# Detailed Walkthrough of the Precision32™ AppBuilder Power Estimator Tool

## 1. Introduction

This document provides a step-by-step tutorial walkthrough for the Precision32 AppBuilder Power Estimator tool using the **Silicon Labs SDK version 1.1.1** and **AppBuilder v1.1.1**.

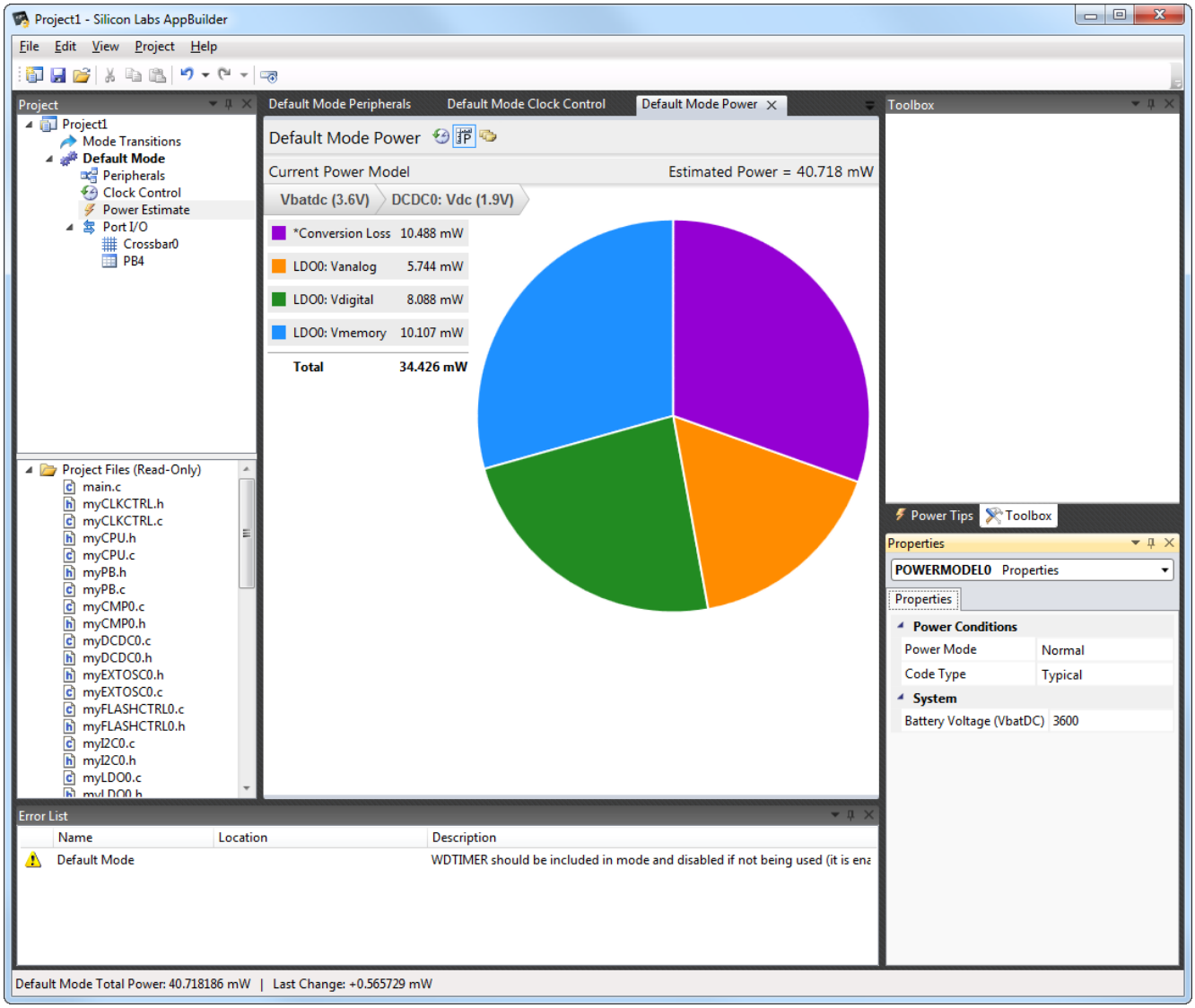



**Figure 1. AppBuilder Power Estimator Walkthrough Overview**

## 2. Relevant Documentation

Precision32 application notes are listed on the following web site: 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. Lab Software Setup

1. Install the Precision32 package from www.silabs.com/32bit-software.
2. Navigate to the v1.1.1 version of AppBuilder and open it.

## 4. Using the Power Estimator Tool

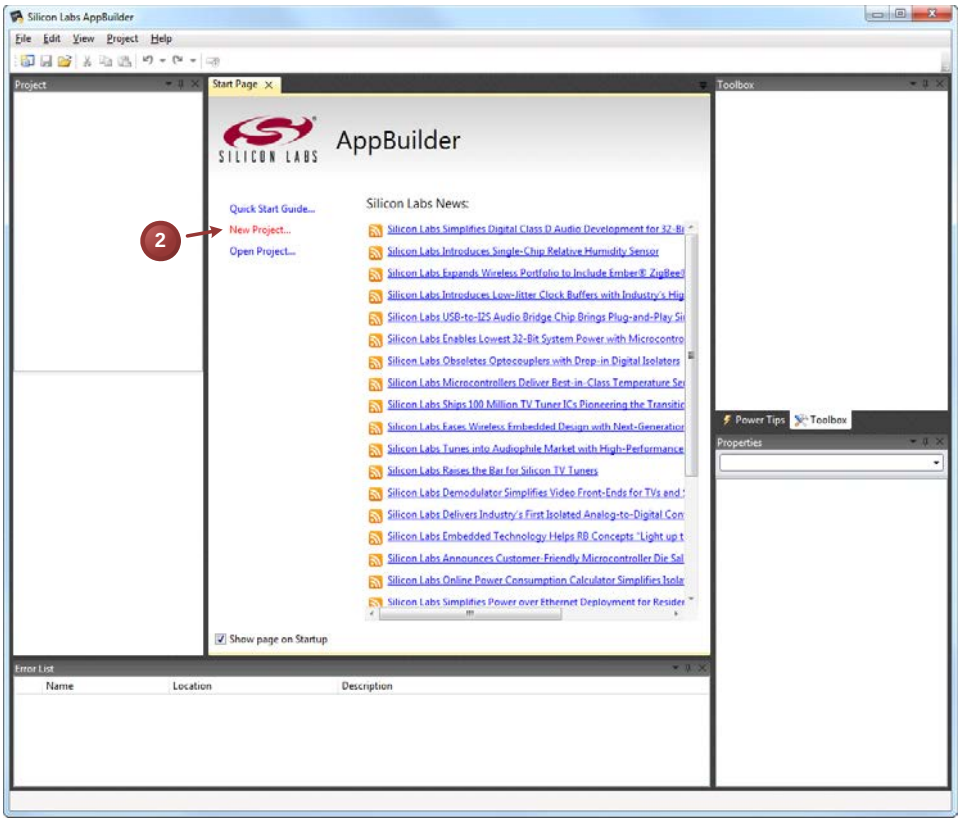
### 4.1. Objectives

The objectives are:

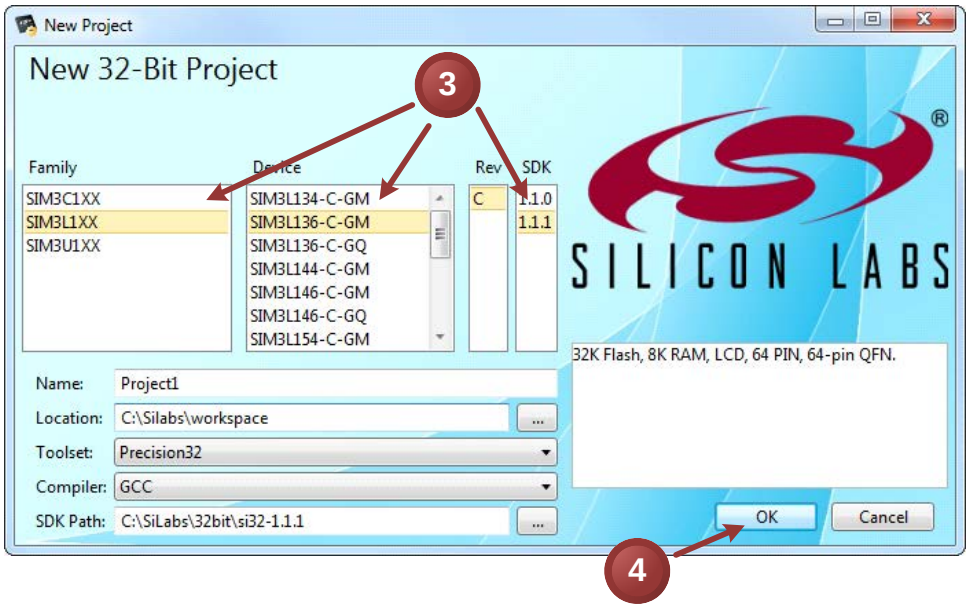
- Learn how to use the Power Estimator tool to estimate device power consumption.
- Explore some of the low-power features of the SiM3L1xx device family.
- (Optional) Measure and compare power consumption on hardware with the Power Estimator estimates.

### 4.2. Instructions

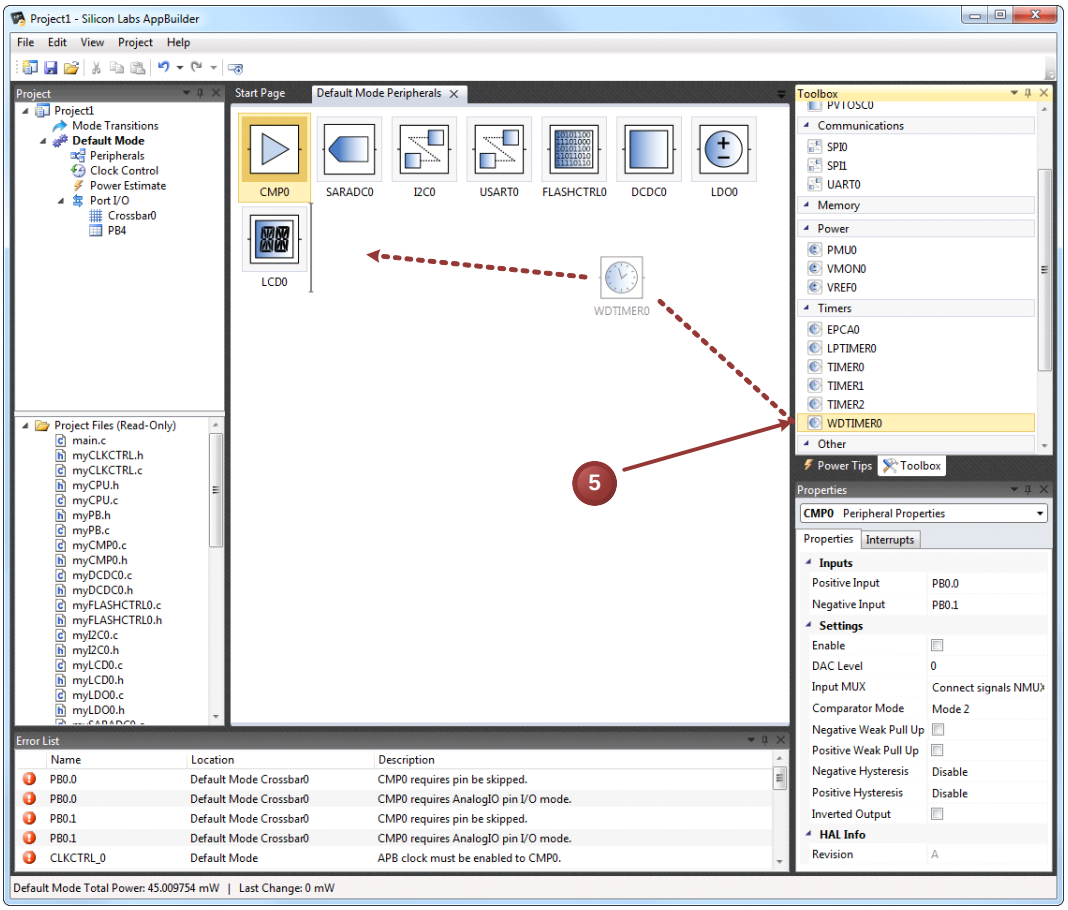
1. Launch the **Precision32 AppBuilder 1.1.1** (default path: C:\SiLabs\Precision32_v1.1.1\AppBuilder).
2. Click on **New Project** in the main startup window.



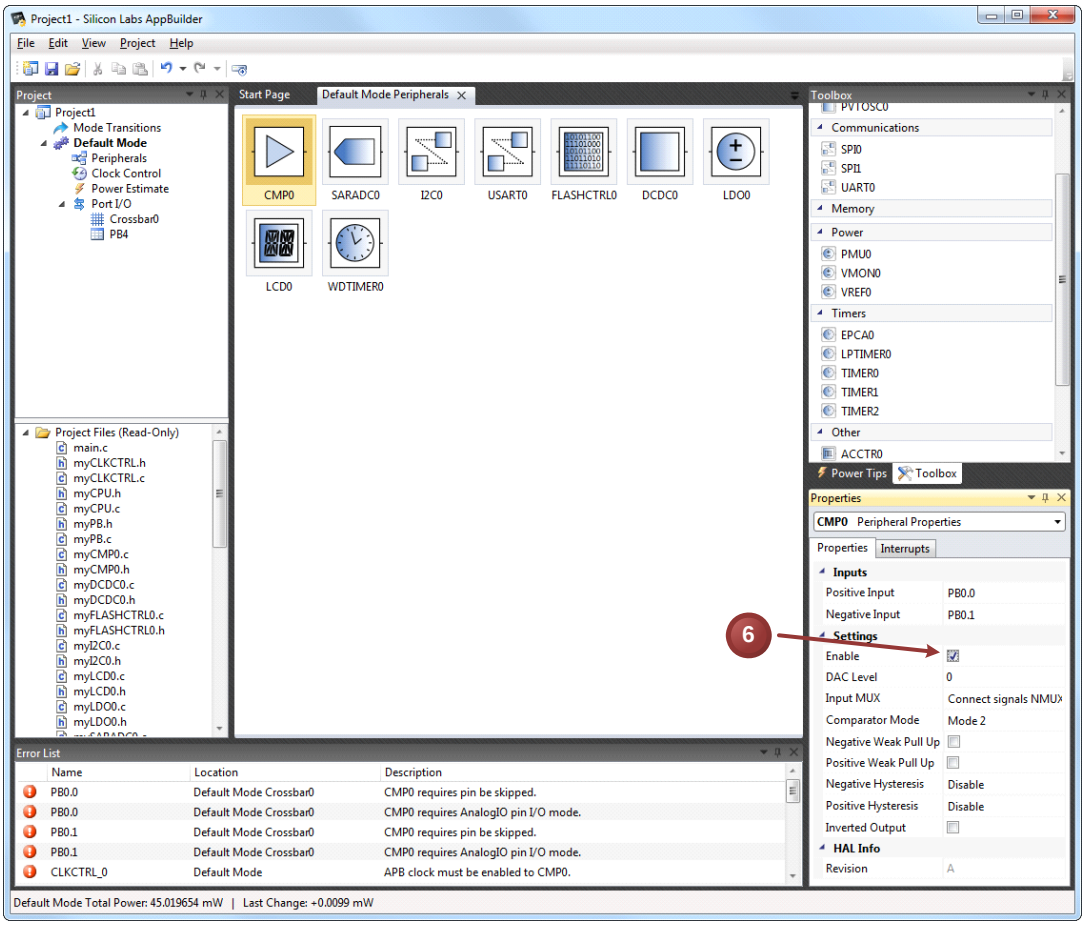
3. In the **New Project** dialog, select **SiM3L1xx**, **SiM3L136-B-GM**, and **SDK 1.1.1**.
4. Click **OK**.



