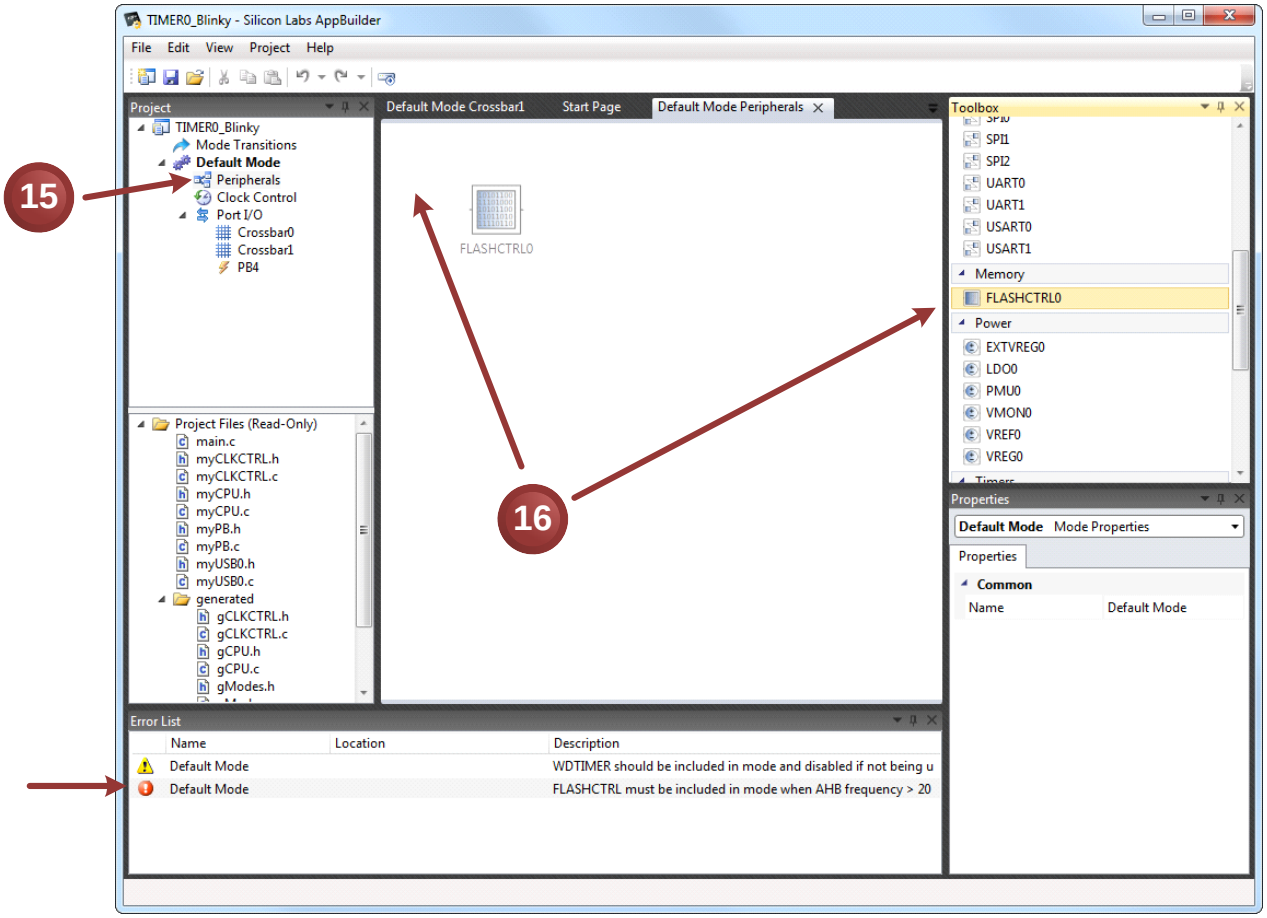
11. Double-click on **crossbar1**.
12. Enable **crossbar1**.



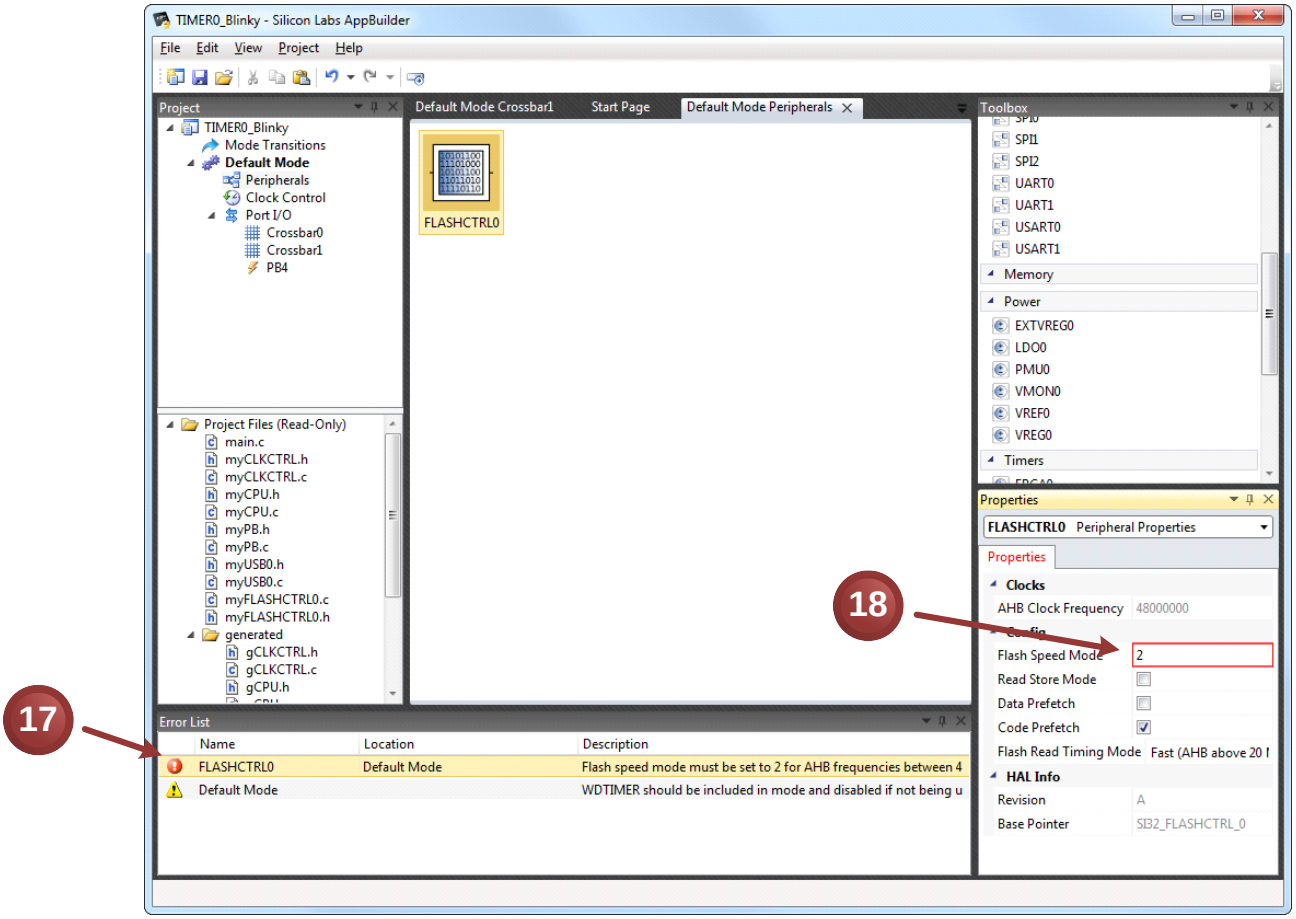
13. Click on the PB2.10 column.
14. Set PB2.10 to Push-Pull in the **Properties** window.



