

**Please note that Cypress is an Infineon Technologies Company.**

The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

**Continuity of document content**

The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

**Continuity of ordering part numbers**

Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.



# PSoC<sup>®</sup> Creator<sup>™</sup> to ModusToolbox<sup>®</sup>

## Porting Guide

Document Number: 002-25408 Rev \*B

Cypress Semiconductor  
An Infineon Technologies Company  
198 Champion Court  
San Jose, CA 95134-1709  
[www.cypress.com](http://www.cypress.com)  
[www.infineon.com](http://www.infineon.com)

© Cypress Semiconductor Corporation, 2019-2020. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, ModusToolbox, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit [cypress.com](http://cypress.com). Other names and brands may be claimed as property of their respective owners.

# Contents



<b>1. Introduction.....</b>	<b>5</b>
Overview .....	5
Before You Begin .....	5
Terminology.....	6
Expectations.....	6
Revision History .....	6
<b>2. Getting Started.....</b>	<b>7</b>
Create New ModusToolbox Application .....	7
Migrate Pin Layout .....	9
<b>3. System Configuration .....</b>	<b>11</b>
System Clocks / Other Settings.....	11
Analog Routing.....	11
Migrate Interrupts .....	12
<b>4. Porting Components .....</b>	<b>14</b>
Common Component Settings .....	15
Voltage DAC (12-bit) .....	16
Scanning SAR ADC .....	18
Low-Power Comparator.....	21
Analog Reference (AREF).....	23
Programmable Analog.....	24
I2S     25	
PDM-PCM .....	26
SCB     27	
SMIF (QSPI).....	31
TCPWM 32	
MCWDT 33	
RTC     34	
BLE     34	
Emulated EEPROM.....	35
DFU     36	
USB     37	
CapSense.....	38
<b>5. Copying Code .....</b>	<b>39</b>
Importing Your Code .....	39
Building <i>project.h</i> File .....	39

Rewriting <i>main.c</i> File .....	39
Rewriting Component-Specific APIs and Migrating ISRs .....	39

# 1. Introduction



## Overview

Porting a project from PSoC Creator to ModusToolbox is a moderately tedious task and this guide aims to make the process a bit easier. That being said the document *does not cover all the corner cases*. This document is a guide to the porting process and points you towards the right documentation based on your project's needs.

ModusToolbox provides a plethora of features that support our devices' capabilities. Connectivity is one such example. It is also highly customizable and configurable to suit your requirements. Your code can now be IDE independent and device agnostic! Porting your PSoC Creator project to ModusToolbox might open up a whole new world of opportunities and possibilities.

If you have any application specific queries or you run into a problem porting your application, ask a question in the [Cypress Developer Community](#). We constantly strive to make ModusToolbox better to enhance the overall user experience. Your feedback is both welcome and appreciated.

## Before You Begin

This guide expects that you have read the [Eclipse IDE for ModusToolbox Quick Start Guide](#). If not, please do that first.

ModusToolbox is a different product than PSoC Creator. Cypress has taken great care to ensure that it is ready to develop production-quality products. However, ModusToolbox does not yet support all the Components and configuration options that are available in PSoC Creator. As a result, there are PSoC Creator-based designs that cannot be migrated to ModusToolbox 2.1. If you discover that your design cannot be reasonably ported to ModusToolbox, Cypress recommends that you continue to develop with PSoC Creator, which remains a fully supported development tool.

This guide covers only those Components that can be reasonably ported to ModusToolbox. If a Component is not listed in this guide, it likely means it is implemented differently in ModusToolbox. For example, ModusToolbox 2.1 either does **not** support the following, or uses a very different approach:

- **UDB-based content.** Here are some examples of PSoC Creator Components that do not have an alternative implementation in ModusToolbox:
  - ☐ Digital Logic Components
  - ☐ Digital Register Components
  - ☐ Digital Utility Components
- **DSI routing.** Digital signals cannot be routed through the DSI, which reduces the on-chip connectivity options available. This is generally presented as using a non-preferred connection (yellow) in PSoC Creator.
- **Analog multiplexing.** There is no analog MUX implementation in ModusToolbox, and all connectivity choices are set up in startup code but remain static thereafter. The only way to modify connectivity at run-time is through register writes to control AMXUXBUS switches.
- **CapSense.** CapSense in ModusToolbox is a middleware library. It is enabled by default for build support packages (BSPs) that support CapSense. This means that you cannot directly port a CapSense design from PSoC Creator. Instead, you re-implement your CapSense solution using the CapSense middleware APIs. The CapSense configuration settings however can be replicated. See [CapSense](#).