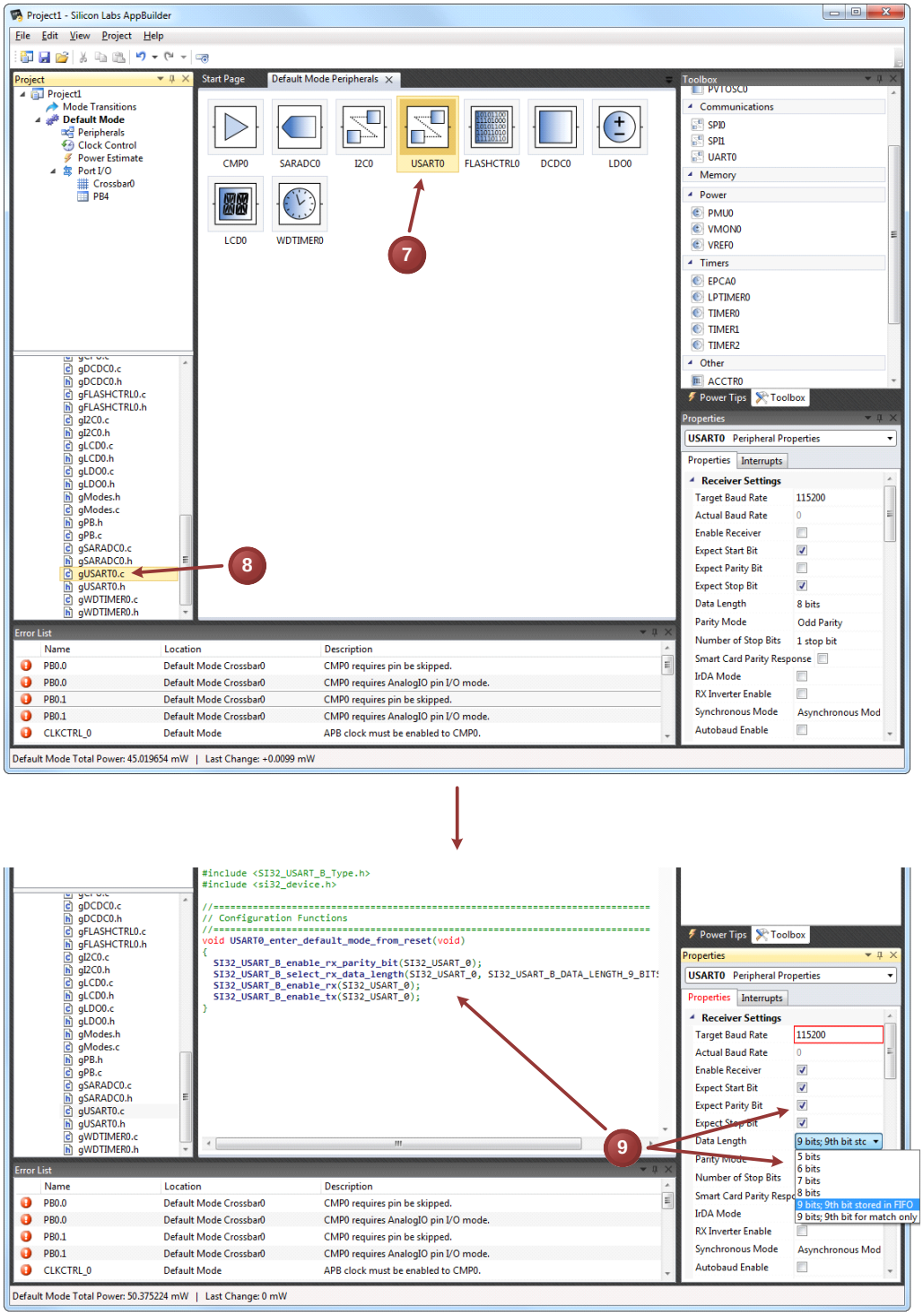
5. In the new project starting screen, add the following peripherals: **CMP0**, **SARADC0**, **I2C0**, **USART0**, **FLASHCTRL0**, **DCDC0**, **LDO0, LCD0**, and **WDTIMER0**. These can be added by double-clicking the peripheral in the **Toolbox** or by dragging and dropping the peripherals to the Peripherals canvas in the middle of the AppBuilder window.



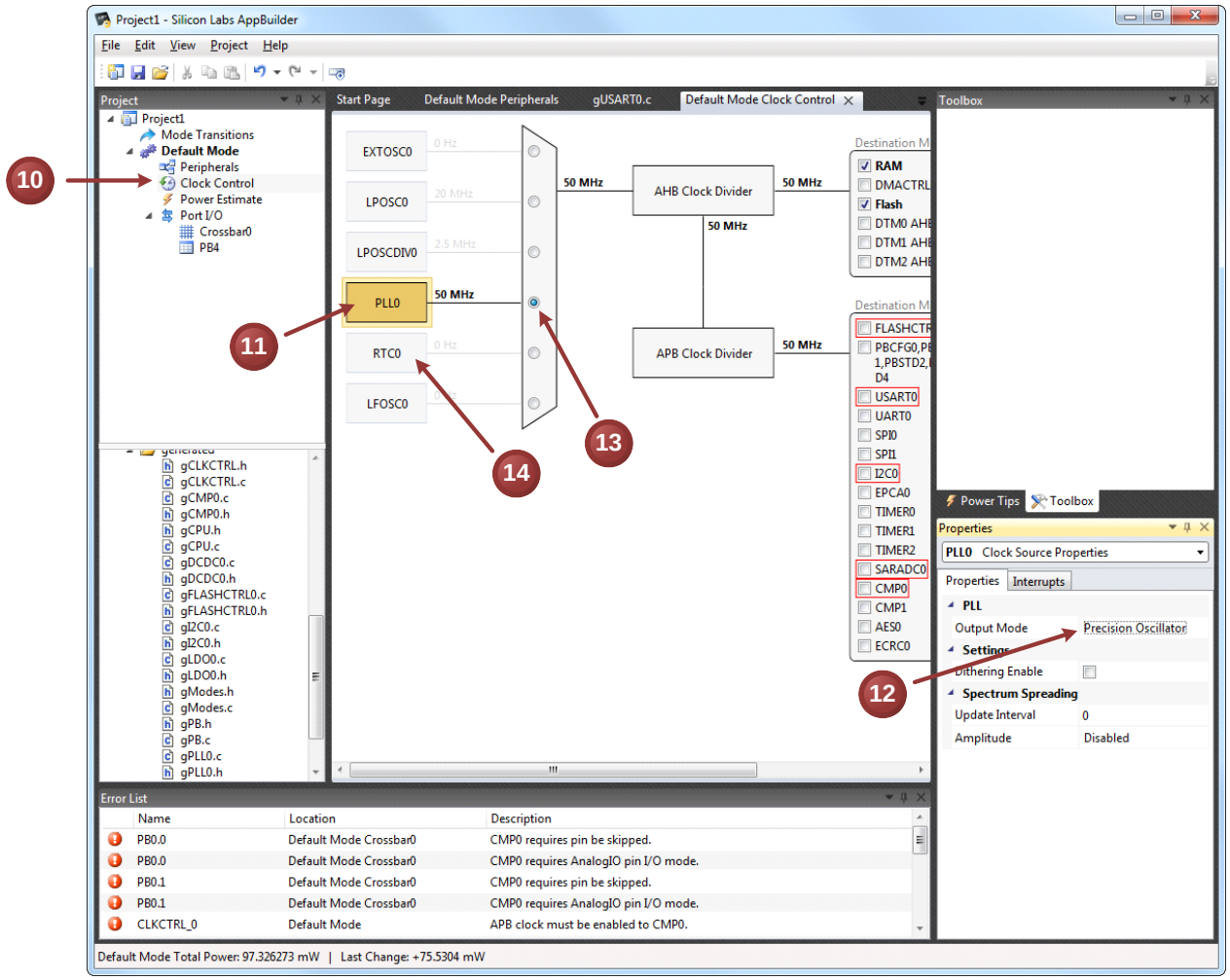
6. Click on each peripheral to select it and populate the properties in the **Properties** window. Enable each of the following peripherals:
   a. **CMP0**: Settings→Enable
   b. **SARADC0**: Measurement Control→ADC Enable
   c. **I2C0**: Common Settings→Enable I2C Module
   d. **USART0**: Receiver Settings→Enable Receiver and Transmitter Settings→Enable Transmitter
   e. **DCDC0**: Control→DC-DC Converter Enable



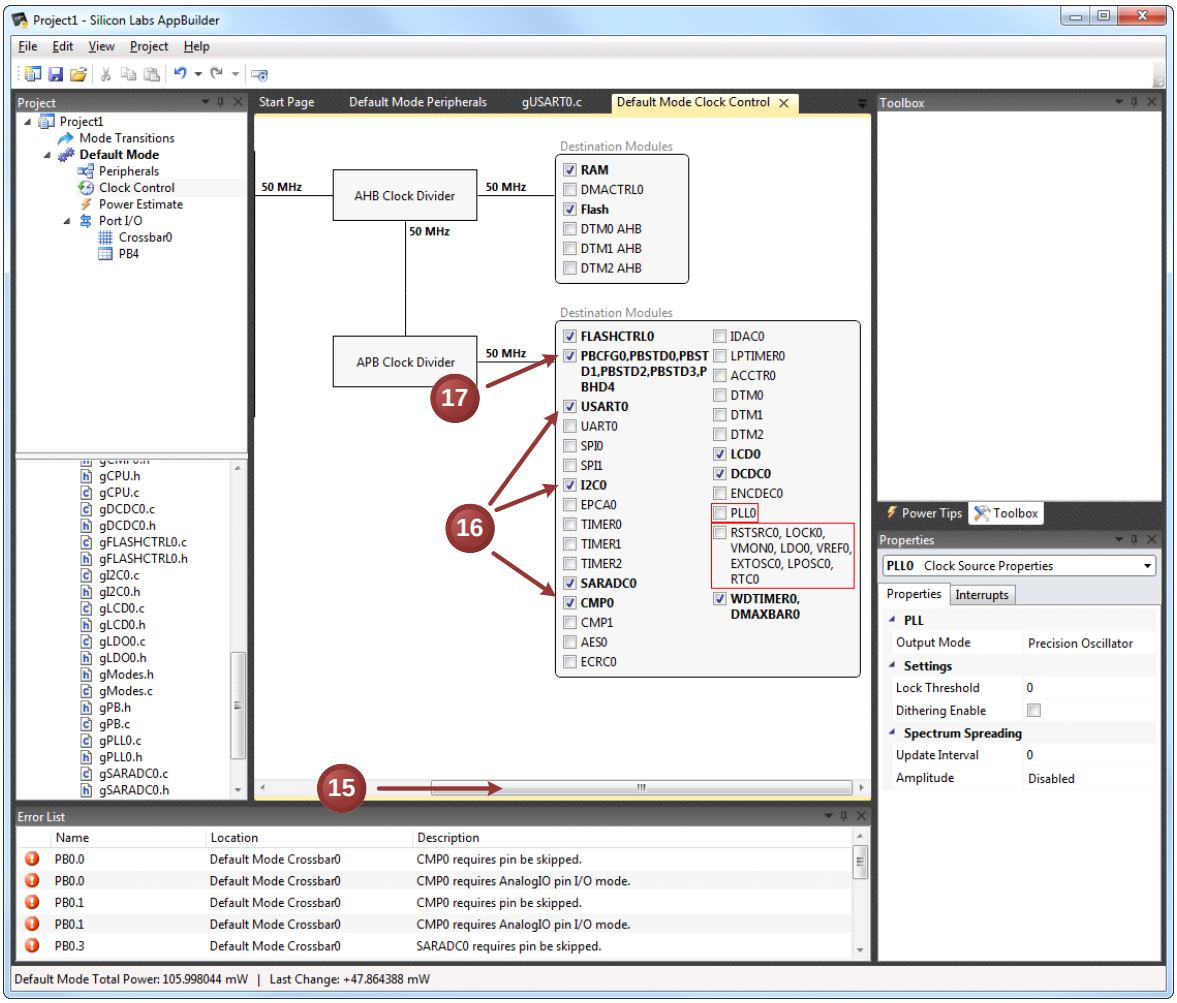
7. AppBuilder updates the code to be output in real-time. The code does not exist, however, until the project is exported using the **Export as Source** menu option or button. To view the changing source code, click on **USART0** in the Peripherals canvas.
8. Double-click on **gUSART0.c** in the **Code Viewer** window to open the file.
9. In the **Properties** window, change the receiver **Data Length** to **9 bits; 9th bit stored in FIFO** and check the **Expect Parity Bit** checkbox. The source code will update to match the changes.



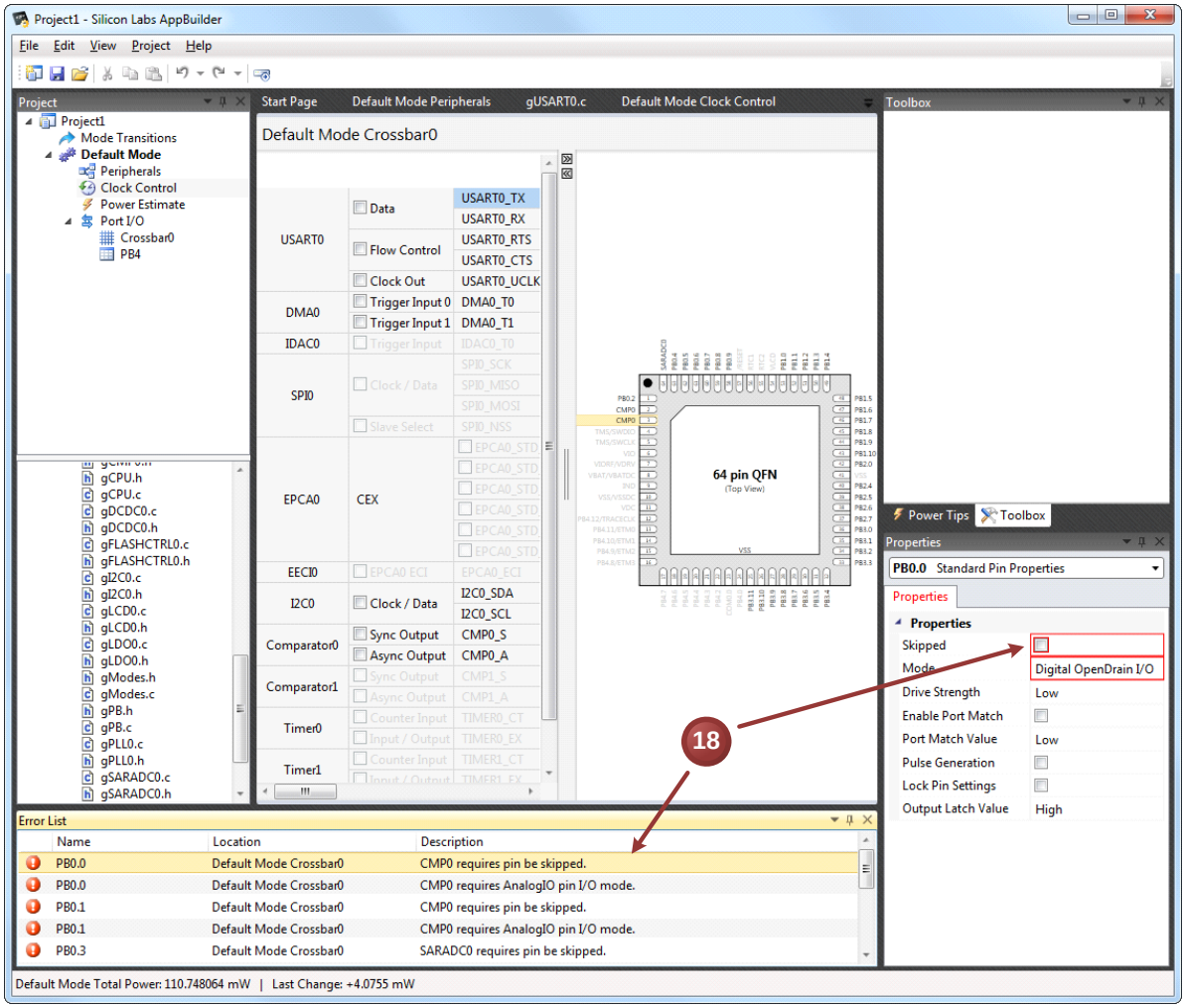
10. Double-click on the **Clock Control** canvas.
11. Let's set the PLL to use the factory-calibrated 50 MHz oscillator setting. Click on **PLL0**.
12. In the Properties window, set **Output Mode** to **Precision Oscillator**.
13. Click the radial button on the MUX to select the PLL0 as the AHB clock source.
14. Place the RTC in Crystal mode by left-clicking on RTC0 and setting **Clocking→RTC Mode** to **Crystal**. This will provide a clock for the LCD0 module, which is included to provide the bias for the DCDC0 module.