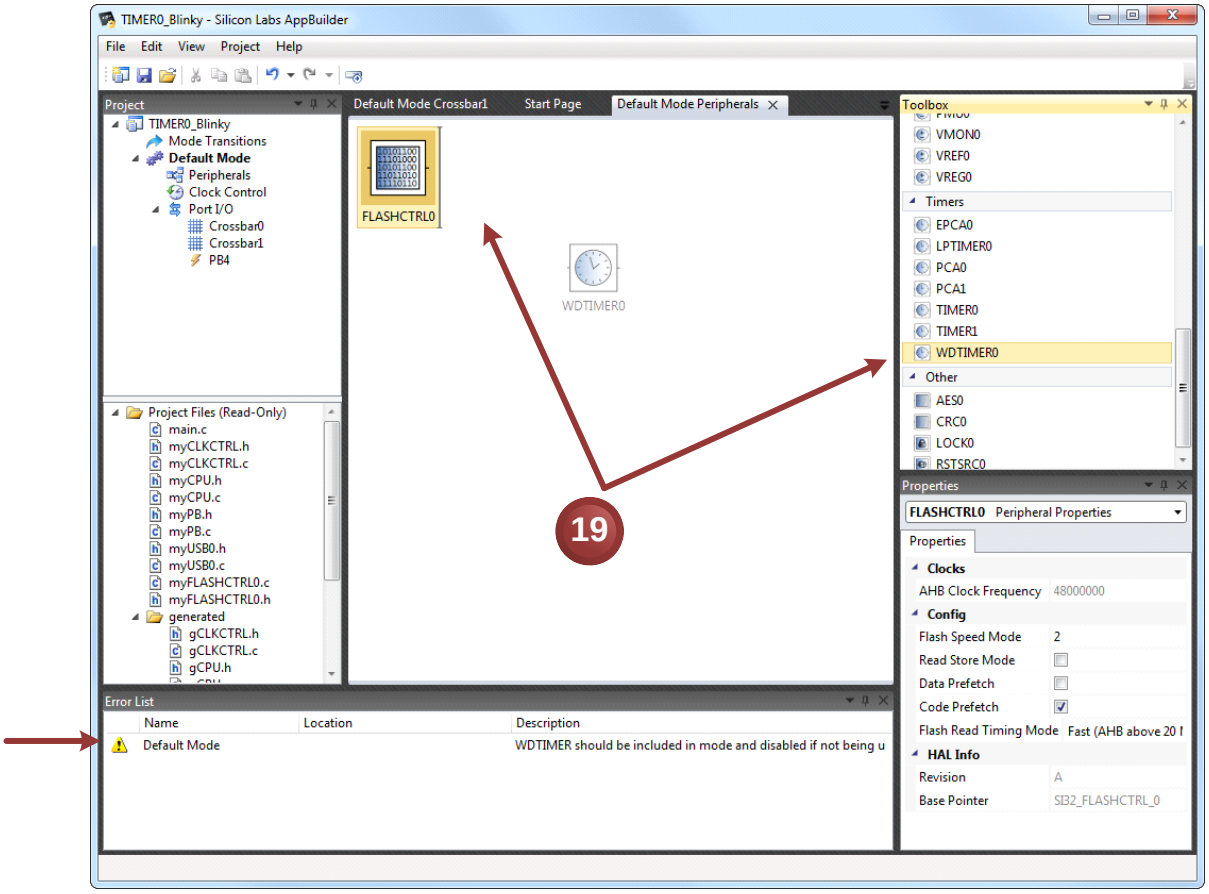
15. Let's fix the **FLASHCTRL0** error in the **Error List**. Double-click on **Peripherals**.
16. Drag and drop **FLASHCTRL0** to the **Peripherals** Canvas.



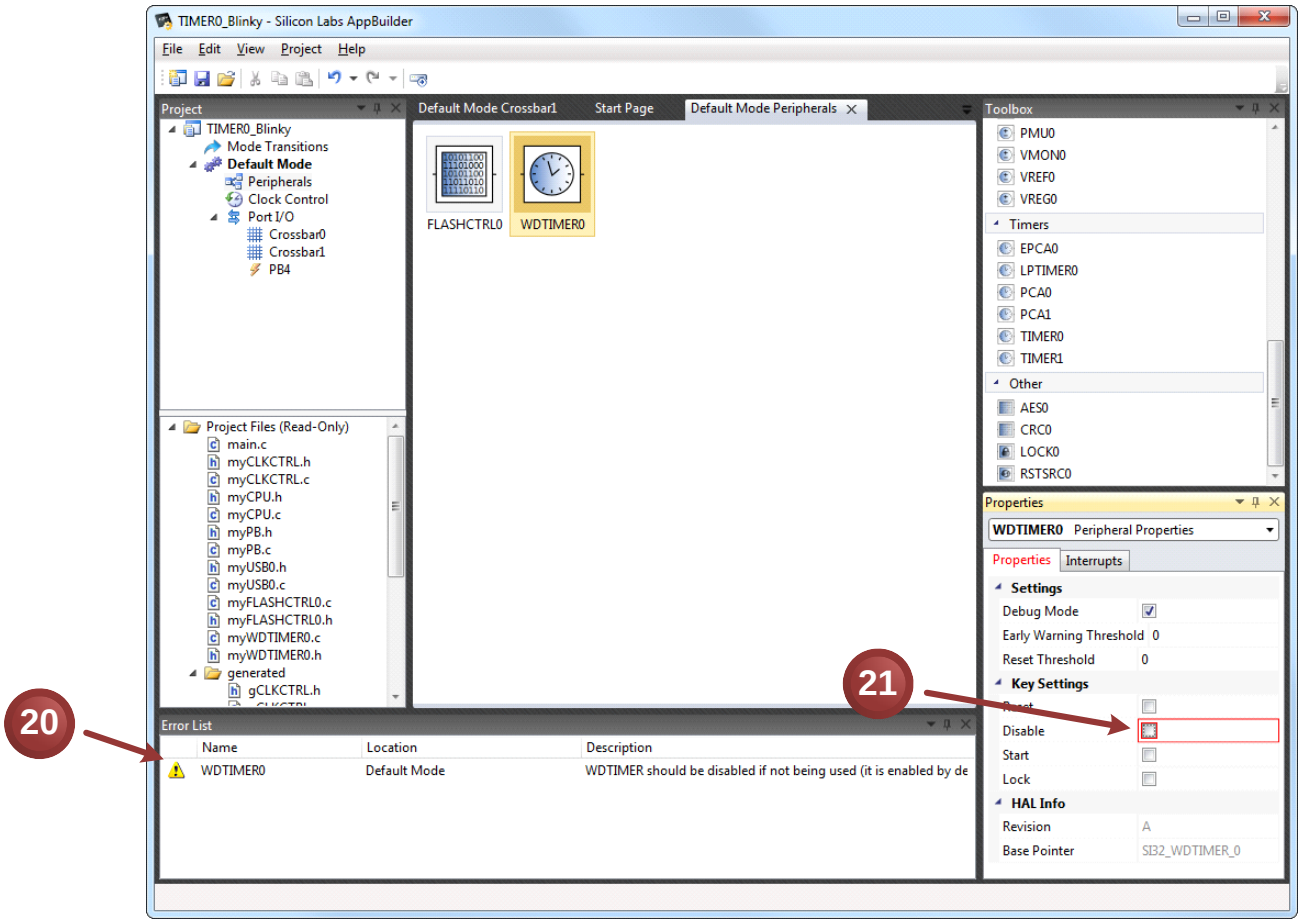
17. Double-click on the error. The program will automatically jump to the **FLASHCTRL0 Properties** window.
18. Set the **Flash Speed Mode** field to **2**. The error will disappear after hitting **Enter**.



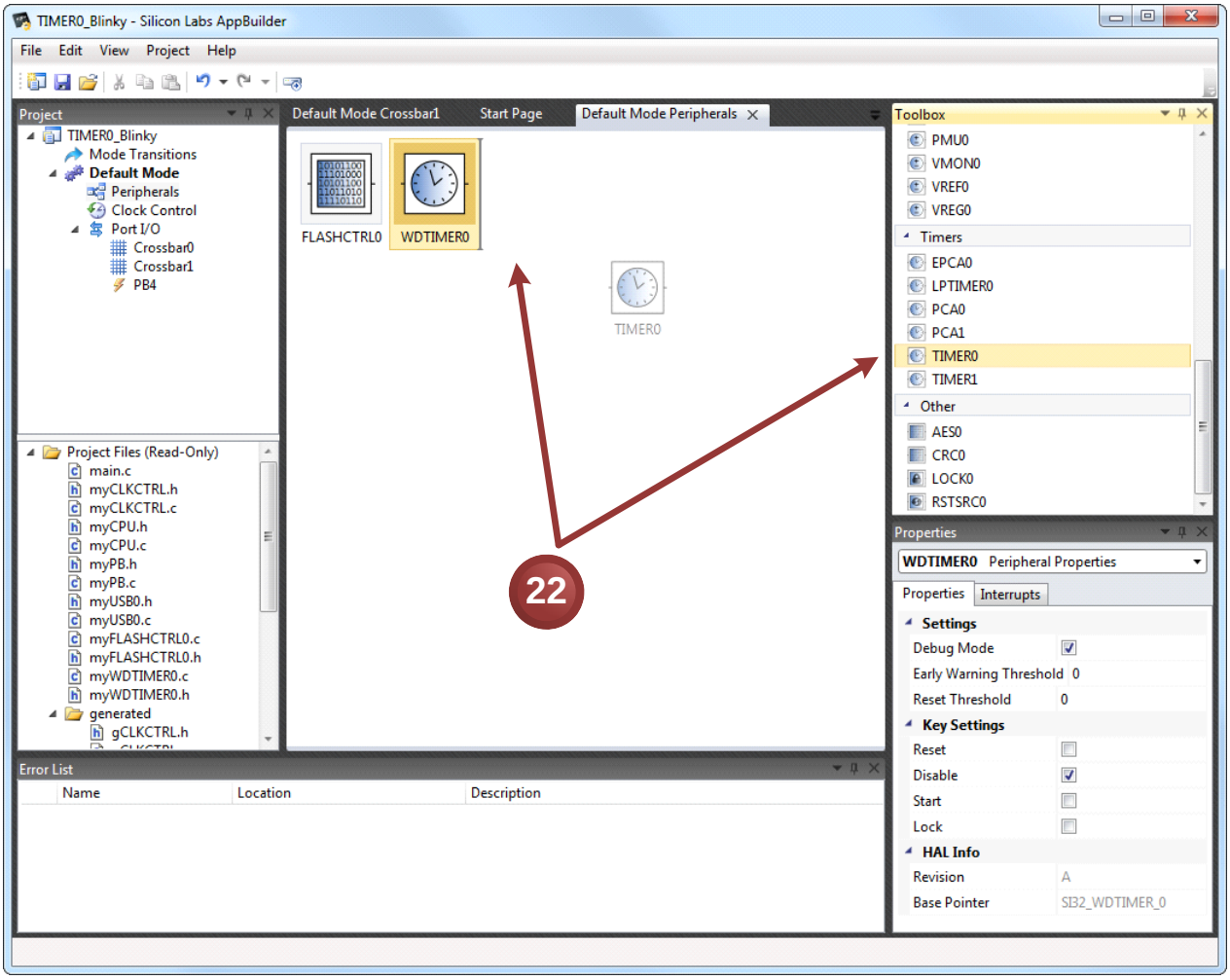
19. Let's fix the last error in the Error List for the Watchdog Timer. Drag and drop **WDTIMER0** to the **Peripherals** Canvas.