- **Component APIs.** In PSoC Creator, you could either use Component APIs or the PDL (and in some cases mix the two). ModusToolbox does not support any Component APIs, which means you must convert those calls to PDL APIs, or you will need to copy the PSoC Creator generated code.

If your PSoC Creator design makes use of these features, and a firmware-based alternative implementation is not available, then ModusToolbox may not be able to re-create the same functionality enabled by PSoC Creator. Please send your migration questions to the [Cypress Developer Community](#). Cypress wants to hear about your issues, advise where possible, and prioritize features for future releases of ModusToolbox.

## Terminology

PSoC Creator uses the term Components for the various peripherals in a design. ModusToolbox uses the terms peripherals and resources.

In PSoC Creator, you create projects in a workspace; oftentimes referred to as a design. In ModusToolbox, you create applications in a workspace. Applications consist of one or more related projects.

The IDE that is bundled with the ModusToolbox installer is called the Eclipse IDE for ModusToolbox. ModusToolbox is an IDE-independent set of tools that you can use with the IDE that you prefer. It does not require the Eclipse IDE for ModusToolbox. However, this document assumes you are using the Eclipse IDE for ModusToolbox.

## Expectations

It will likely take an advanced PSoC Creator user a day or two to port a moderately complex design that uses the PDL to the ModusToolbox environment. Moderate in this case is defined as a design that uses several Components, with about 500 lines of application code. This time requirement should include porting and debugging the application.

In general, PSoC Creator does many things automatically that you need to do manually in ModusToolbox. Some of these are common items and they are addressed in the common porting section. Other items are Component-specific, and they are addressed in the appropriate section for that Component.

## Revision History

Document Title: PSoC® Creator™ to ModusToolbox® Porting Guide		
Document Number: 002-25408		
Revision	Date	Description of Change
**	1/30/19	New document.
*A	4/1/19	Updated for ModusToolbox 1.1
*B	6/10/20	Updated for ModusToolbox 2.1

## 2. Getting Started



### Create New ModusToolbox Application

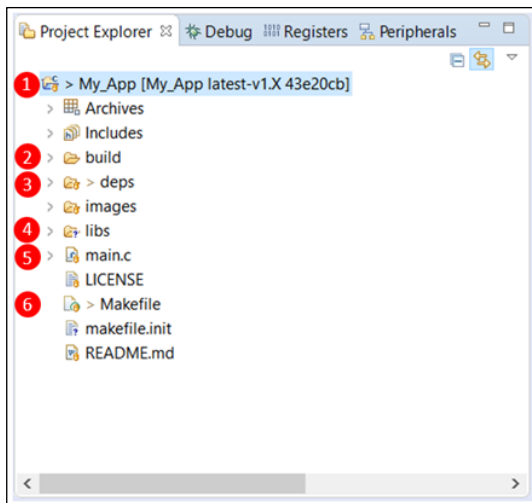
Open the Eclipse IDE for ModusToolbox, select your workspace, and create a new application using the Project Creator tool. Refer to the [Eclipse IDE for ModusToolbox Quick Start Guide](#) for details about the IDE, and refer to the [ModusToolbox Project Creator Guide](#) for details about the Project Creator tool.

When creating the new application:

- Select the BSP for your device. If you are using a custom board, select the BSP that is closest to your device. Refer to the “Creating a BSP for Your Board” section in the [ModusToolbox User Guide](#).
- Select the application that you want to start with. If you want an empty project, you can select the **Empty PSoC6 App**. You can also rename your application. Once this is done, click **Create**.