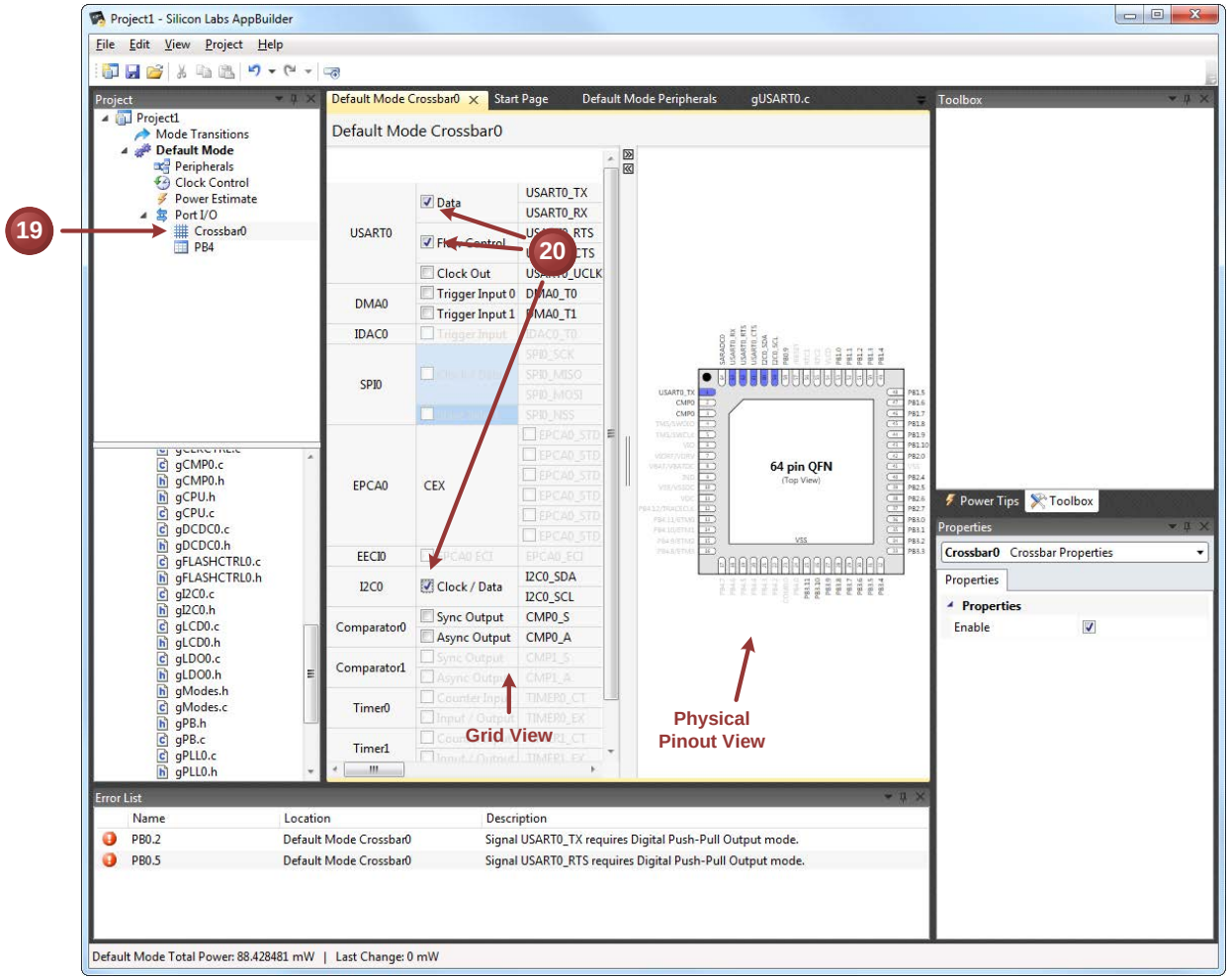
15. Scroll to the right of the Clock Control screen. Peripherals that have been configured in some way without the clocks enabled are highlighted in red.
16. Enable the clocks to all peripherals highlighted with red.
17. Also enable the clocks to the **Ports** (**PBCFG0, PBSTD0, ...**).



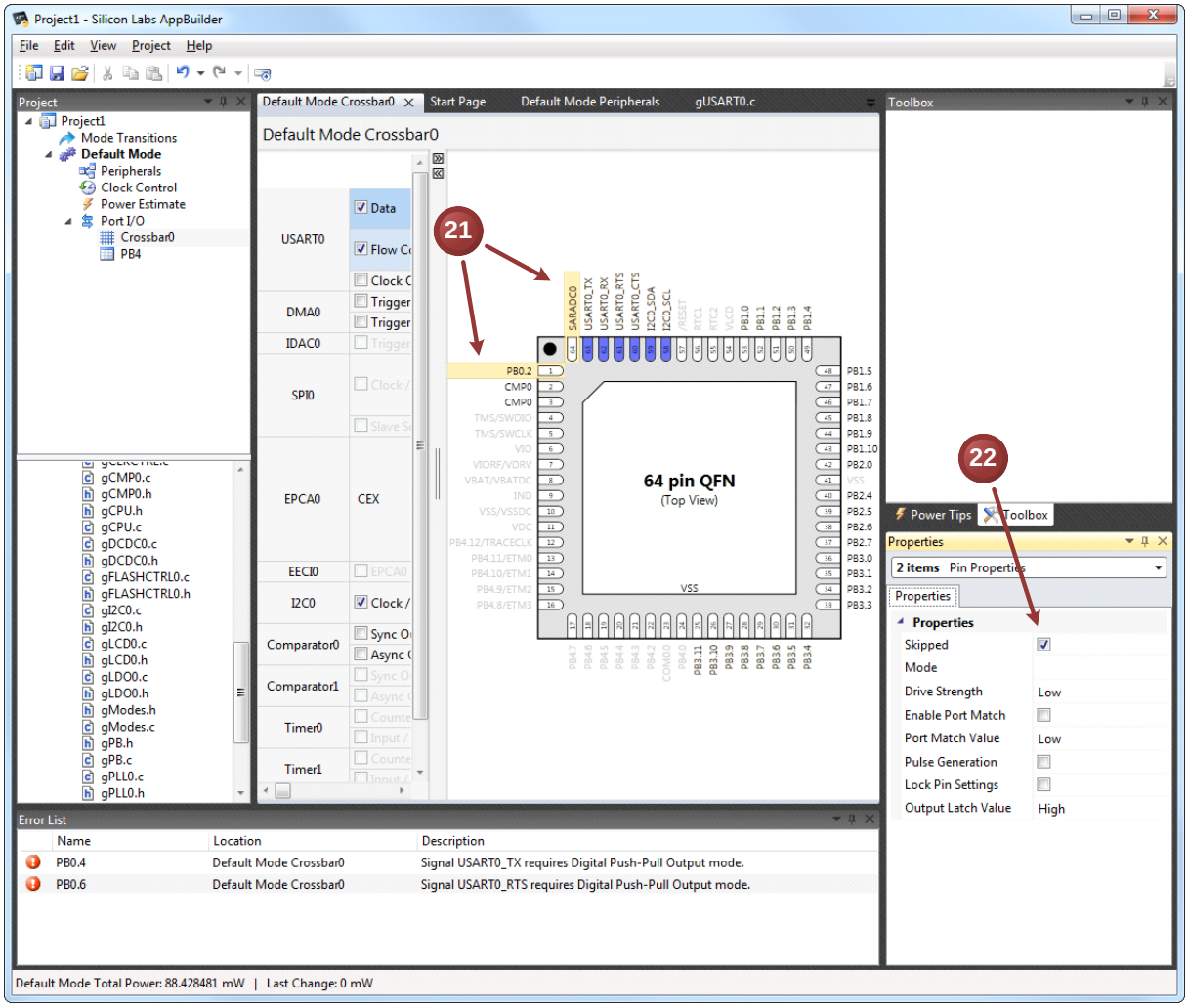
18. AppBuilder highlights any errors in a non-intrusive way. AppBuilder will not prevent a code export operation, even if there are errors or warnings. Double-click on an entry in the **Error List** to jump to the place in the configuration where the error can be fixed. Clear out all of the current errors. If an error cannot be immediately cleared, continue with the rest of the errors, as it may be related to another configuration error.



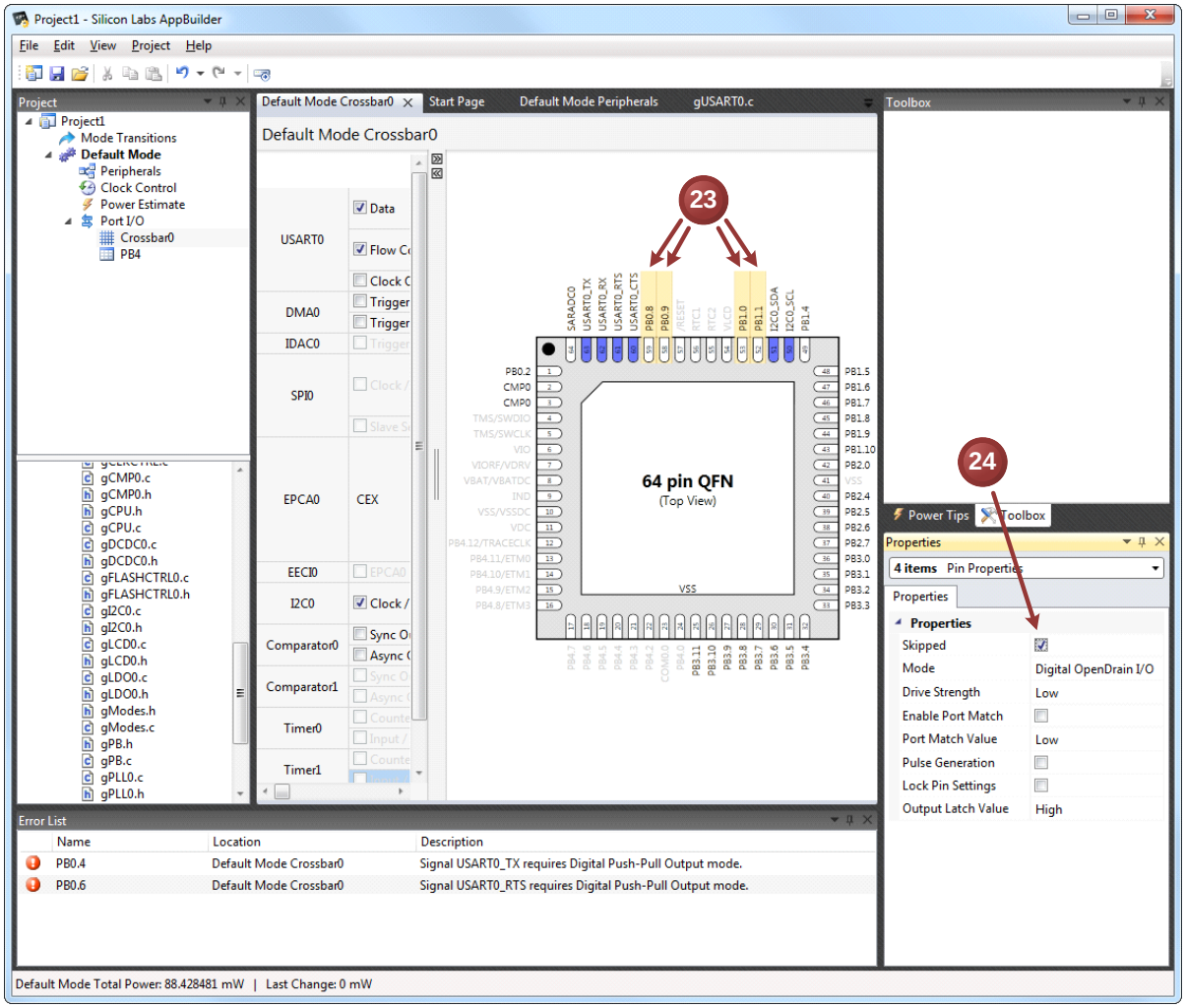
19. The SiM3L1xx devices have one priority-decoded crossbar to provide pin configuration flexibility. Double click on **Crossbar0** under **Port I/O** in the **Project** window. This window has two views: grid view on the left, and physical pinout view on the right. The size of each view can be adjusted using the divider in the middle.
20. Enable **USART0 Data**, **USART0 Flow Control**, and **I2C0 Clock / Data** by clicking the checkboxes in the left-hand grid area.



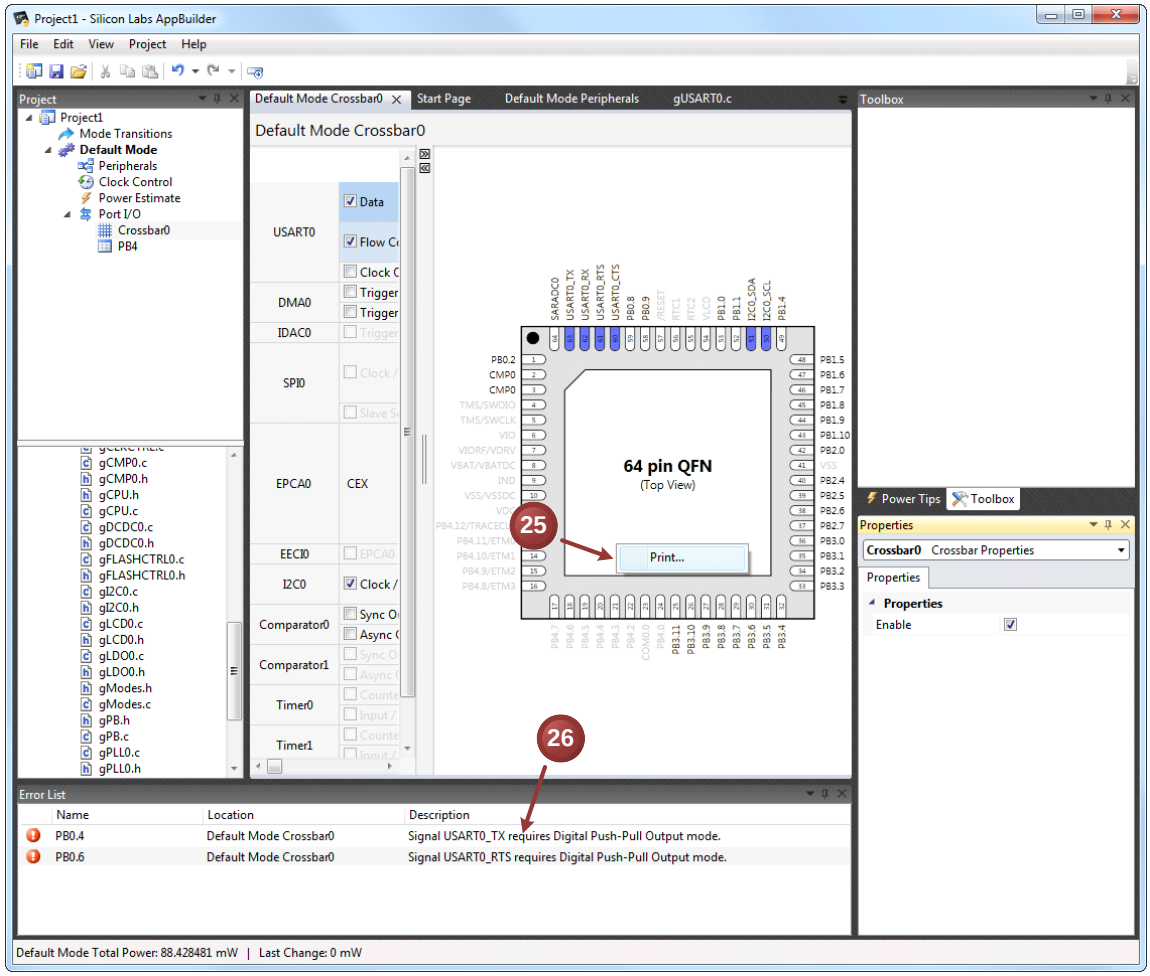
21. Pins PB0.0 and PB0.1 should already be skipped for CMP0 when clearing out the errors in the **Error List**. To move the USART0 signals to all be on the top of the device, shift-click on pins 1 and 64.
22. Check the **Skipped** checkbox in the **Properties** box.