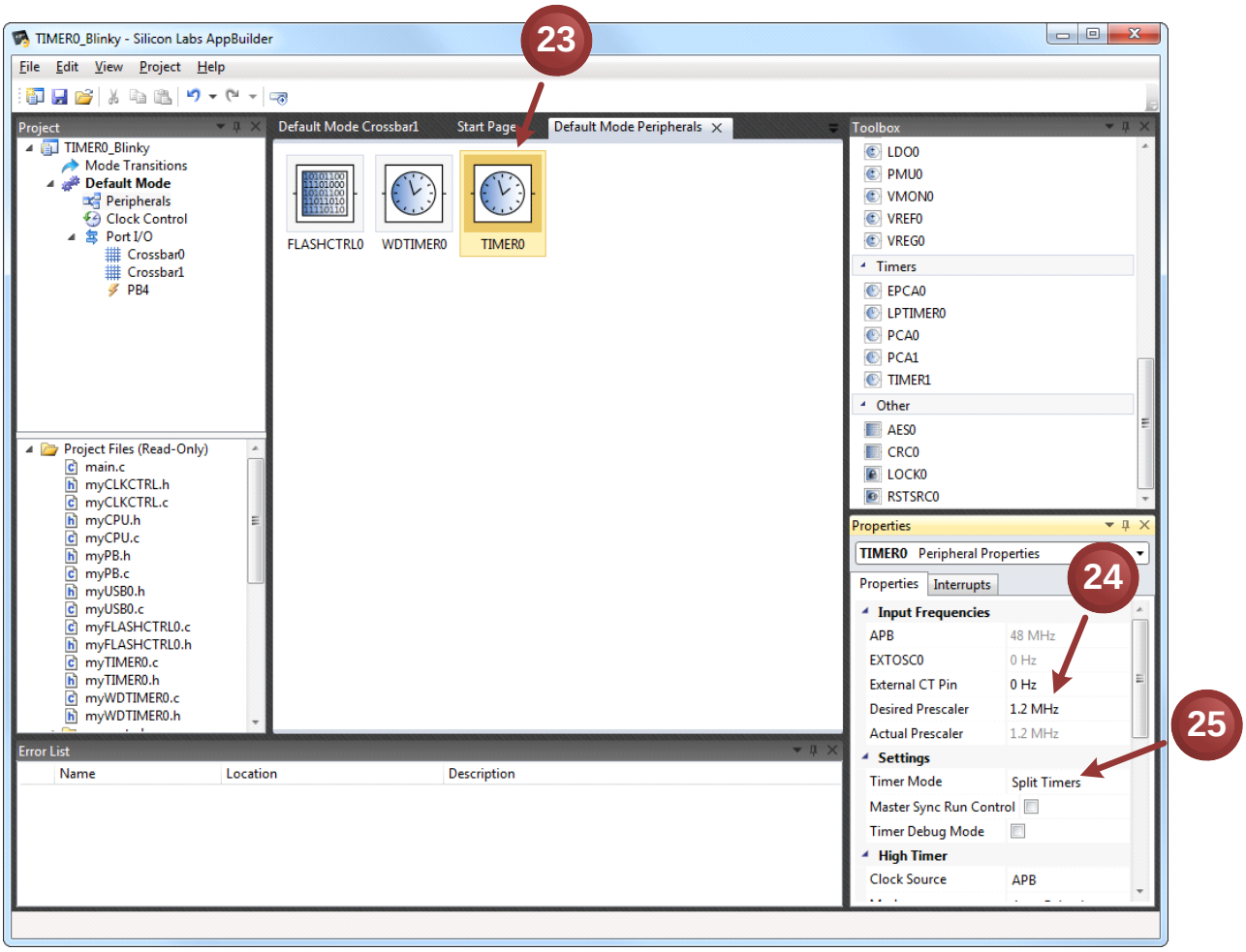
20. Double-click on the error. The program will automatically jump to the **WDTIMER0 Properties** window.
21. Disable the Watchdog Timer. The **Error List** should now be empty.



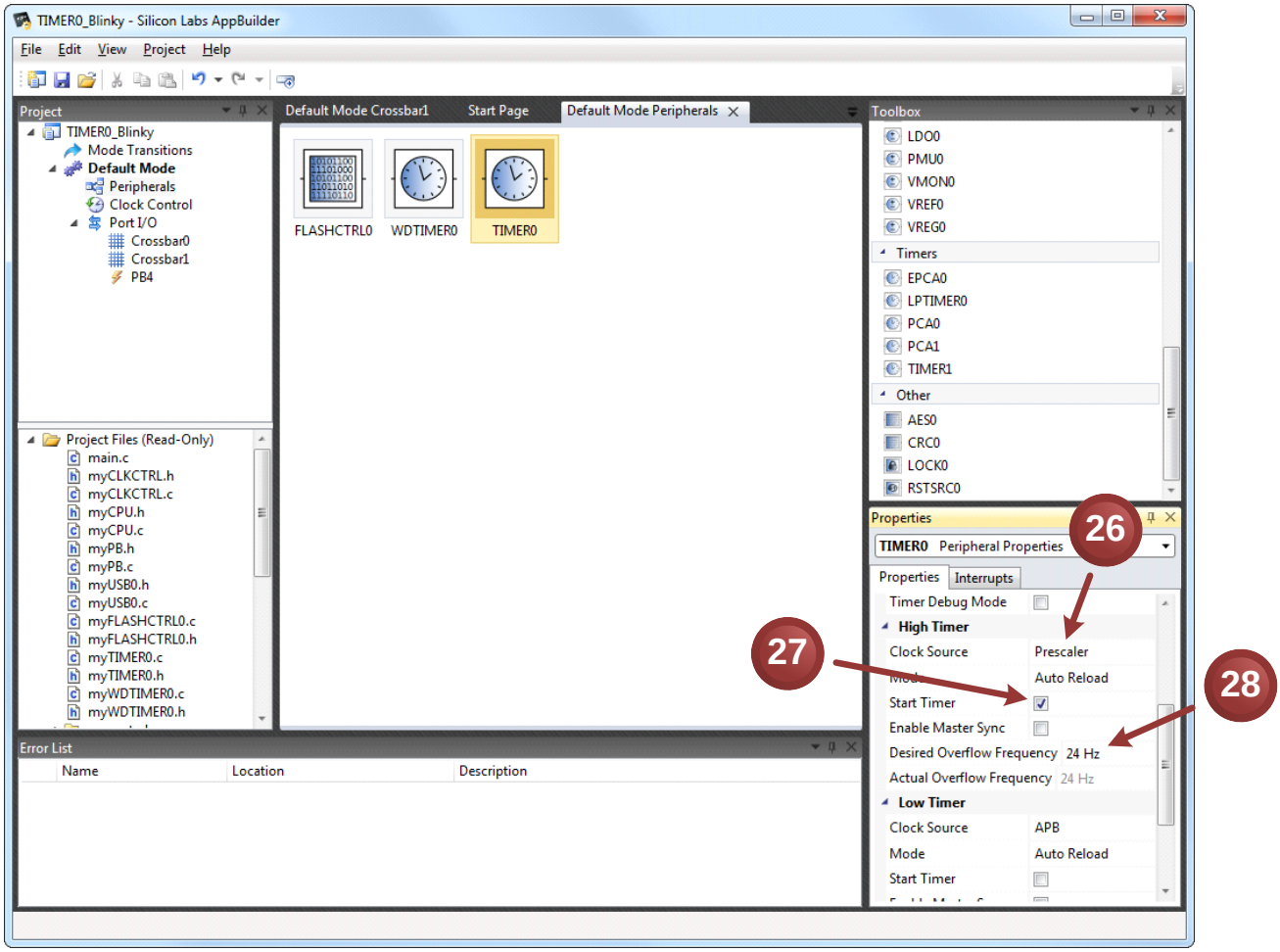
22. Drag and drop **TIMER0** to the **Peripherals** Canvas.