### A Quick Tour of a ModusToolbox Application

Once you create the application, it is imported into the IDE. Use the Project Explorer to view and open various application files and directories. The following image highlights several important files and directories:



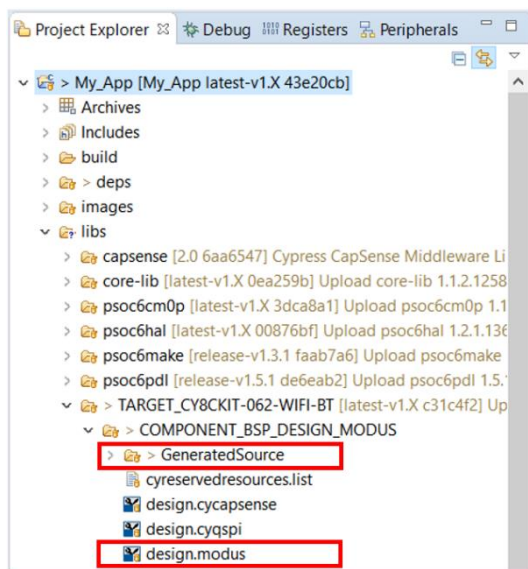
1. This is the application directory that contains all the source code and other files required to successfully build your application. You can have more than one application in a workspace.
2. The **build** directory contains the build artifacts generated after the application is built. The build artifacts for each target are put into separate subdirectories.
3. The **deps** directory contains the *.lib* files for your application. The build system will use these files to include and update BSPs and libraries needed for your application.
4. The **libs** directory stores the imported BSPs and libraries. See [BSP Directory](#) later in this section.



5. Your main code is in the *main.c* file. This has some default calls already filled in, such as initializing the BSP which in turn configures the pins and clocks based on the generated code.
6. The project makefile consists of all the build configuration settings for your project. For more details, refer to the “ModusToolbox Build System” section in the [ModusToolbox User Guide](#). You can also refer to the [Managing the Makefile for ModusToolbox v2.x](#) Knowledge Based Article.

## BSP Directory

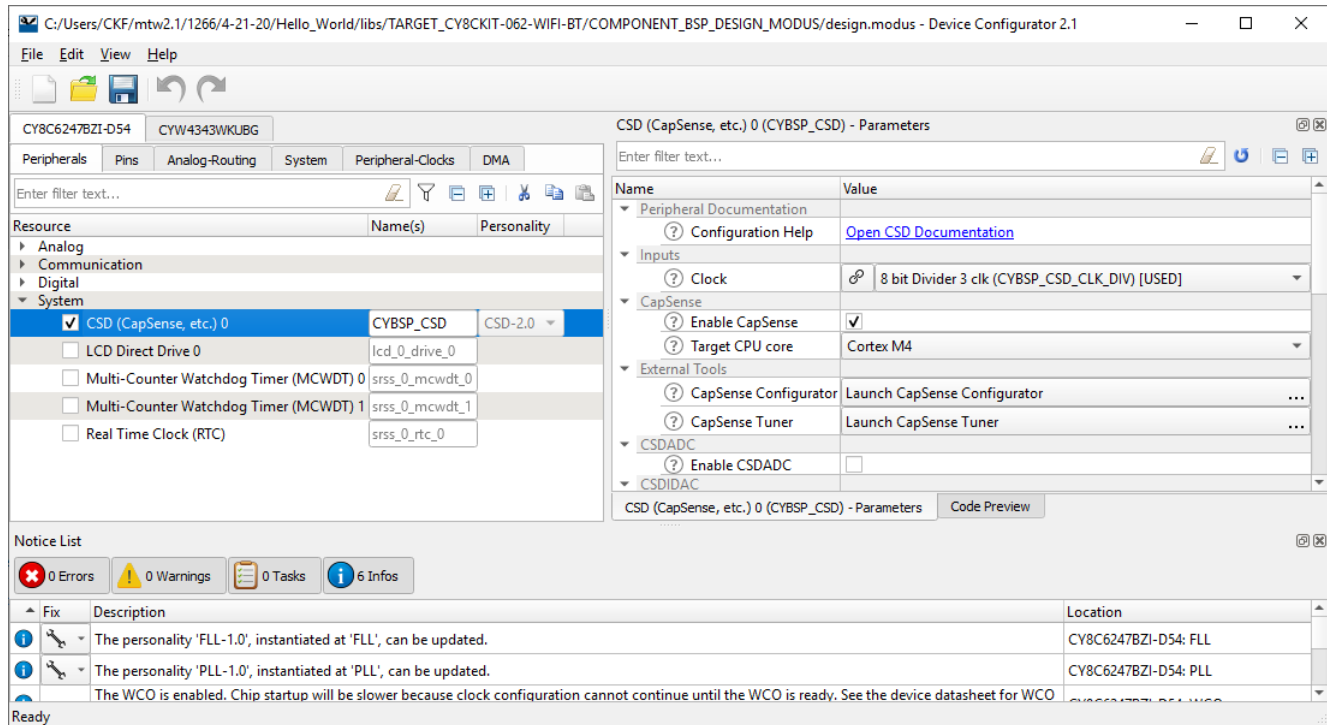
Inside the **libs** directory, there is at least one BSP subdirectory that begins with “TARGET\_”. This subdirectory contains all the necessary files to support the kit and devices. It also contains the *design.modus* file and the Generated Source files in *COMPONENT\_BSP\_DESIGN\_MODUS*, as shown in the following image:



The *design.modus* file is where the Device Configurator stores all the device configuration information for the selected BSP.

## Device Configurator

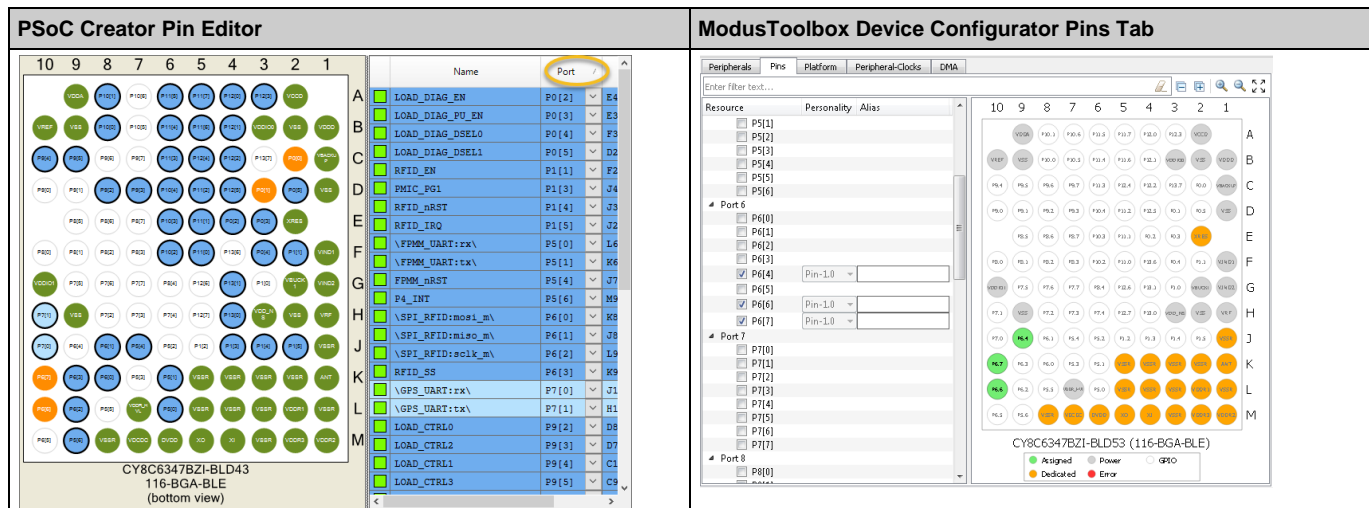
The following image shows the Device Configurator, where you enable and configure peripherals, pins, clocks, etc.



You can access the Device Configurator in the Tools section of Quick Panel in the Eclipse IDE for ModusToolbox. Refer to the [Device Configurator Guide](#) for more details.

## Migrate Pin Layout

Open the Design-Wide Resources (DWR) Pin Editor in PSoC Creator and click on the **Port** heading to sort your pins by port/pin. Then, open the ModusToolbox Device Configurator and select the **Pins** tab. Organize your windows so you can see both at the same time.