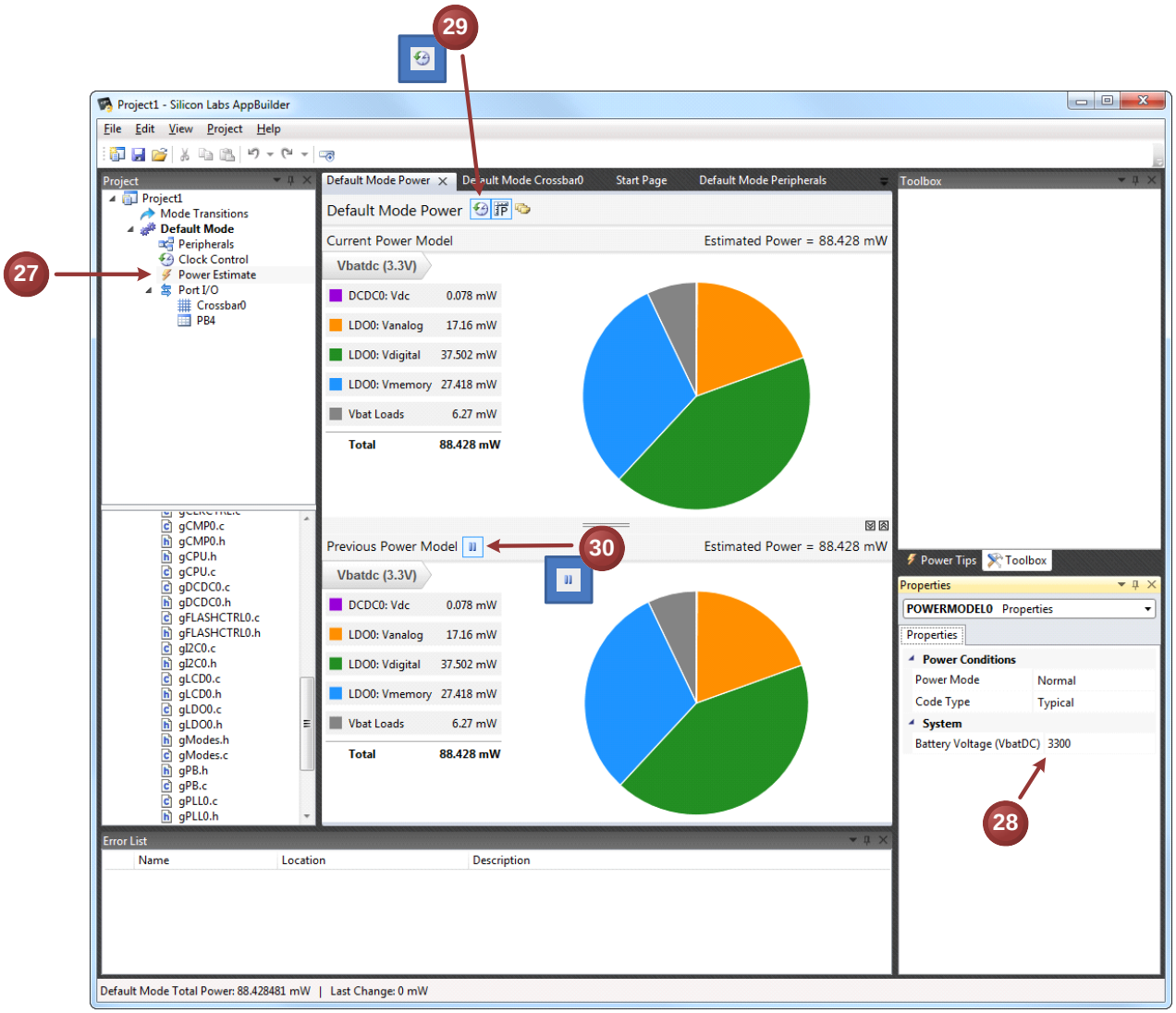
23. To isolate I2C0 from USART0 or make room to add another peripheral in the future (i.e. SPI0), move the I2C0 SDA and SCL signals to the right side of the device by skipping pins 59, 58, 53, and 52. Shift-click pins 59, 58, 53, and 52.
24. Check the **Skipped** checkbox in the **Properties** box.



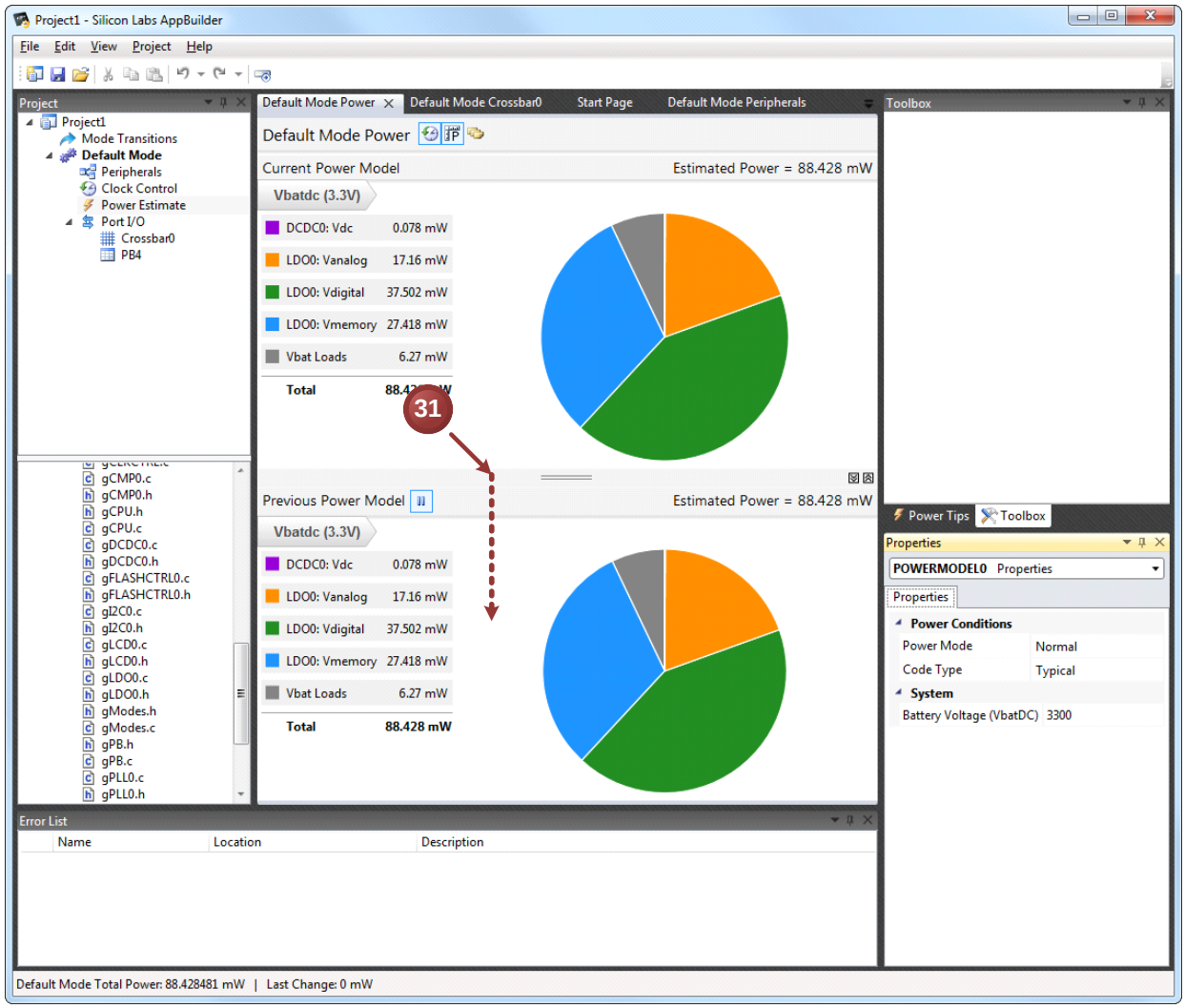
25. The physical pinout can be printed by right-clicking on the diagram and selecting **Print...**.
26. Clear any newly generated errors from the **Error List**.



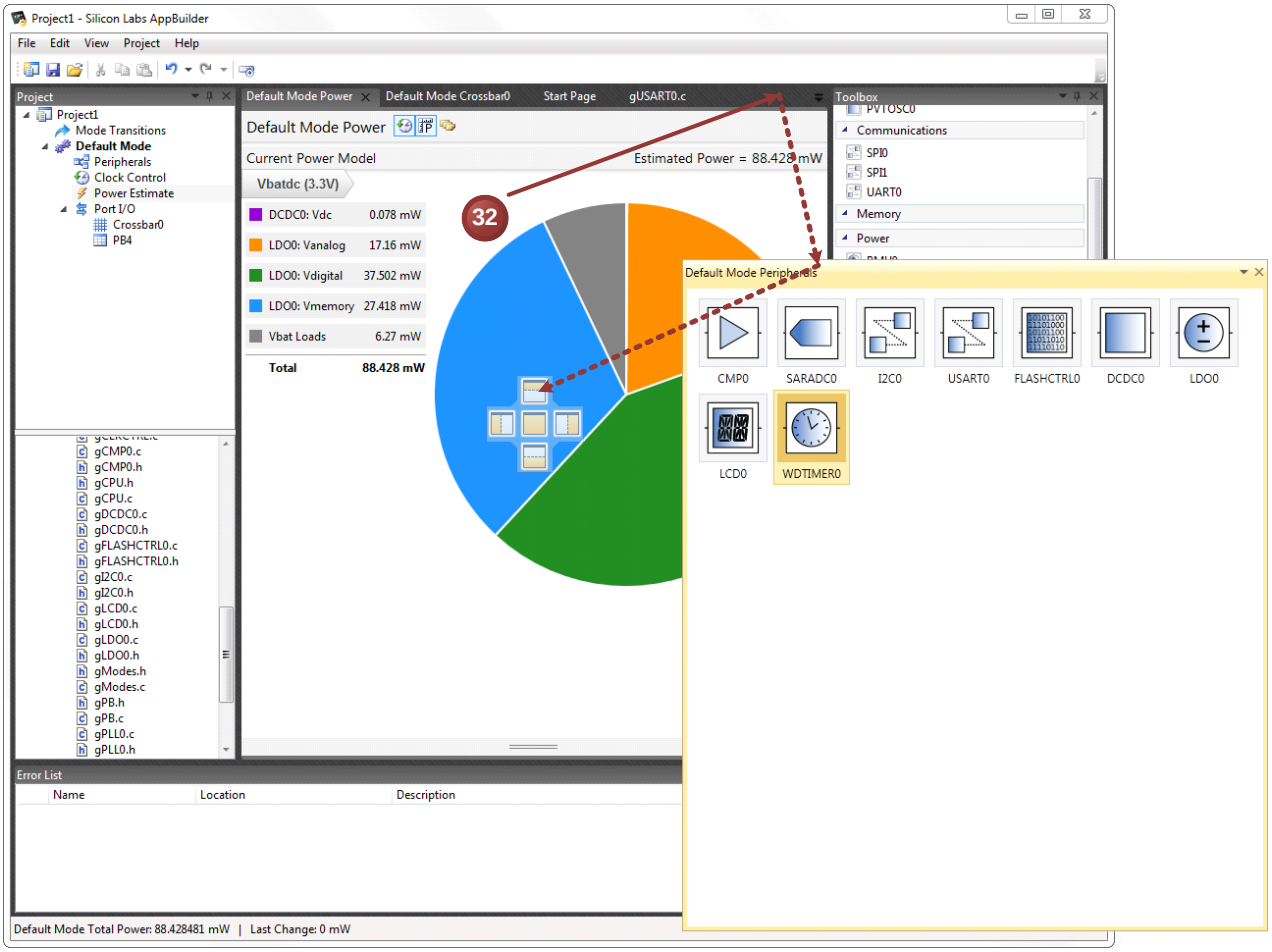
27. Now that the basic peripherals of the device are configured, we can look at the estimated power consumption of the system. Double click **Power Estimate** in the **Project** window. This feature is currently available for SiM3L1xx devices and uses sophisticated power models to estimate the power consumption of the device.
28. We will leave the battery voltage at 3.3 V (**Battery Voltage (VbatDC)** at **3300**) since this is the default power for the MCU card.
29. Take a snapshot of power by pressing the clock icon next to **Default Mode Power**.
30. Pause the snapshot by clicking the **Pause updates** button.