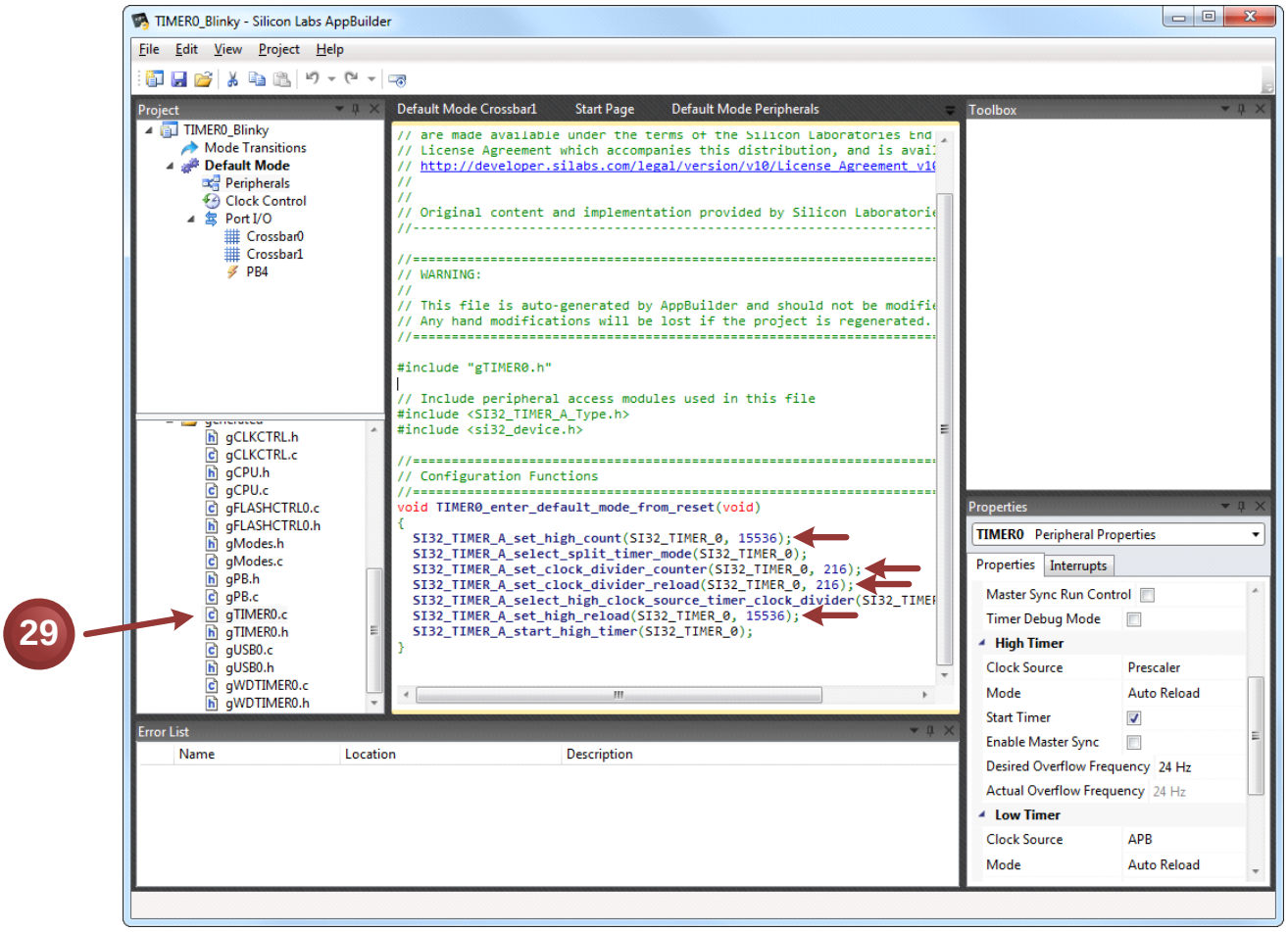
23. Click on TIMER0.
24. Set the **Desired Prescaler** value to **1.2 MHz.**
25. Select **Split Timers** for the **Timer Mode**.



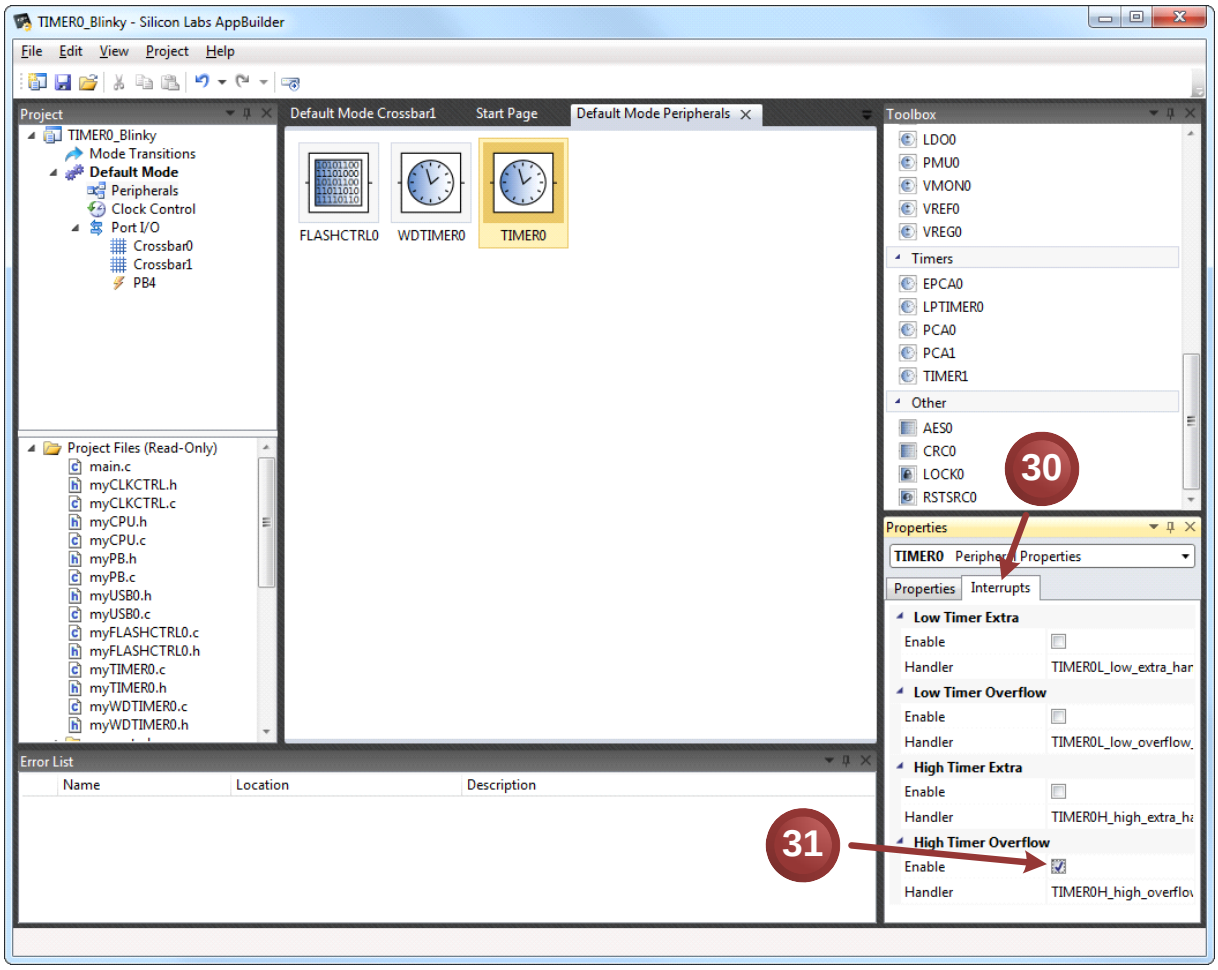
26. Select the **Prescaler** as the TIMER0 high clock source.
27. Start the high timer.
28. Set the **Desired Overflow Frequency** to **24 Hz** (12 Hz x 2 for toggling the LED on and off). TIMER0 high will be in auto-reload mode by default.



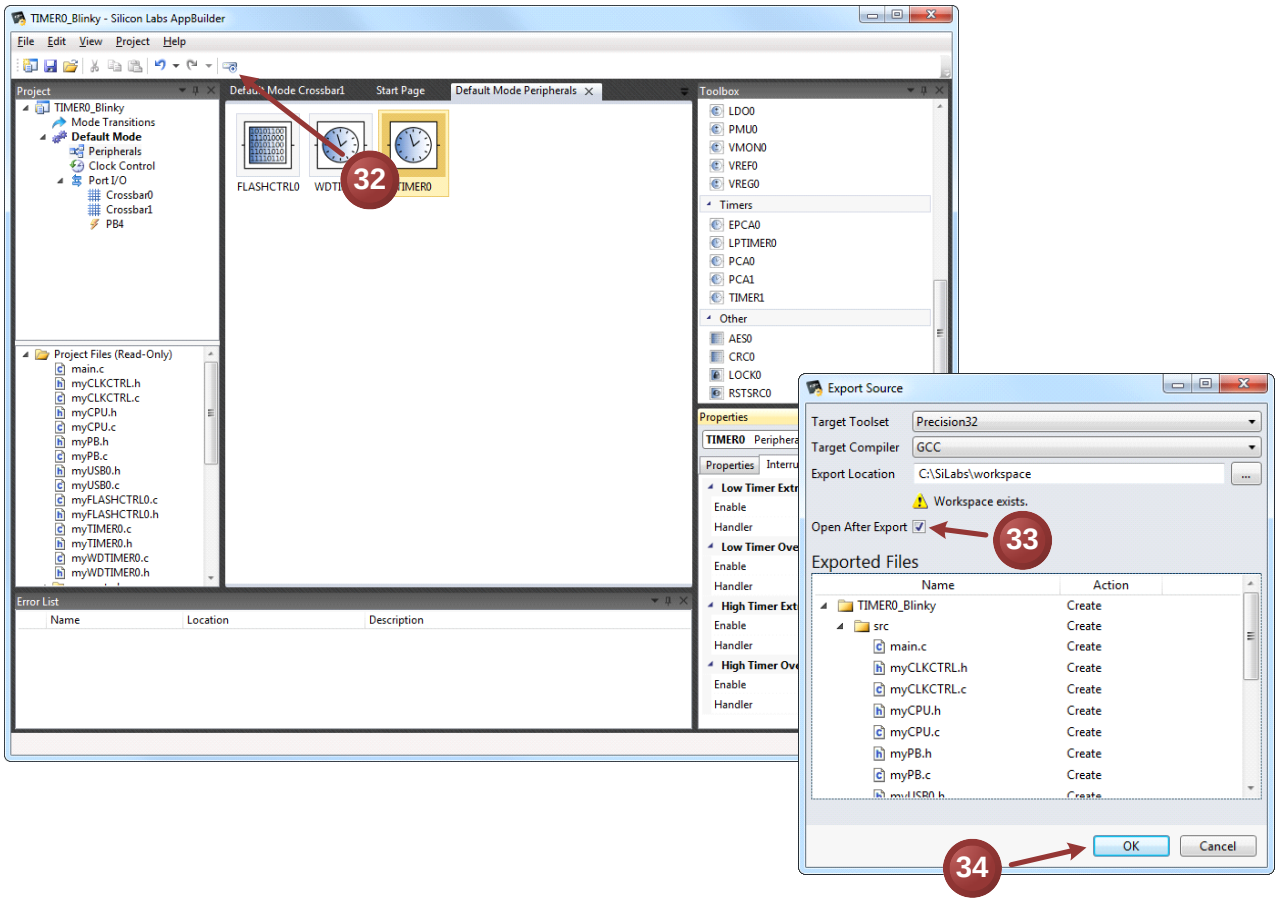
29. We can view the generated code by clicking on **gTIMER0.c** in the source files list. AppBuilder used the same values we calculated at the beginning of the lab for the prescaler (**216**) and reload values (**15536**).