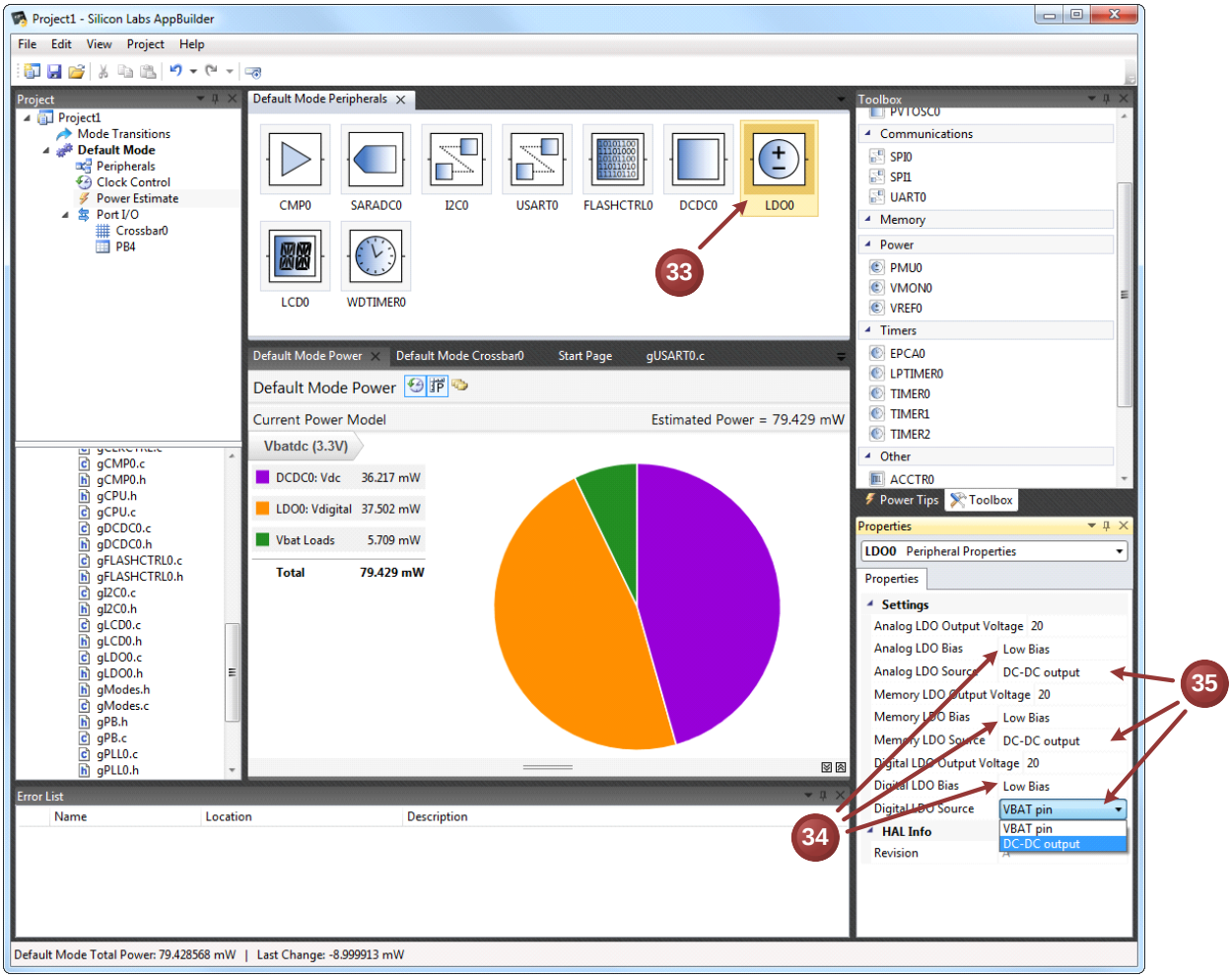
31. Hide the snapshot window (**Previous Power Model**) by dragging it down to the bottom until only the window handle is visible. We'll look at this window again at the end as a comparison.



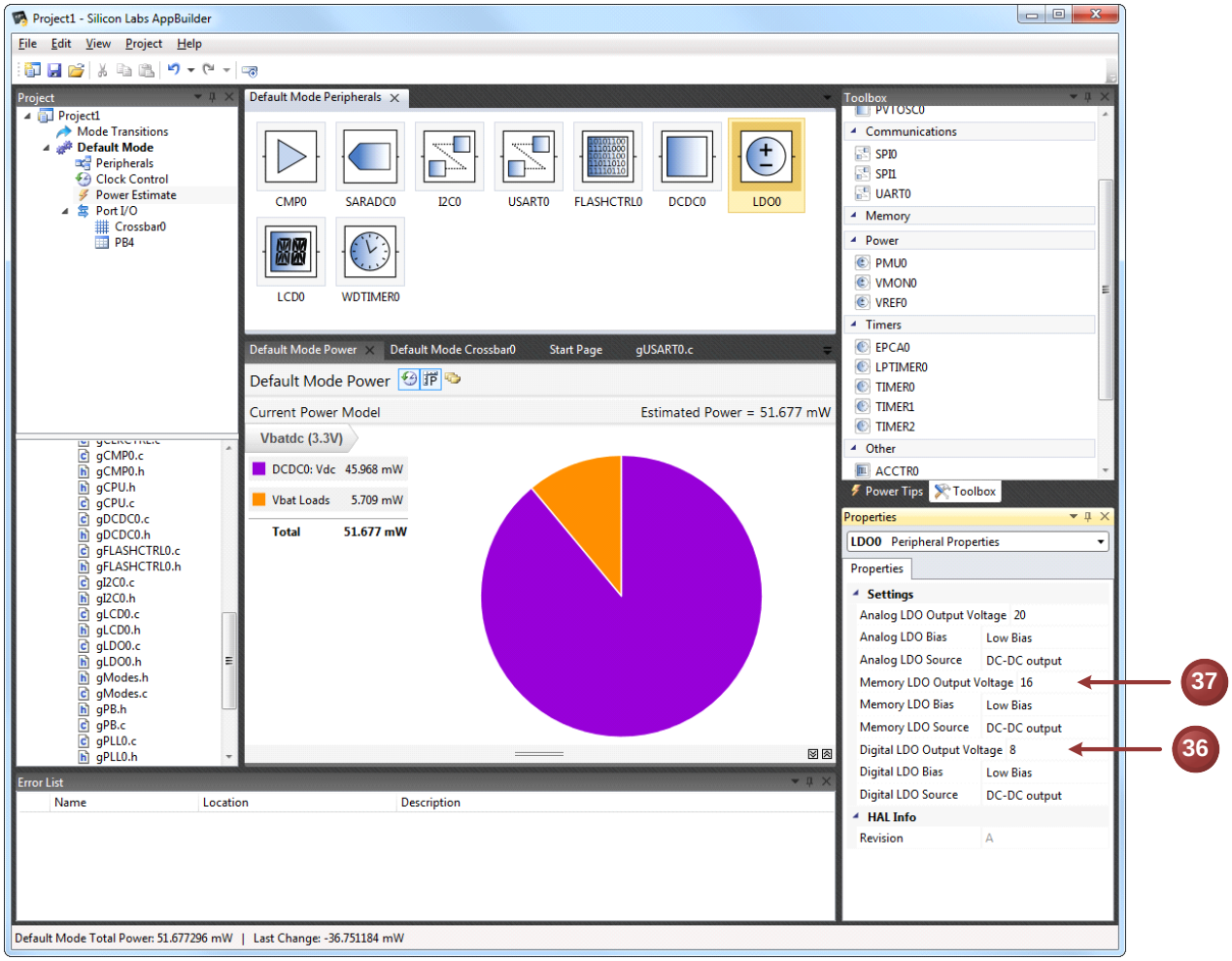
32. Click and drag the **Default Mode Peripherals** window at the top out of the frame and set it up so it's at the top. This allows visibility of both the **Default Mode Power** and the **Peripheral** windows at the same time, making it easier to make changes and observe the results.



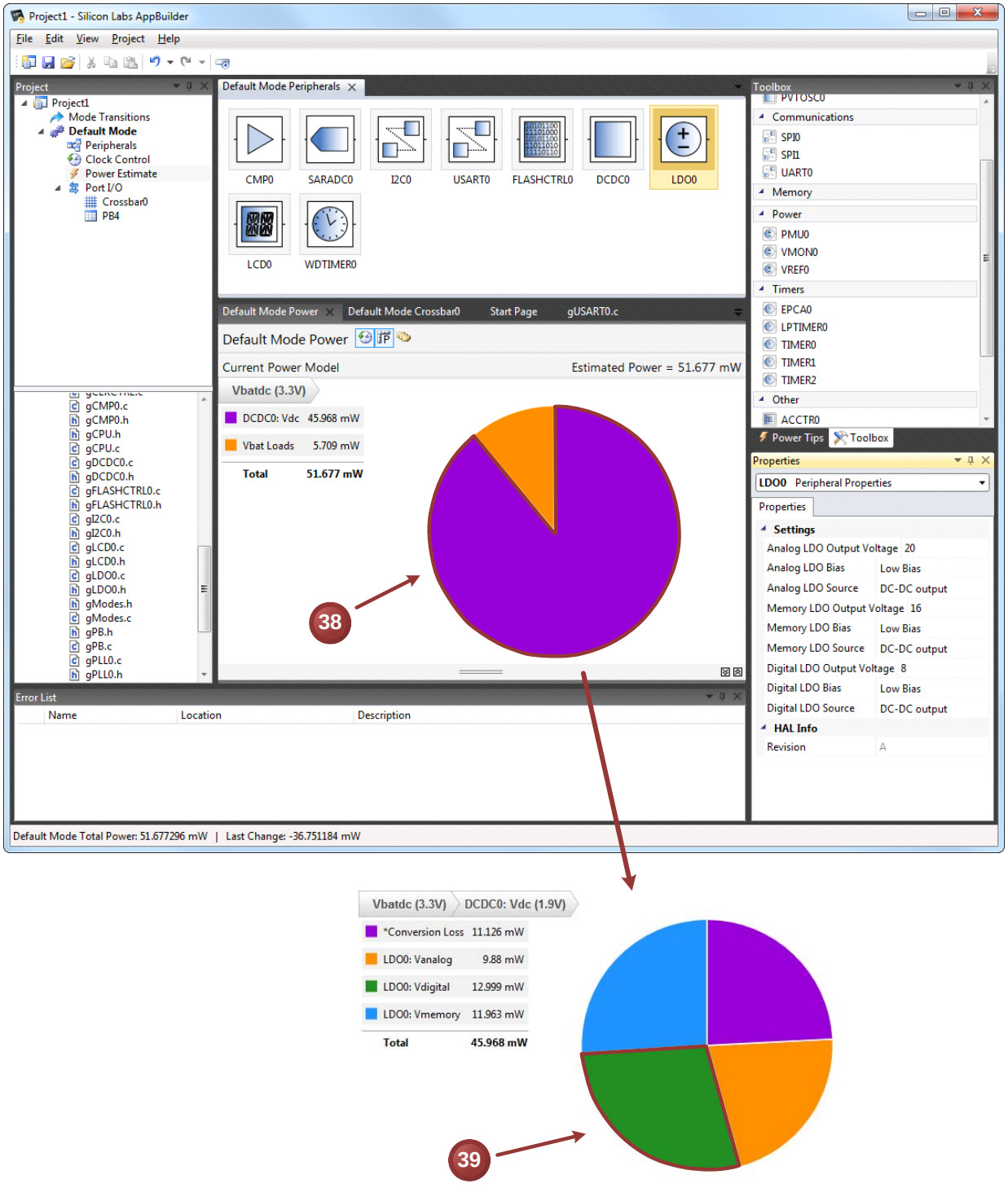
33. The first step in reducing the power consumption of the system involves taking advantage of the higher efficiency of the dc-dc converter. The LDO architecture results in about 50% efficiency. With a conversion from 3.6 V on the battery to the default of 1.8 V output on the LDO, this results in quite a bit of energy lost in the LDOs. We can reduce this energy loss by using the dc-dc converter, which can have much higher efficiency than the LDOs at higher loads (>2–3 mA). The voltage is converted from 3.6 V to 1.9 V in the dc-dc converter at higher efficiency, then to 1.8 V on the LDO output with the LDOs. Click on the **LDO0** peripheral in the **Peripheral** window.
34. Set the **Analog LDO Bias**, **Memory LDO Bias**, and **Digital LDO Bias** to **Low Bias**. This setting reduces the dynamic response of the LDOs in favor of lower power consumption.
35. Switch **Analog LDO Source**, **Memory LDO Source**, and **Digital LDO Source** to the **DC-DC output** selection from the **VBAT pin**.



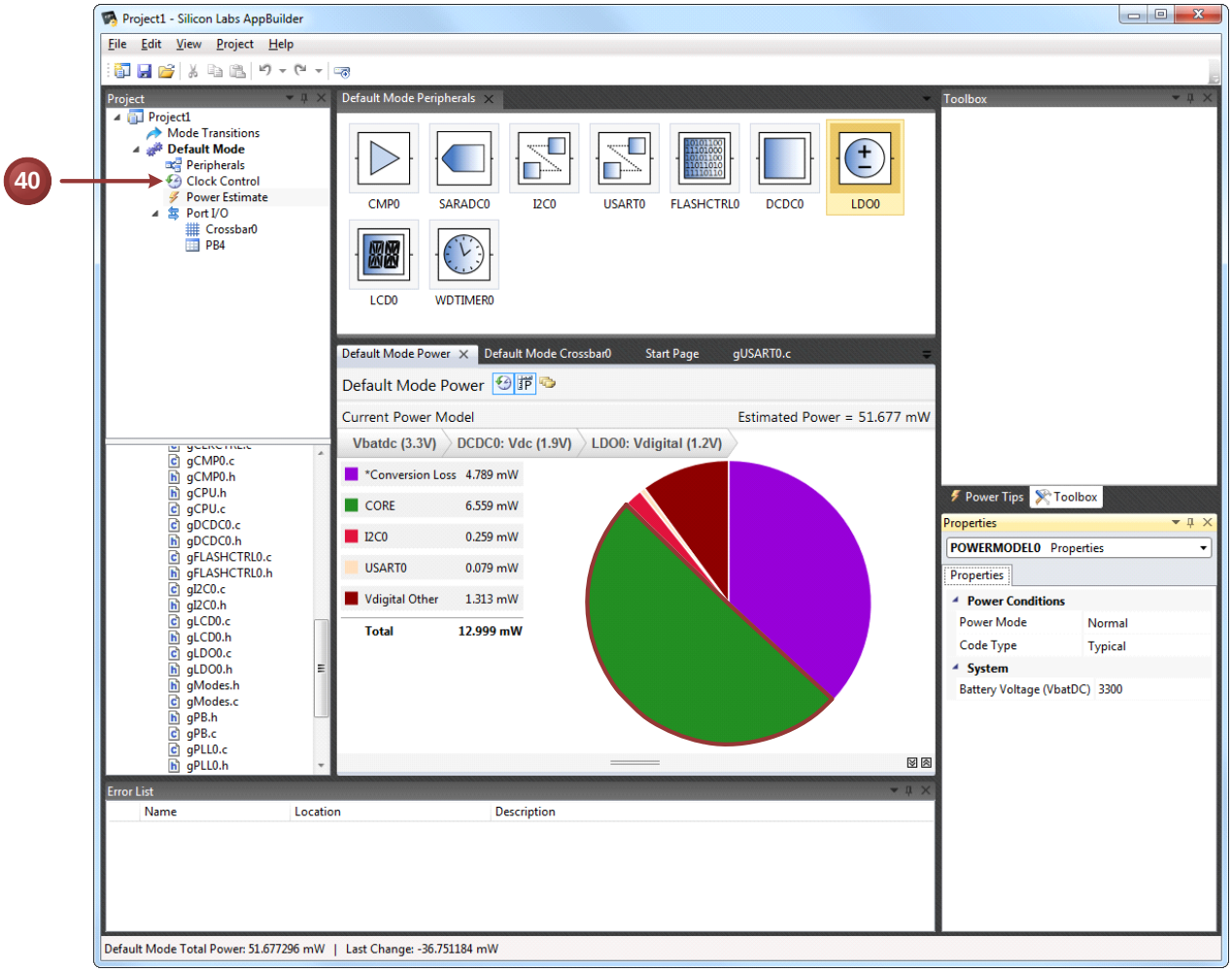
36. We can also reduce general power consumption by independently lowering the output voltage of the LDOs. The default output of 1.8 V provides enough headroom for proper operation across all process, voltage, and temperature variations, but it is not needed by the digital and memory logic the vast majority of the time. These LDOs feature programmable voltage scaling to reduce power whenever possible. We can set the digital LDO to 1.2 V and the memory LDO for 1.6 V. The analog LDO should remain at 1.8 V for proper operation.In the LDO peripheral, set the **Digital LDO Output Voltage** to 1.2 V (value of **8**).
37. Set the **Memory LDO Output Voltage** to 1.6 V (value of **16**).



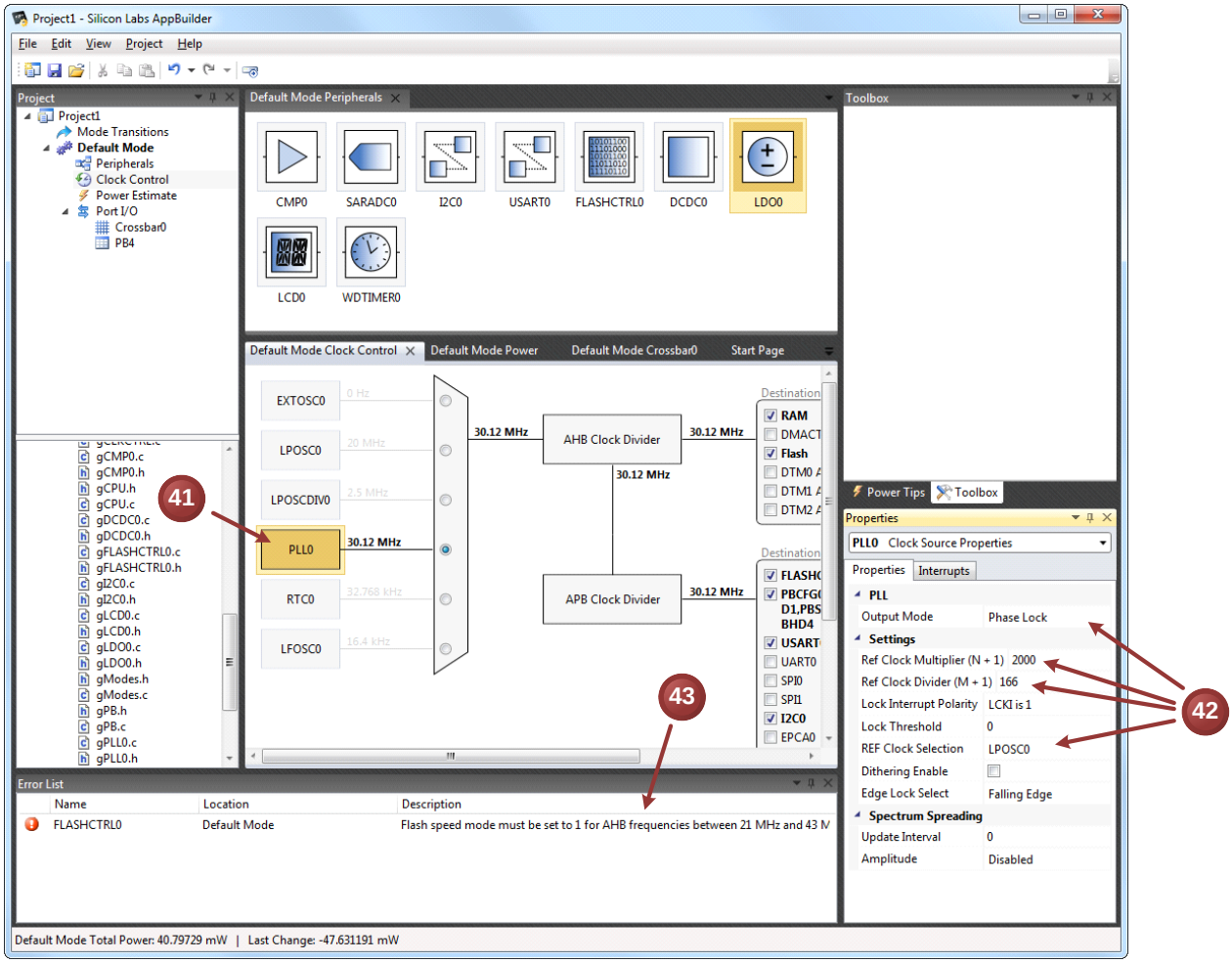
38. Now, let's look at each of these domains (digital, memory, and analog) to find ways to reduce more power. Let's start with the digital domain. Double-click on the **DCDC0** piece of the pie.
39. Double-click on the **LDO0: Vdigital** piece of the pie.



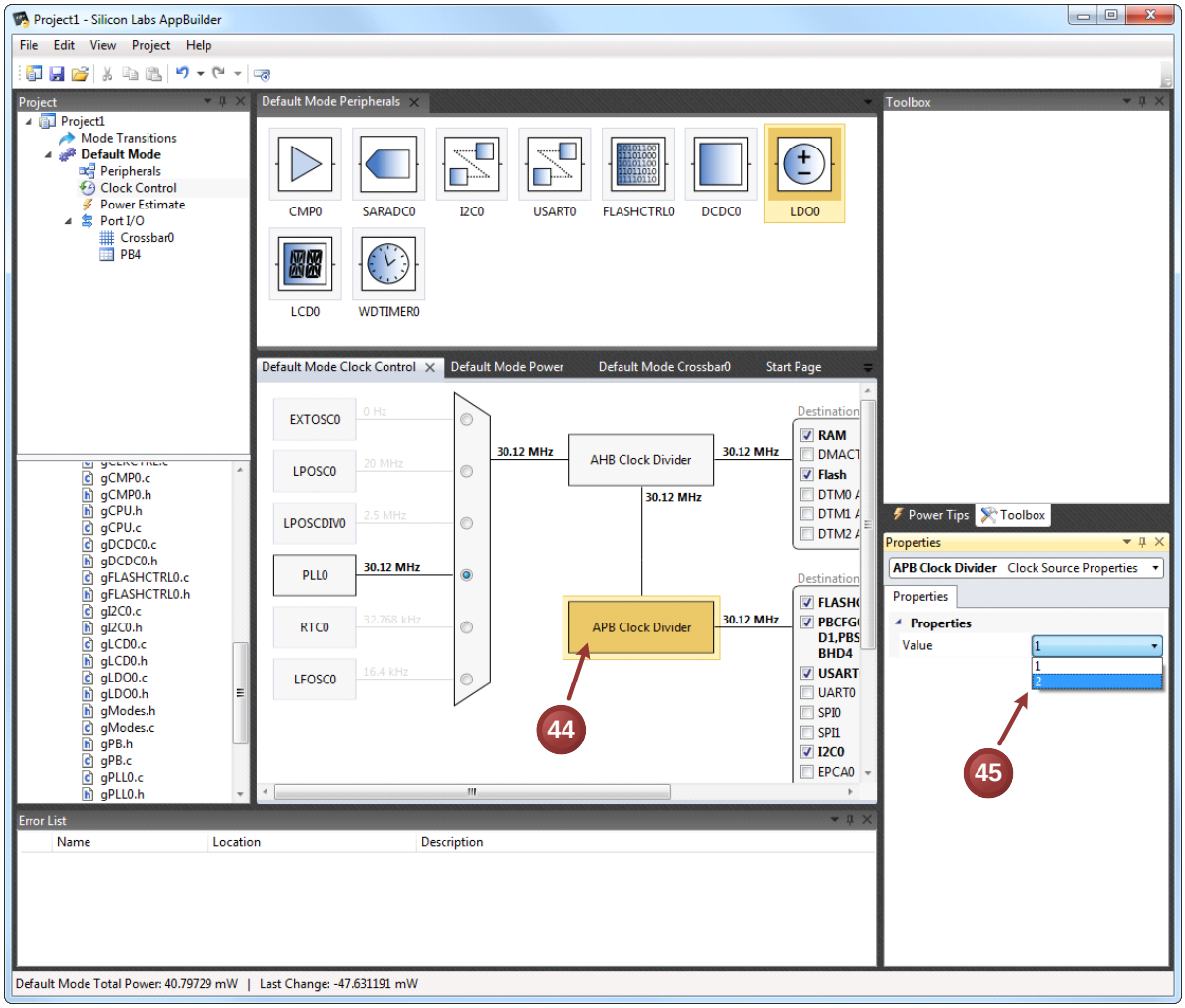
40. The core is currently the largest contributor to digital power consumption. At this time, the core is running at 50 MHz using the precision oscillator. However, what if 50 MHz is more than what is needed to run the main algorithm of the system? We can reduce this power consumption by using an external oscillator and reducing clock speeds to the minimum required. Click on **Clock Control** in the **Project** window.



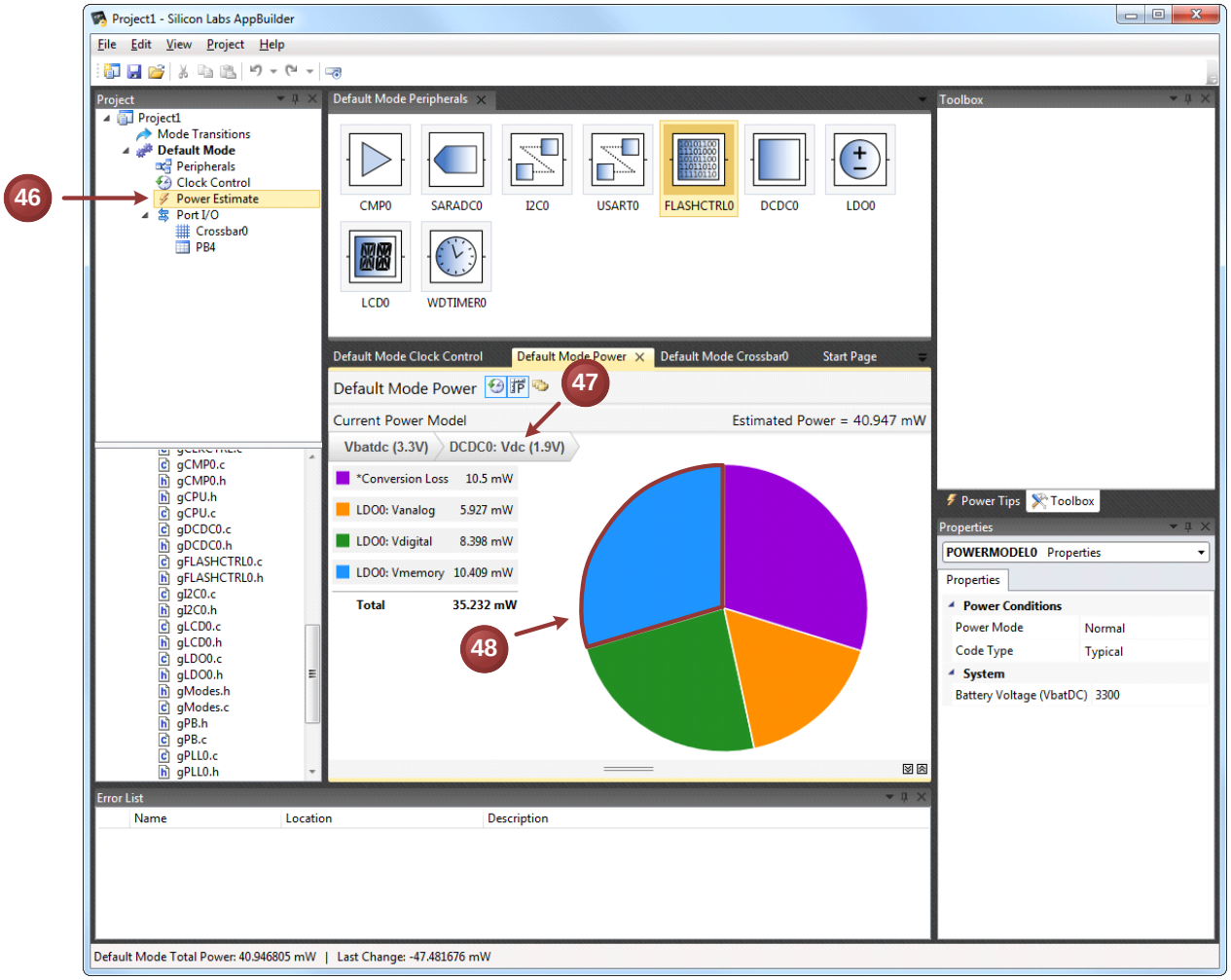
41. If the algorithm only needs ~30 MHz for ideal operation, then we can switch to this frequency instead of 50 MHz and reduce the power consumption. Click **PLL0**.
42. Set **Output Mode** to **Phase Lock**, set **REF Clock Selection** to **LPOSC0**, set **Ref Clock Multiplier (N + 1)** to **2000**, and set **Ref Clock Divider (M + 1)** to **166** for a ~30 MHz output frequency.
43. Fix the flash access speed error by double-clicking on it in the **Error List**. This changes the flash access speed to match the new AHB frequency.



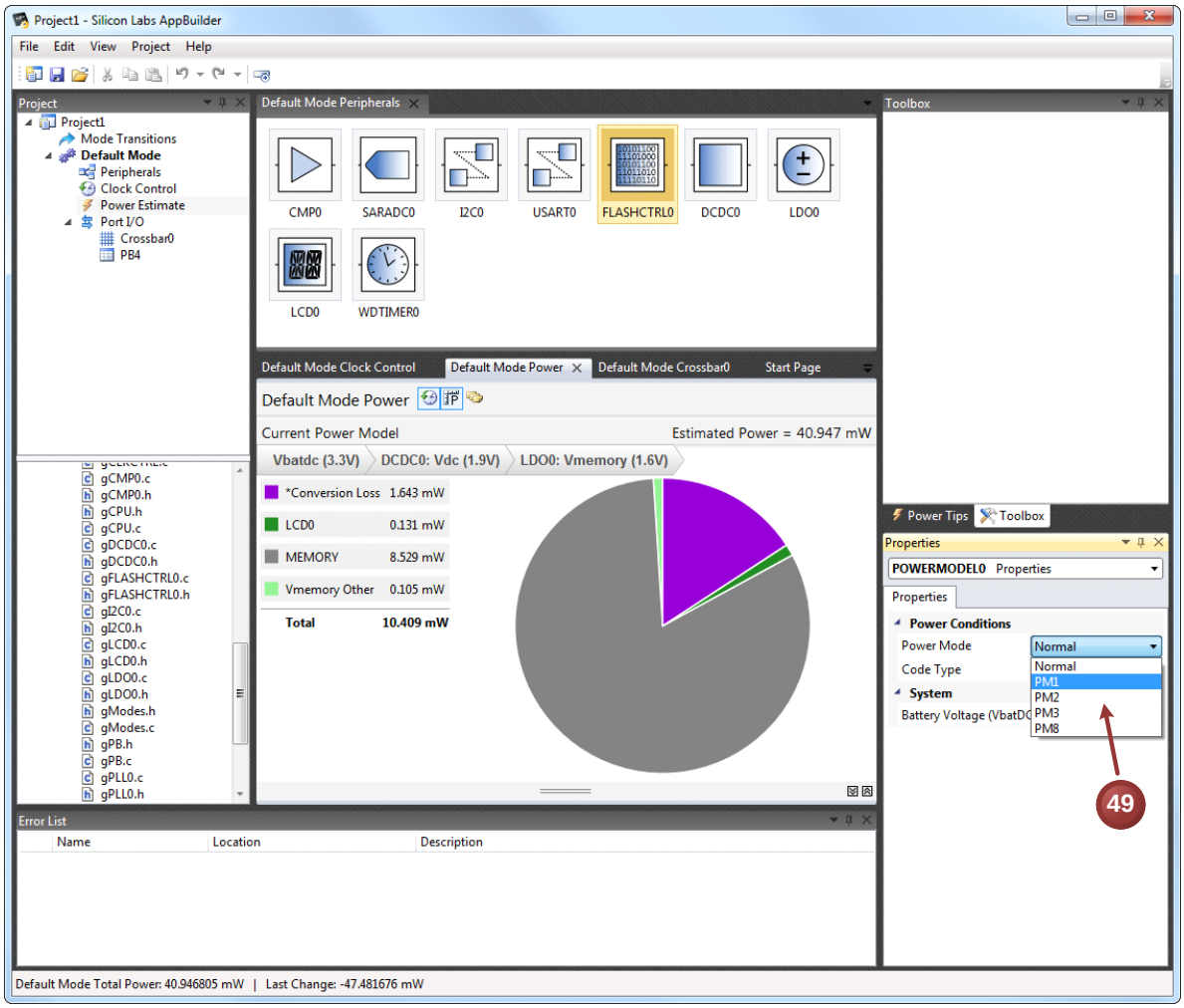
44. Suppose the peripherals only need 10 MHz to run at the required baud rate for USART0, for example. We can set the APB Clock Divider to half the AHB clock to reduce the digital power consumption even more. Click on **ABP Clock Divider**.
45. Set **Value** to **2** in the **Properties** window.