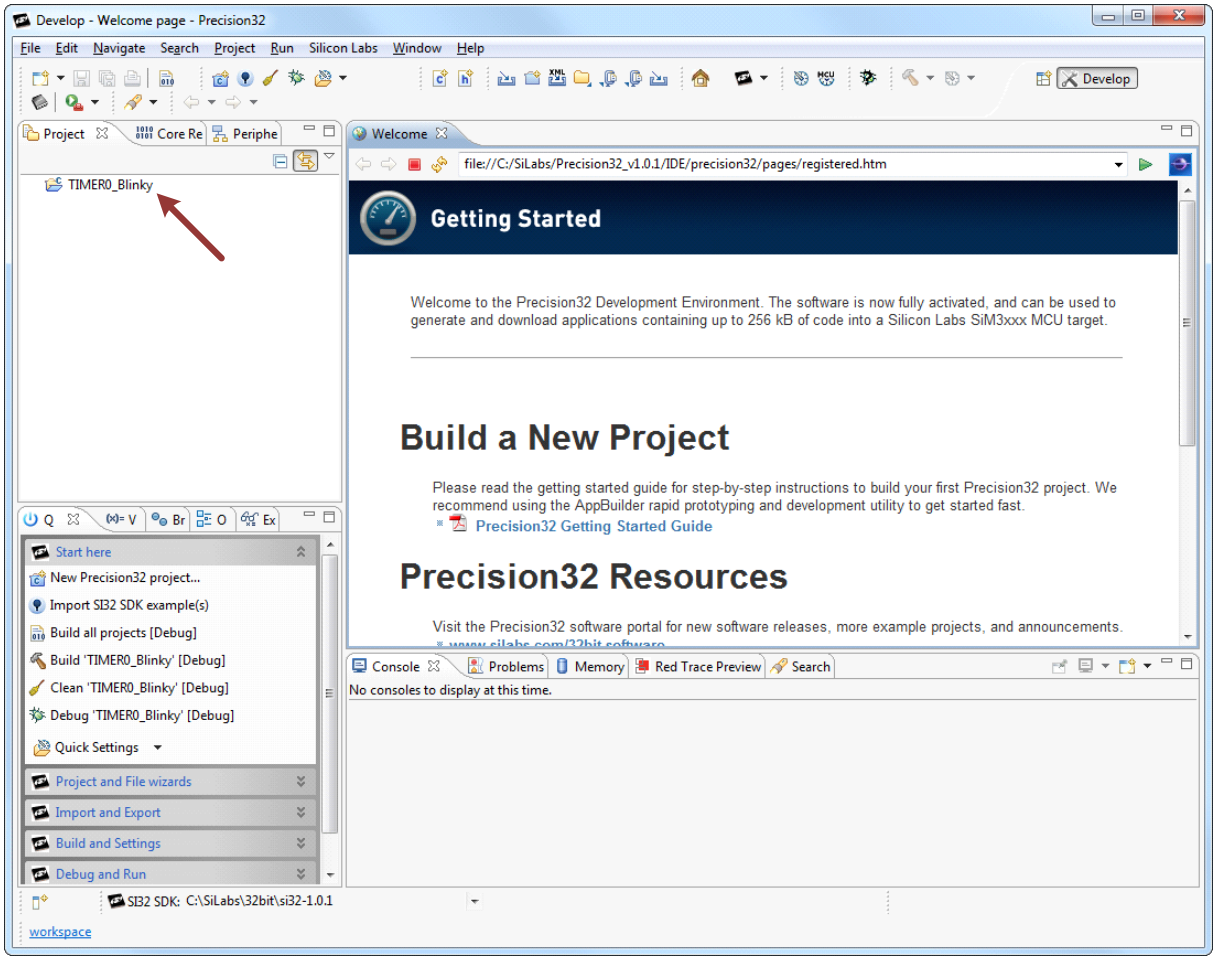


30. Click the TIMER0 **Interrupts** tab.
31. Enable **High Timer Overflow** interrupts.



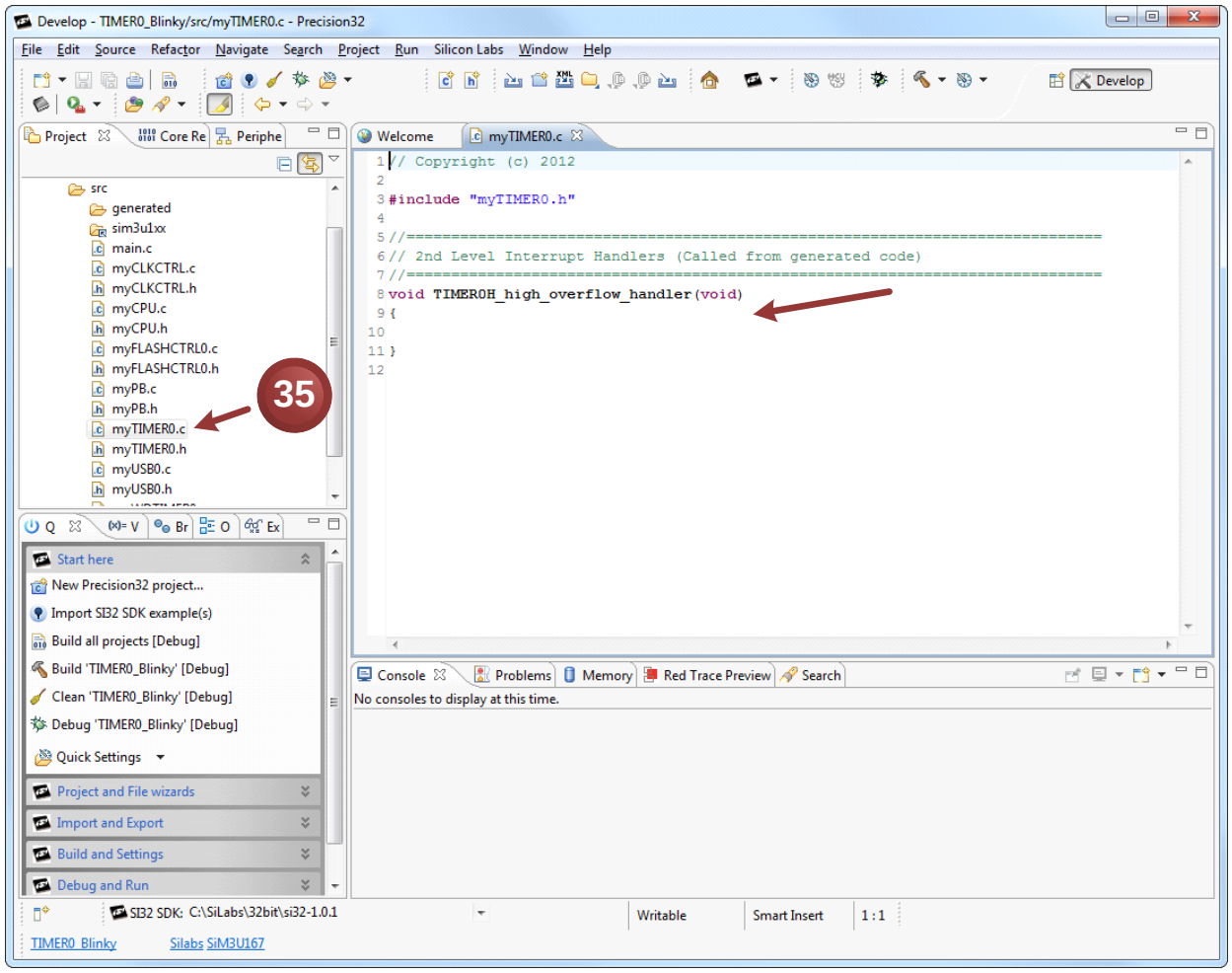
32. Click the **Export** button.
33. Verify **Open After Export** is automatically selected.
34. Press **OK**. The IDE will automatically open and have the AppBuilder project added to the workspace.




Develop - Welcome page - Precision32
File Edit Navigate Search Project Run Silicon Labs Window Help
Develop
Project
Core Re
Periphe
TIMER0_Blinky
Welcome
file://C:/SiLabs/Precision32_v1.0.1/IDE/precision32/pages/registered.htm
Getting Started
Welcome to the Precision32 Development Environment. The software is now fully activated, and can be used to generate and download applications containing up to 256 kB of code into a Silicon Labs SiM3xxx MCU target.
Build a New Project
Please read the getting started guide for step-by-step instructions to build your first Precision32 project. We recommend using the AppBuilder rapid prototyping and development utility to get started fast.
Precision32 Getting Started Guide
Precision32 Resources
Visit the Precision32 software portal for new software releases, more example projects, and announcements.
Start here
New Precision32 project...
Import SI32 SDK example(s)
Build all projects [Debug]
Build 'TIMER0_Blinky' [Debug]
Clean 'TIMER0_Blinky' [Debug]
Debug 'TIMER0_Blinky' [Debug]
Quick Settings
Project and File wizards
Import and Export
Build and Settings
Debug and Run
Console
Problems
Memory
Red Trace Preview
Search
No consoles to display at this time.
SI32 SDK: C:\SiLabs\32bit\si32-1.0.1
workspace

35. In the IDE, double-click on **myTIMER0.c**. The TIMER0H high overflow interrupt handler will be empty

.



The TIMER0 high overflow handler needs to perform two functions: 1) clear the TIMER0 high overflow interrupt flag in the TIMER0 module, and 2) toggle the PB2.10 pin.

36. Add the includes to the top of the **myTIMER0.c** file:

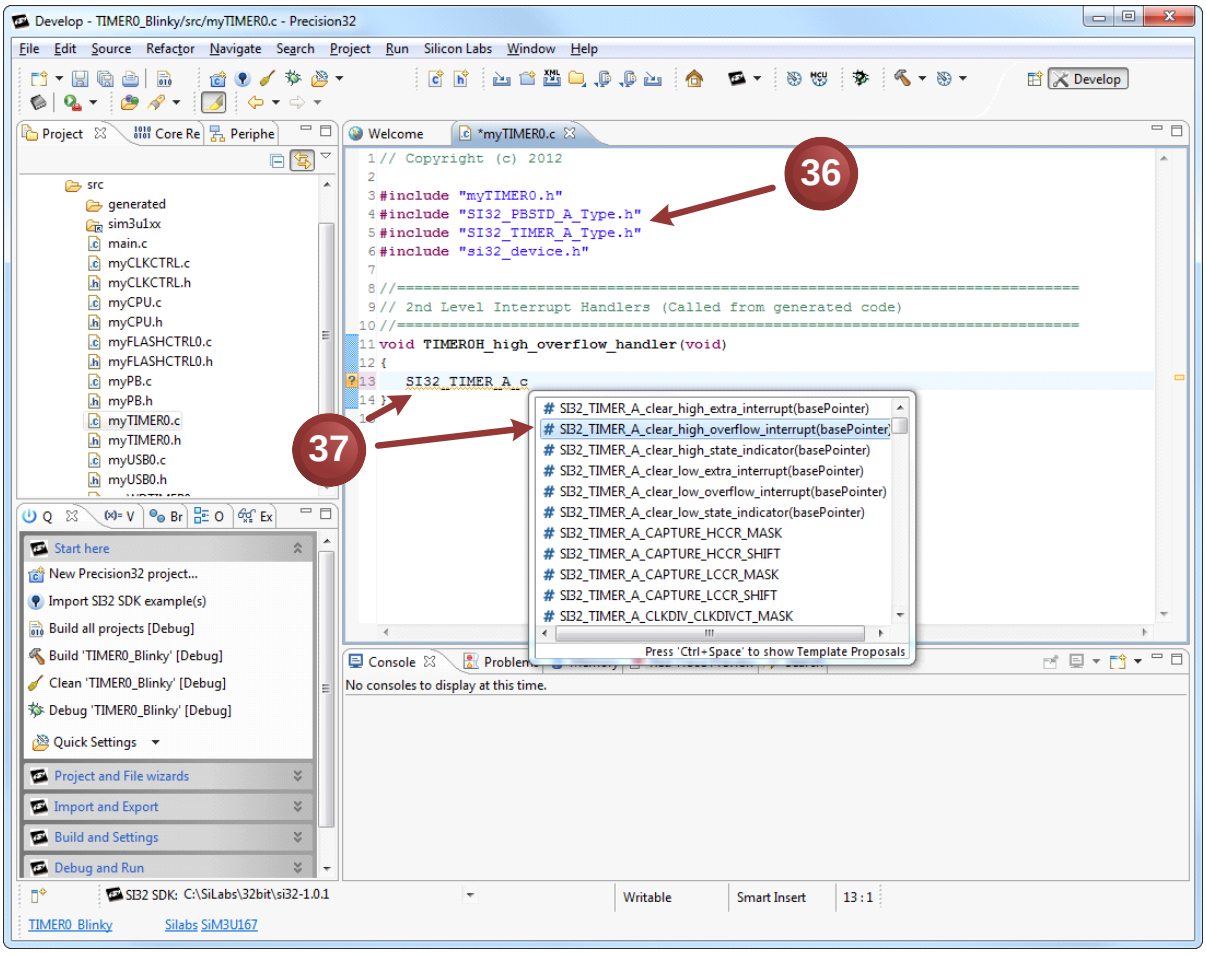
```
#include <SI32_PBSTD_A_Type.h>

#include <SI32_TIMER_A_Type.h>

#include <si32_device.h>
```