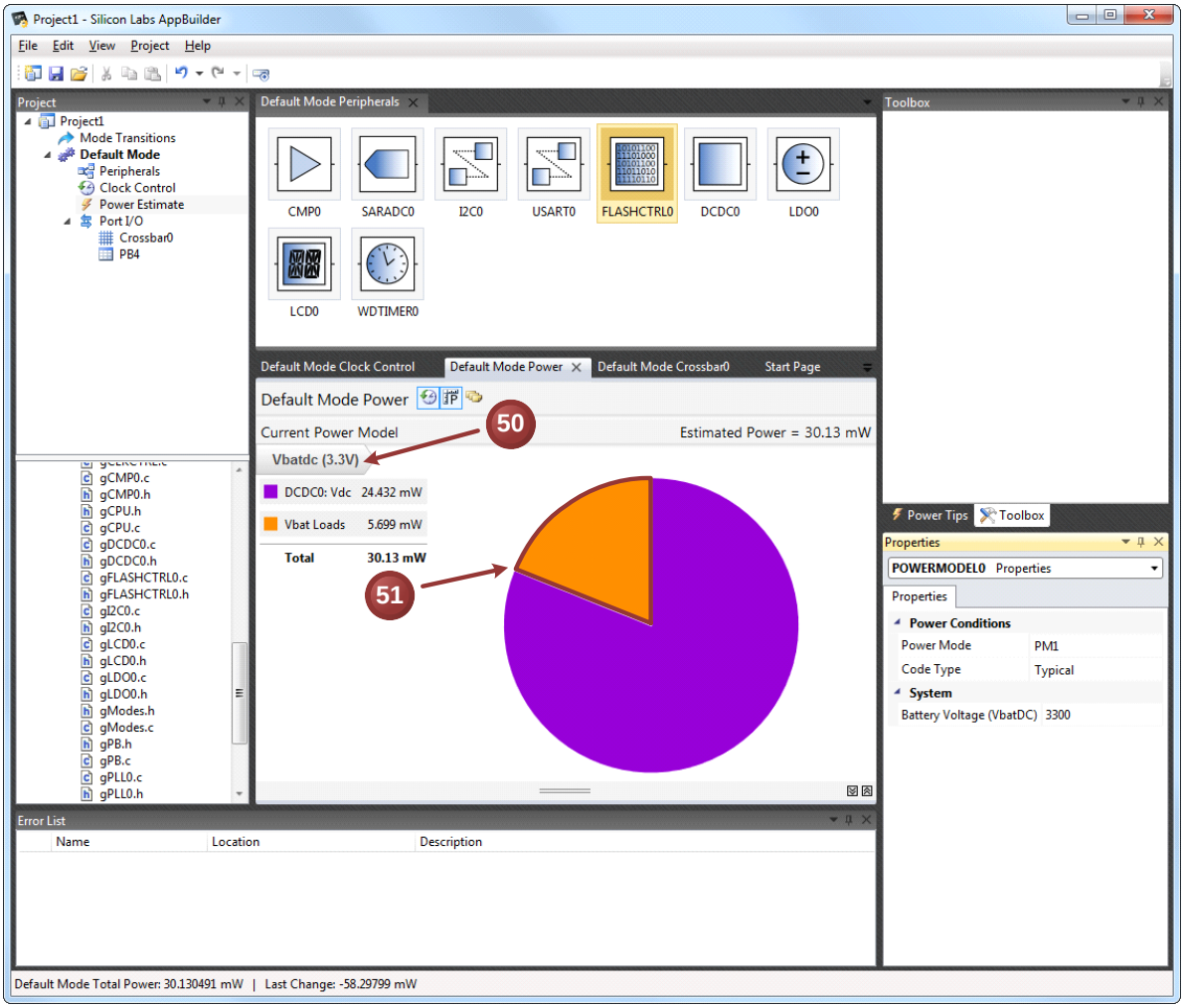
46. We can now look at ways to reduce the memory domain consumption. Click on the **Default Mode Power** tab or double-click on **Power Estimate**.
47. Click on the **DCDC0:Vdc (1.9 V)** label at the top to move up one voltage domain in the pie charts.
48. Double-click on the **LDO0: Vmemory** piece of the pie.



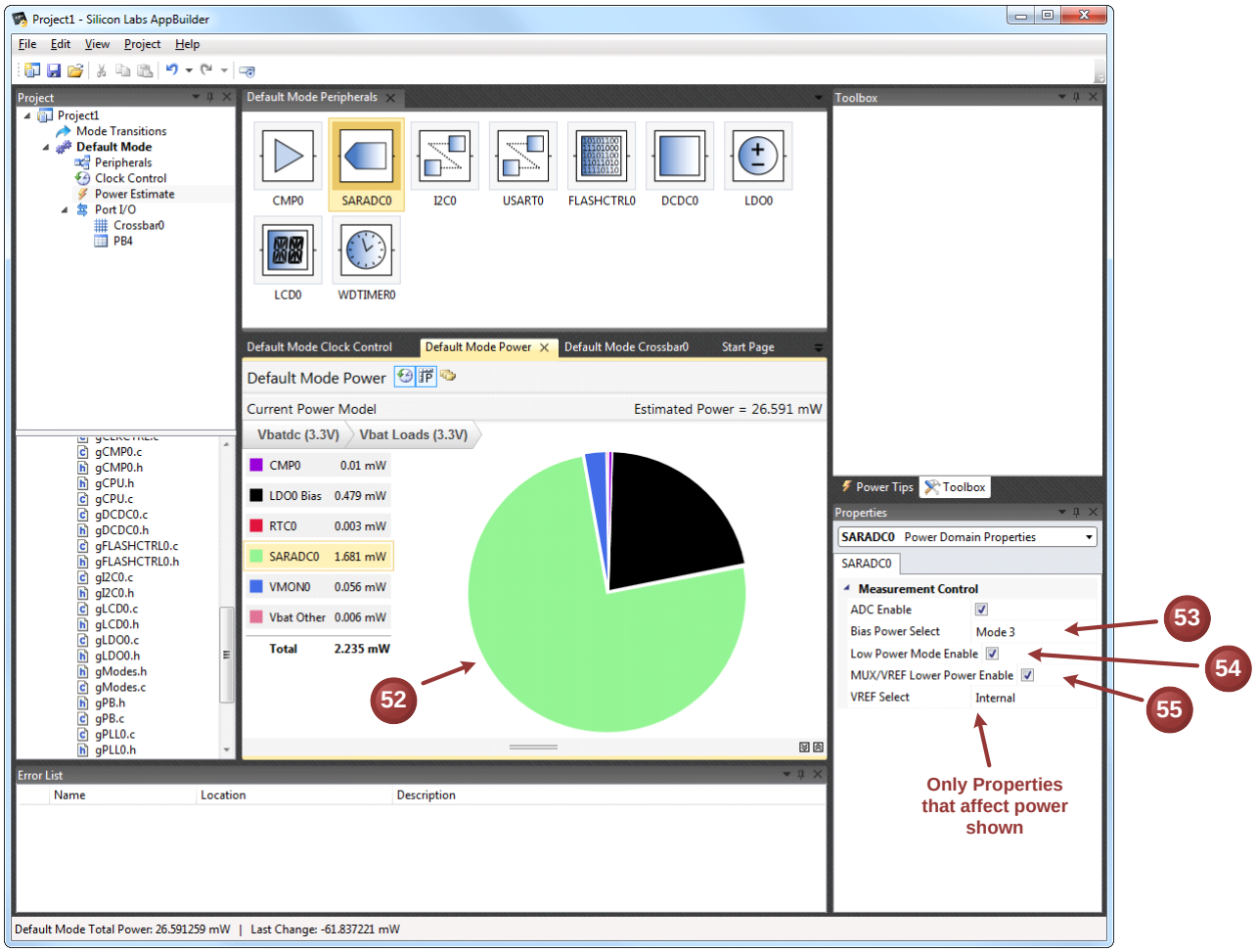
49. Power Estimator includes models for different usage cases. The default is **Normal**, where code runs from flash. However, if the firmware includes a small algorithm that needs to run often, we can save power by moving this algorithm to RAM. This occurs in Power Mode 1, or PM1. Switch **Power Mode** from **Normal** to **PM1** in the **Properties** window.



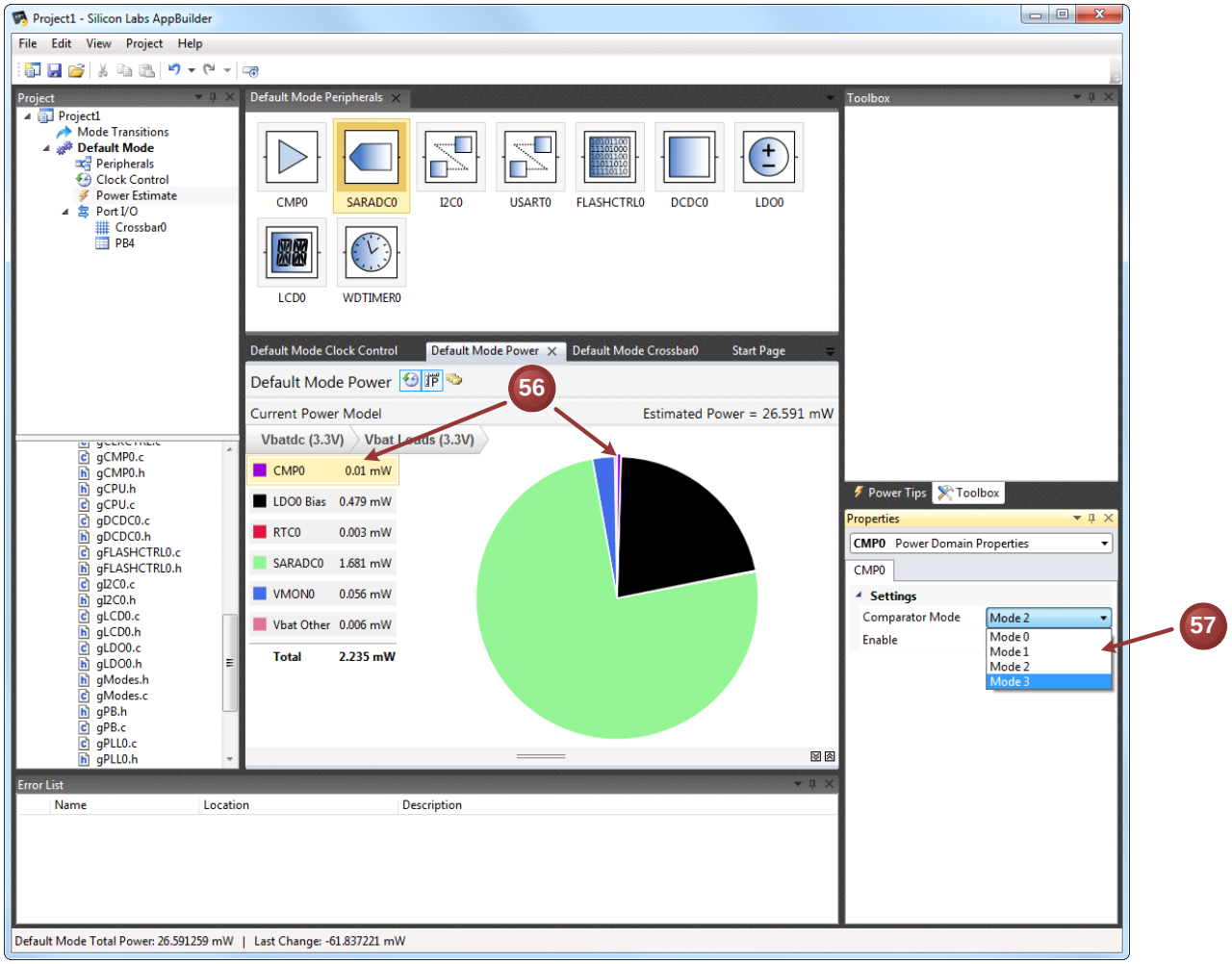
50. The analog domain is split into two parts: some analog peripherals run from LDO0, e.g.,Vanalog, and some run from VBAT directly. Let's look at the VBAT domain. Click on the **Vbatdc (3.3 V)** label at the top.
51. Click on the **Vbat Loads** piece of the pie.



52. If we look at the general analog power consumption, the ADC is largely the dominant player. Single click on the **SARADC0** piece of the pie.
53. When single clicking a pie piece, Power Estimator shows only the peripheral properties that are related to power. This helps narrow down the properties and modes that will help reduce overall system power consumption. We can switch the ADC to a lower power mode to reduce the power consumption using these properties. Change **Bias Power Select** from **Mode 0** to **Mode 3**.
54. Check the **Low Power Mode Enable** checkbox.
55. Check the **MUX/VREF Lower Power Enable** checkbox.



56. We can also reduce the comparator 0 power consumption, though it is already relatively small. Click on **CMP0** in the pie chart area or in the domain list.
57. Switch **Comparator Mode** from **Mode 2** to **Mode 3**.



We've now addressed each of the main power areas. Here's a review of everything we did:

a. Moved LDOs to the DC-DC converter

b. Used Voltage Scaling to reduce the LDO output voltage for digital and memory

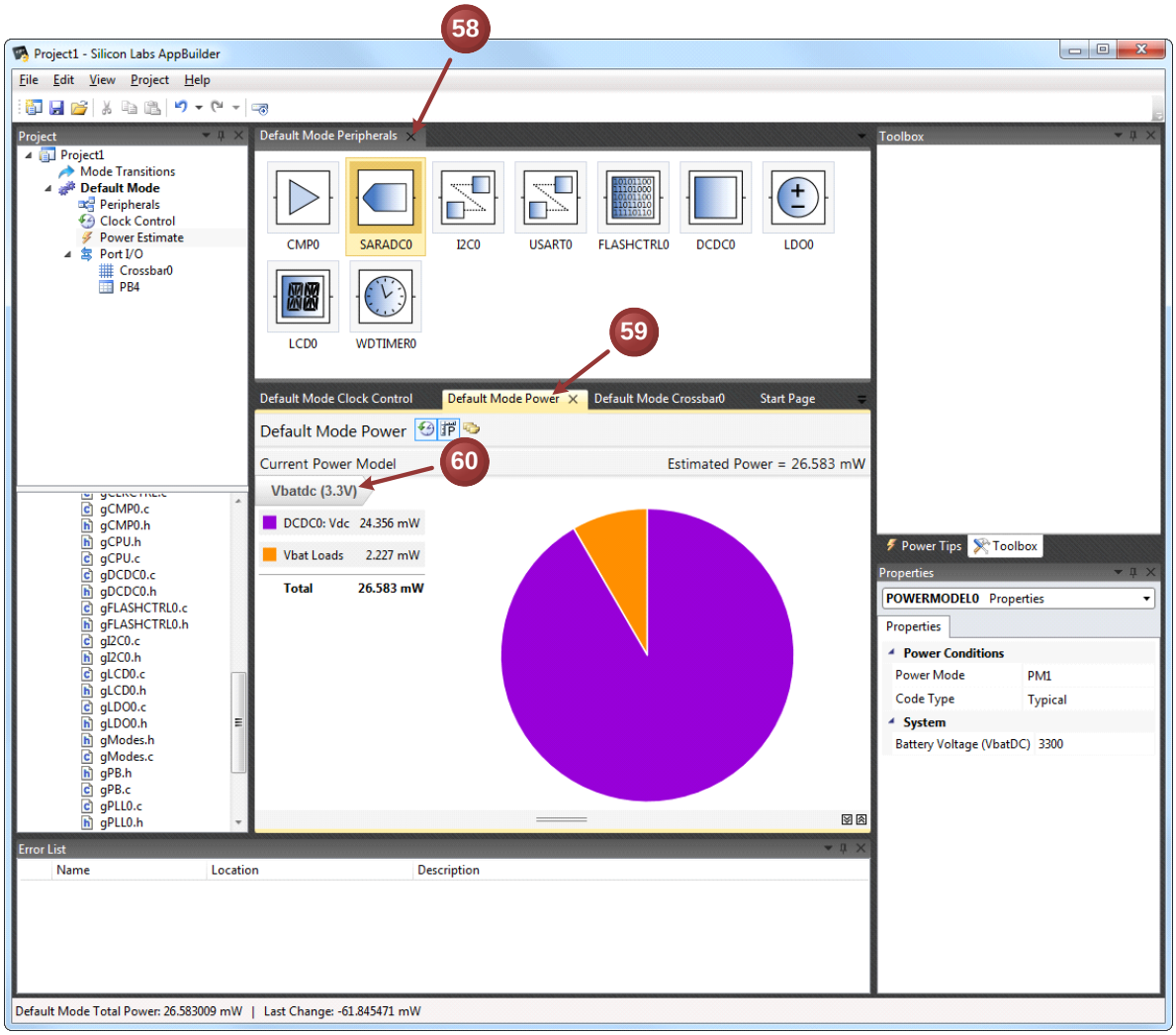
c. Reduced digital power consumption by reducing the AHB and APB clocks to the minimum required by the system and by switching the PLL to the internal divided Low Power Oscillator (LPOSC0).

d. Reduced memory power consumption by switching the main algorithm to RAM.

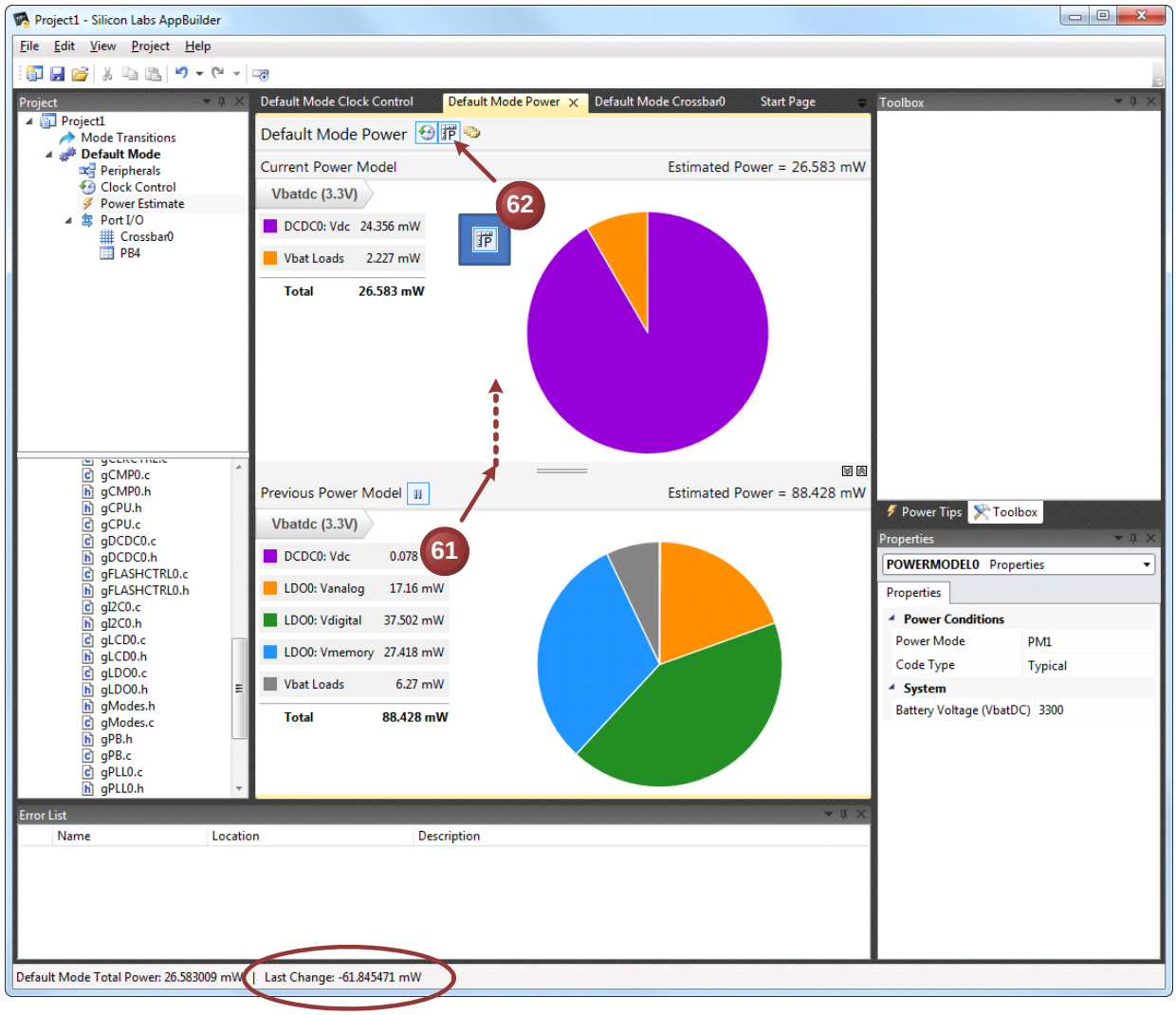
e. Reduced analog power consumption by switching our peripherals to their lowest power mode.

Now let's look at the difference in the power consumption from the original configuration.

58. Close the **Default Mode Peripherals** tab.
59. Click on the **Default Mode Power** tab.
60. Click on the **Vbatdc (3.3 V)** label at the top to move back to the top level. The power consumption should be somewhere around ~26 mW.



61. Drag the snapshot tab up from the bottom of the screen. The difference is staggering: ~88 mW (26.797 mA) versus ~26 mW (8.055 mA) represents a ~70% reduction in power consumption. These Power Aware tools enable easy manipulation of the device's power consumption with lots of visibility into the operation of the device.
62. To switch the power units to current instead of power, click the measurement units button at the top.




63. Now that we know the system current consumption, we can set the DCDC0 module to the correct settings. For a load between 5 and 15 mA, **Power Switch Mode** (PSMD) should be set to **Mode 0**; **Asynchronous Mode Enable** (ASYNCEN) should be disabled (unchecked), and the **Minimum Pulse Width** (MINPWSEL) should be set to **Disabled**. Click on **Peripherals**; click on the **DCDC0** module, and set the fields appropriately.

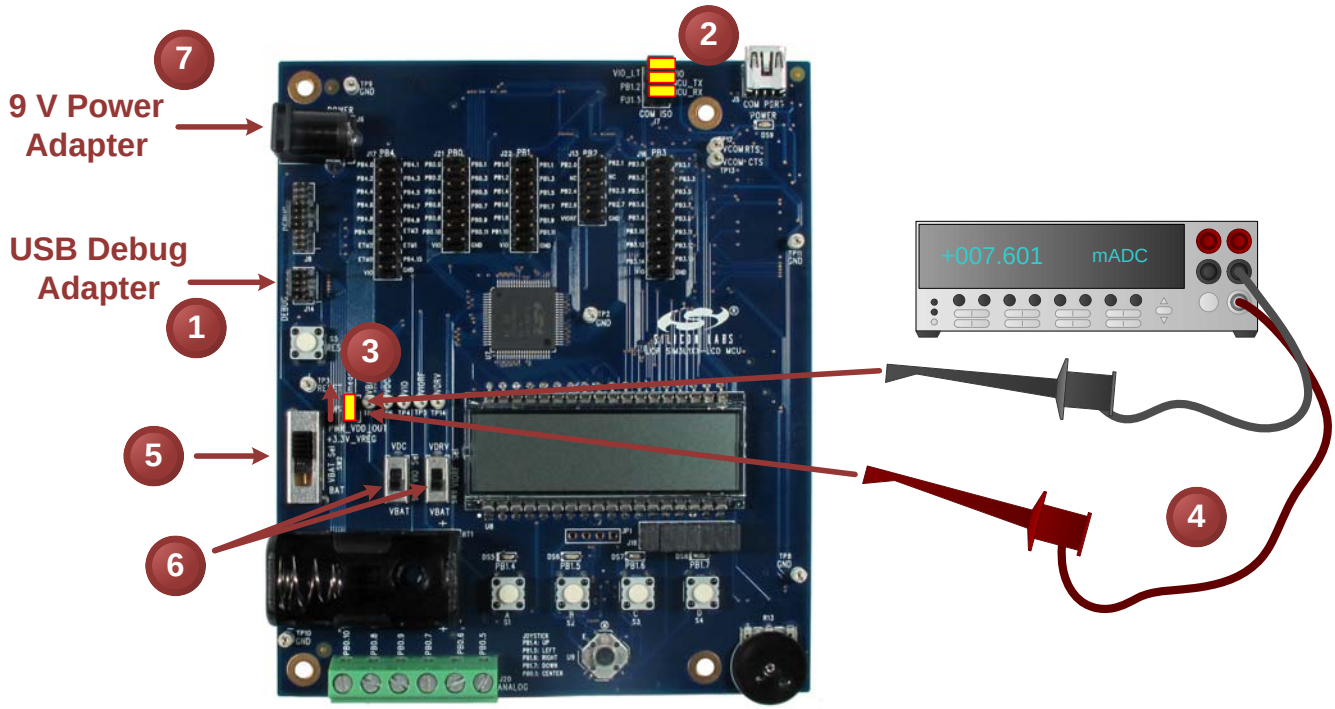
# 5. Testing Power Consumption on Hardware

## 5.1. Hardware Setup

To configure the SiM3L1xx MCU card for power measurement with a fixed VBAT:

1. Connect a USB Debug Adapter to the 10-pin CoreSight connector (J14).
2. Remove the three shorting blocks from J7.
3. Remove the JP2 **Imeasure** shorting block.
4. Connect the positive terminal of a multimeter to the bottom pin of JP2, and connect the negative terminal of a multimeter to the top pin of JP2. The arrow next to this jumper indicates the direction of current flow.
5. Move the **VBAT Sel** switch (SW2) to the middle **+3.3V_VREG** position.
6. Move the **VIO Sel** (SW8) and **VIORF Sel** (SW9) switches to the bottom **VBAT** position.
7. Connect the 9 V Power Adapter to J6.
8. Download the code to the board.
9. Remove the debug adapter connection (typically not necessary in the mA range).
10. Measure the power of the device.

**Note:** Ensure the multimeter is in the correct current range for the measurements, as an incorrect setting can cause issues when running or debugging code.



## 5.2. Firmware Setup

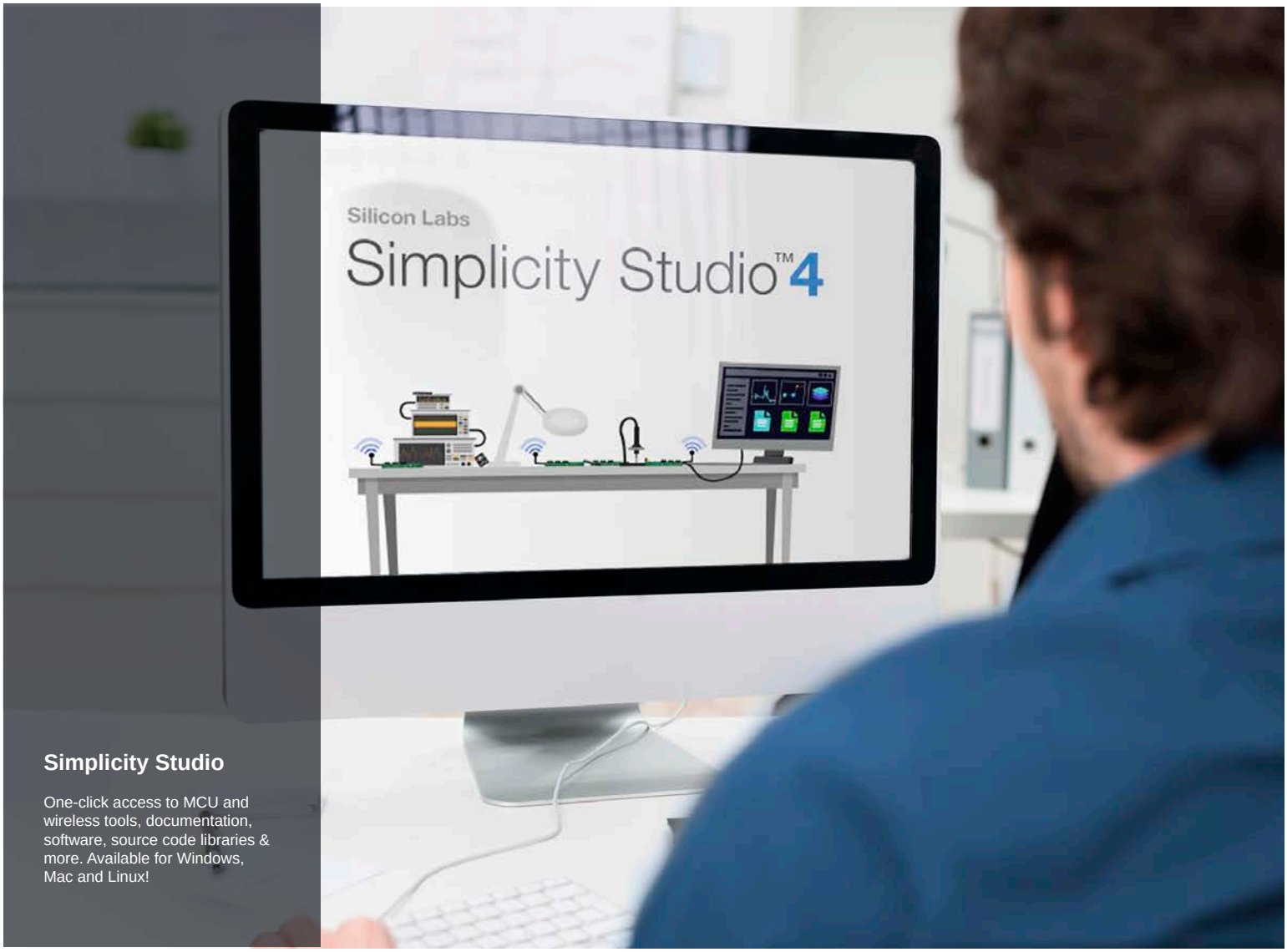
At any point in time in the v1.1.1 AppBuilder exercise:

1. Export code to a Precision32 project using the **Export as Source** menu option or button. Selecting the **Open After Export** option will open a new instance of the IDE.
2. Build and debug the exported source. The hardware should be configured as described in “5. Testing Power Consumption on Hardware”.
3. Run the code.
4. Measure the current consumption compared against the Power Estimator estimated result.

Two AppBuilder projects and code projects are included with this application note:

- “AN756_Power_Estimator_Pre_Power_Reduction” corresponds to step 26 in the AppBuilder procedure. Expected current consumption is ~25 mA.
- “AN756_Power_Estimator_Post_Power_Reduction” corresponds to step 63 in the AppBuilder procedure. Expected current consumption is ~8 mA.

The Post Power Reduction project includes code to jump to a while(1) loop in RAM to illustrate the Power Mode 1 (PM1) power consumption.




## 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**