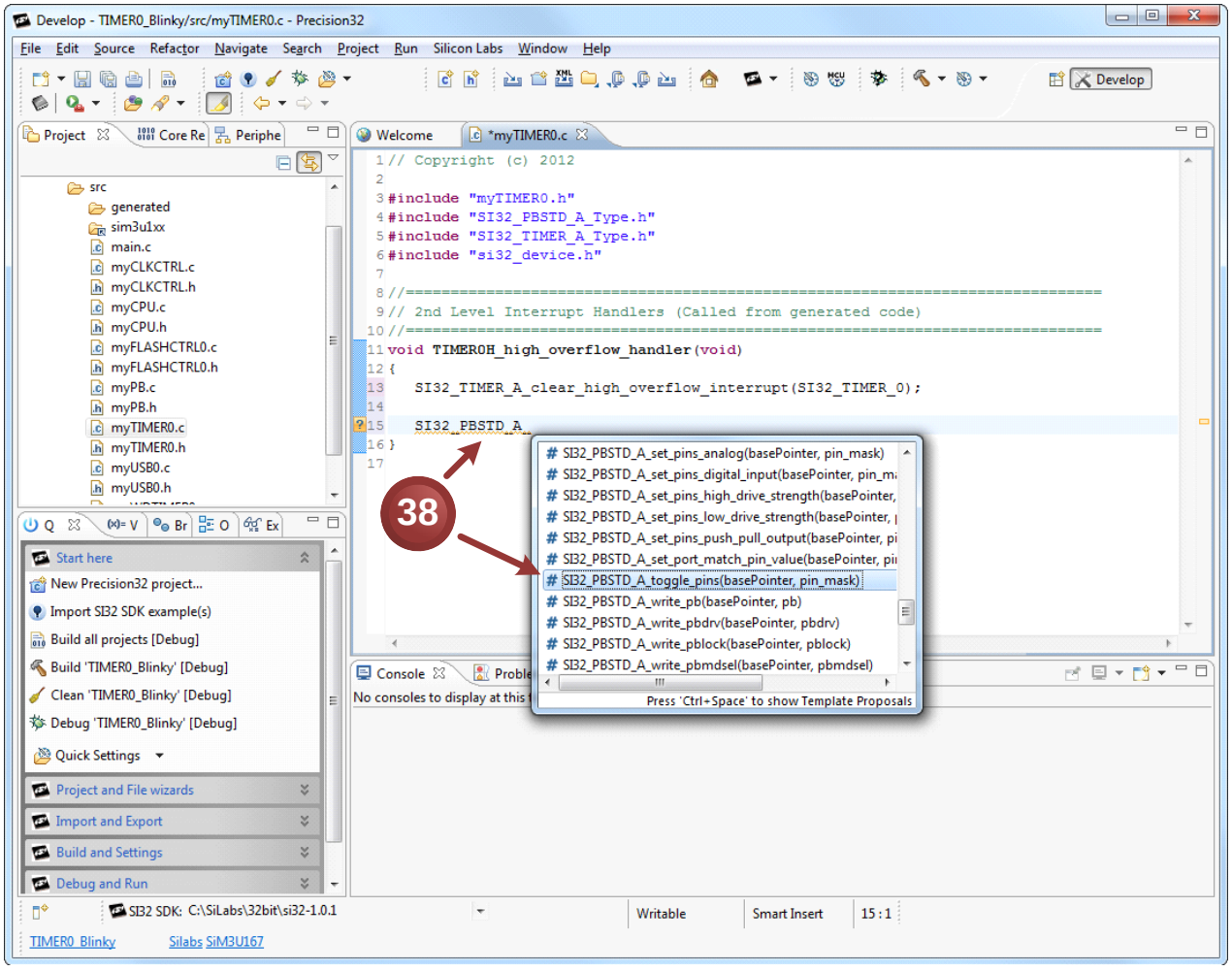
37. Because we already added the header files, we can use the auto-complete feature of the IDE to help us find the correct functions. Move the cursor inside the interrupt handler, type in "**SI32_TIMER_A_c**", and press **CTRL** + **SPACEBAR**. This will open a window that shows all the available defines and functions that start with SI32_TIMER_A _c. The "**clear_high_overflow_interrupt**" function will be the second choice. Once the correct function is selected, fill in the correct base pointer for TIMER0 (**SI32_TIMER_0**). The line of code should look like:

```
SI32_TIMER_A_clear_high_overflow_interrupt(SI32_TIMER_0);
```

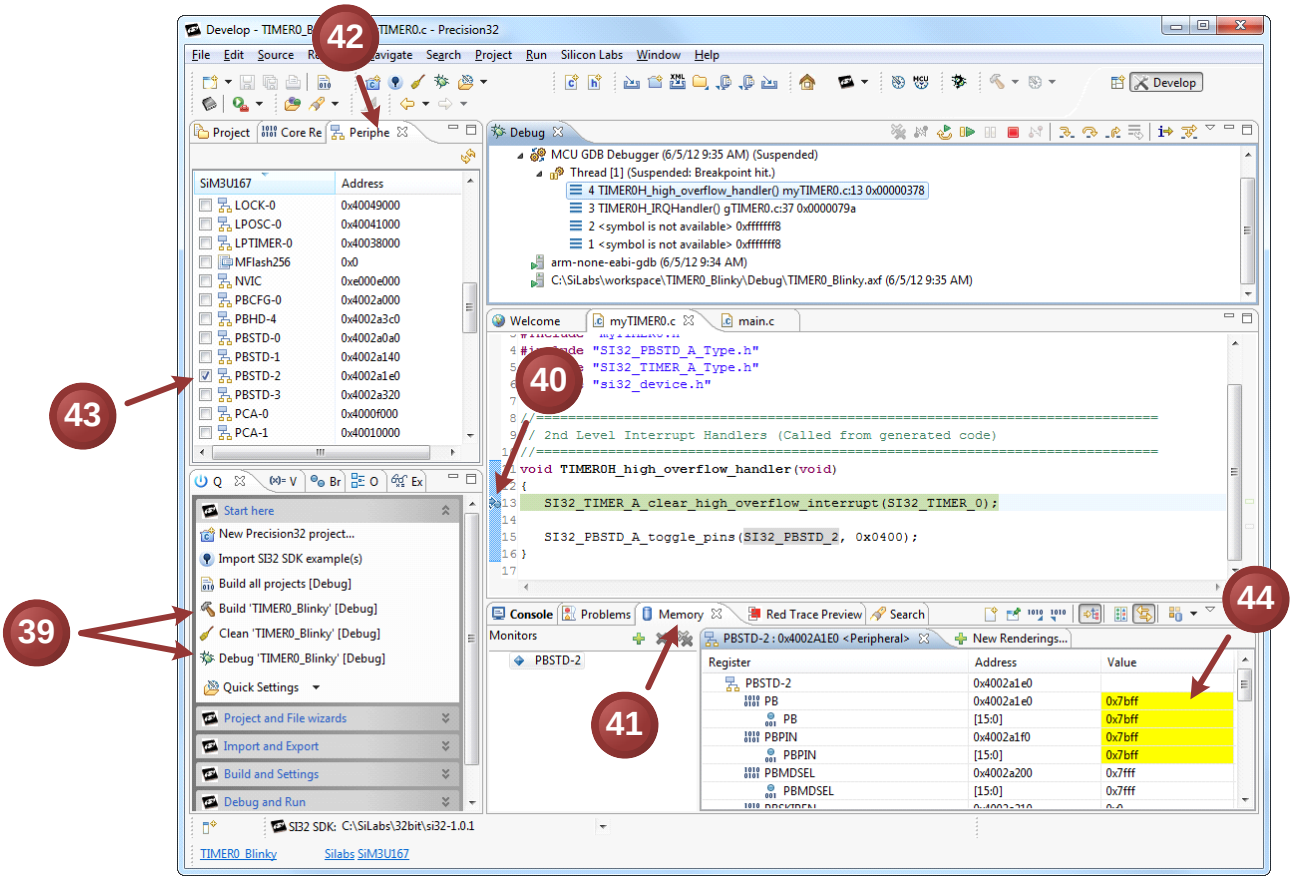


38. We can use the same process to enter in the second function, but we'll search for it in the full list this time. Move to a new line, type in "**SI32_PBSTD_A**", and press **CTRL** + **SPACEBAR**. The list will show all functions and defines for the PBSTD A module. We know we want a function that will toggle the pin, so look for a function of the appropriate name (**toggle_pins**). Once the correct function is selected, fill in the correct base pointer for PB2 (**SI32_PBSTD_2**) and the correct mask for PB2.10 (**0x0400**). The line of code should look like:

```
SI32_PBSTD_A_toggle_pins(SI32_PBSTD_2, 0x0400);
```



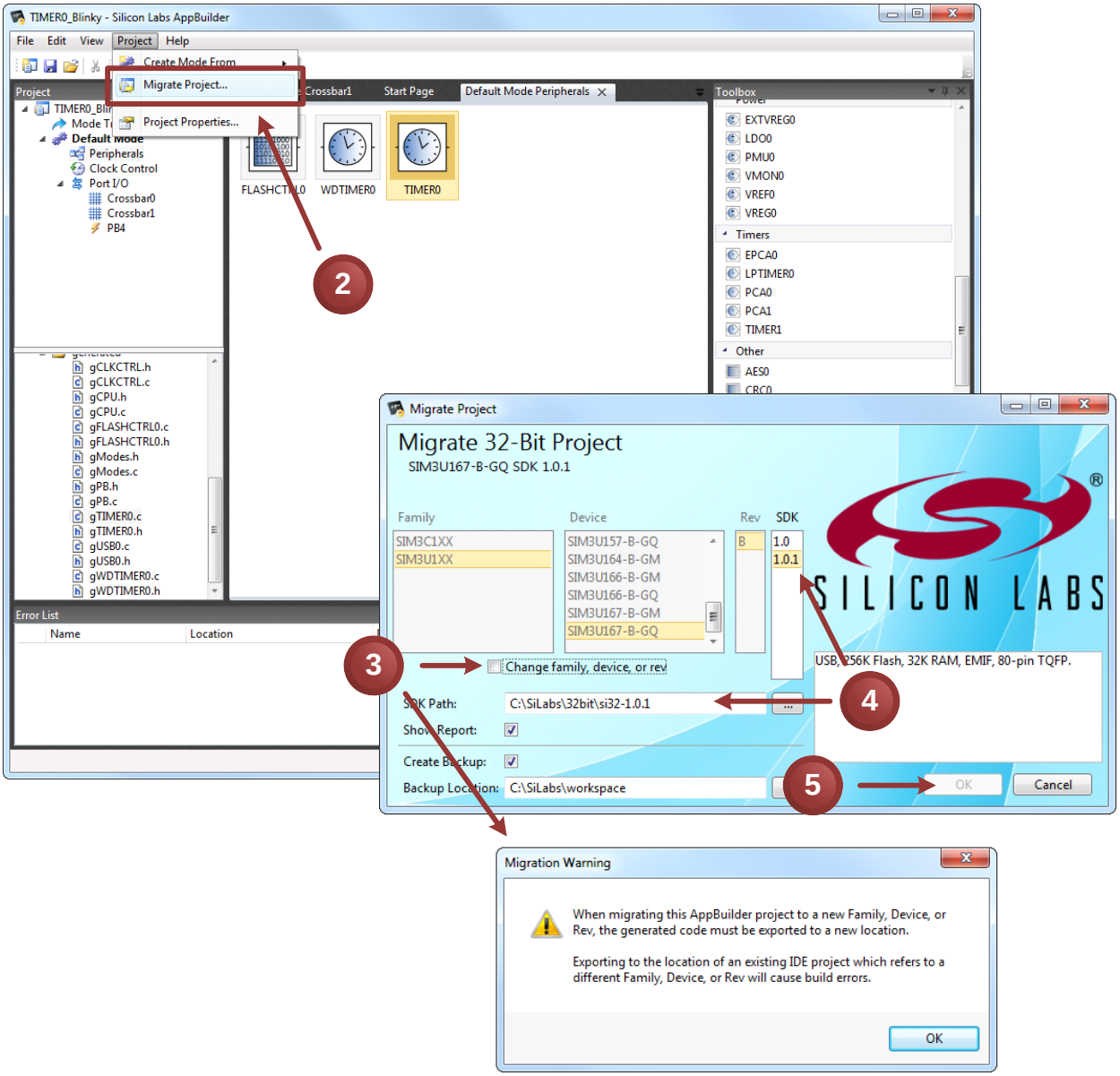
39. Build the code and start a debug session. Then start the program. The LED will blink at a 12 Hz rate.
40. Add a breakpoint in the TIMER0 handler in **myTIMER0.c**.
41. Select the **Memory** view.
42. Select the **Peripheral** view.
43. Open the PB2 register window by selecting **PBSTD-2** in the **Peripheral** view.
44. Pressing the start program button repeatedly will update the PB2 pin value in the register window.



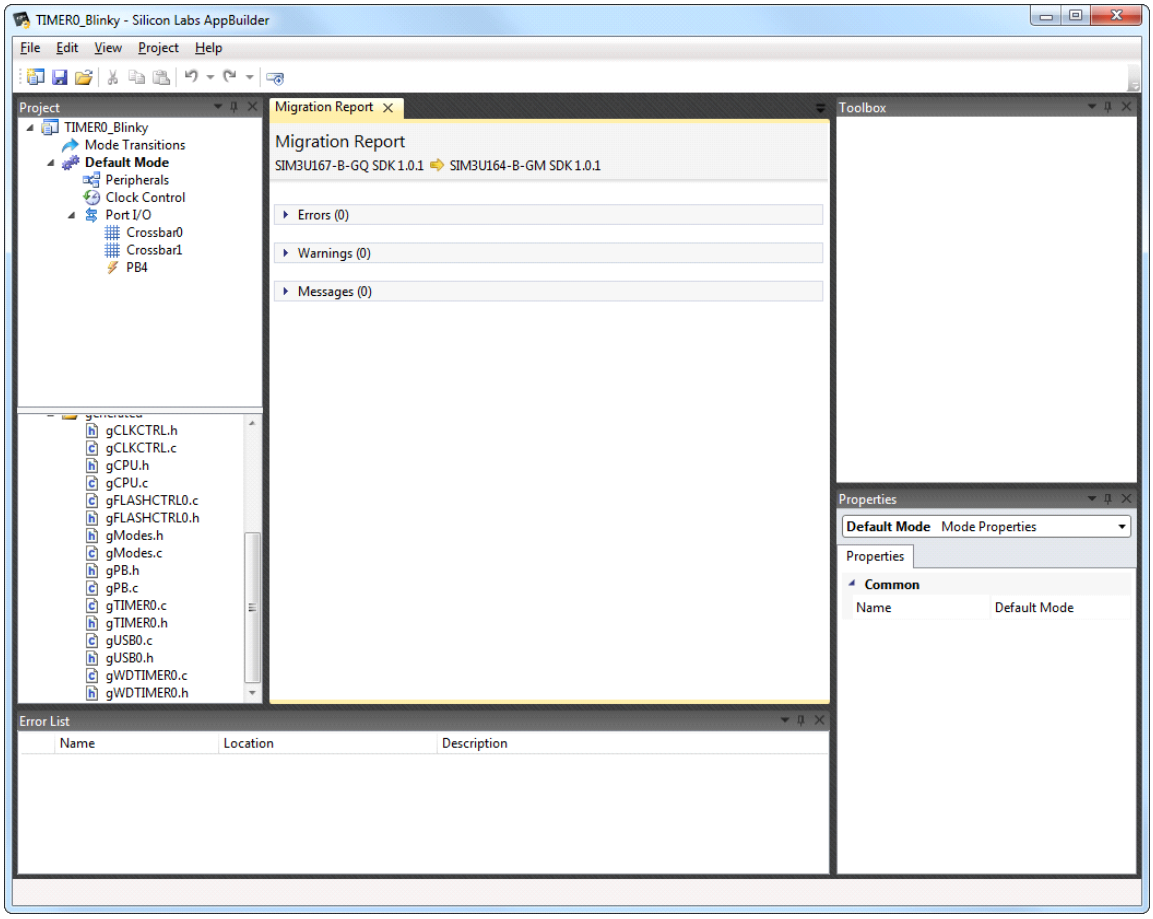
6.4. Additional Notes: Migrating a Project

The AppBuilder migrate feature allows a project switch to a different version of the SDK, a different device family, or device within the same family. To migrate a project:

1. Open the project in **AppBuilder 1.0.1**.
2. Go to the **Project→Migrate Project...** menu selection.
3. Check the **Change family, device, or rev** checkbox to change the device family, device within the selected family, or the revision of the device. This will prompt a message window that requests a new export location.
4. Changes to the SDK version do not require a new export location.
5. Make any other desired modifications to the Migrate Project dialog and press **OK**.



If the report option was not disabled, AppBuilder will generate a report of the migration operation. This report will output a summary of the migration operation (from→to) and any potential Errors, Warnings, or Messages.



NOTES:

Contact Information

Silicon Laboratories Inc.

400 West Cesar Chavez
Austin, TX 78701
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032

Please visit the Silicon Labs Technical Support web page:
https://www.silabs.com/support/pages/contacttechnicalsupport.aspx
and register to submit a technical support request.