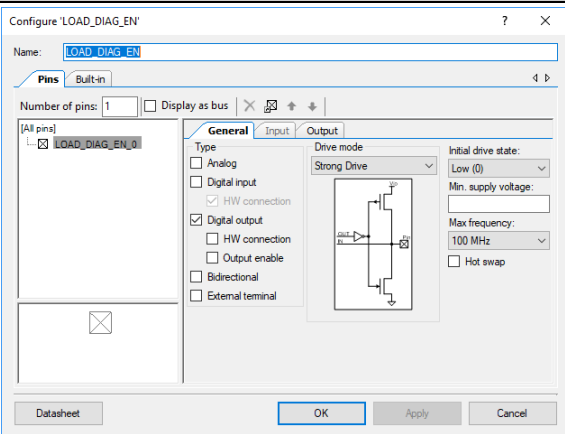
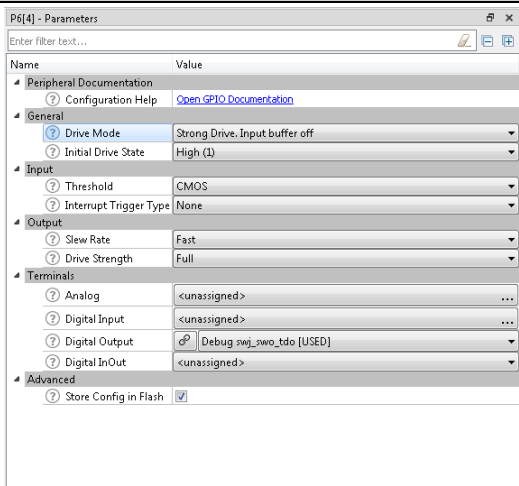


1. Use the PSoC Creator list as a guide for which pins to enable in the Device Configurator.
2. After enabling a pin, edit the alias to match the PSoC Creator name. The names are case sensitive. There are two translations needed while aliasing a pin based on the PSoC Creator name.
  - ☐ If you see a '\ ' in the PSoC Creator name, ignore the pin for the alias.
  - ☐ If you see a '.' in the PSoC Creator name, replace it with an underscore ('\_') in the alias.

3. Once you have all the pins enabled and aliased correctly, it is time to start configuring the pins.

Pins that start with a '\ ' are embedded in a Component. This means you will not be able to find their settings in your schematic. Ignore those for now, once they are connected to their Components, the design rule checks will help you set up those pins.

4. In PSoC Creator, double-click on the name of the first pin. It will take you to the Configure Pins dialog. In this case, the pin is set to strong drive with an initial state of low and it is a digital out.
5. Change the settings in the Device Configurator as appropriate (in this case Strong Drive, input buffer off, with an initial drive state of 0).

PSoC Creator Configure Pins Dialog	Device Configurator Pins Parameters
	

6. Go back to the PSoC Creator DWR Pin Editor and repeat the process with every pin.

## 3. System Configuration



There are various system settings in PSoC Creator that do not apply in ModusToolbox. For example, there is no UDB support in ModusToolbox. However, there are various ModusToolbox system settings available to configure your device.

### System Clocks / Other Settings

Replicate your system clocks, debugging setup, and voltage from PSoC Creator in the Device Configurator **System** tab. This is straightforward and easy to do. The following image shows the **System** tab:

Name	Value
<b>Documentation</b>	
System API Reference	<a href="#">Open SysPm Documentation</a>
<b>General</b>	
System Active Power Mode	LP
Core Regulator	Normal Current LDO
Enable External PMIC Output	<input type="checkbox"/>
vBackup Source	VDDD
<b>RTOS</b>	
System Idle Power Mode	System Deep Sleep
Deep Sleep Latency (ms)	0
<b>Operating Conditions</b>	
VDDA Voltage (mV)	3300

**Notice List**

Fix	Description	Location
	The personality 'FLL-1.0', instantiated at 'FLL', can be updated.	CY8C6247BZI-D54: FLL
	The personality 'PLL-1.0', instantiated at 'PLL', can be updated.	CY8C6247BZI-D54: PLL
	The WCO is enabled. Chip startup will be slower because clock configuration cannot continue until the WCO is ready. See the device datasheet for WCO	CY8C6247BZI-D54: WCO

### Analog Routing

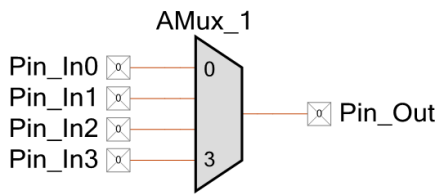
PSoC Creator automatically routes and configures analog switches for all analog connections based on the schematic view and will auto assign resources like pins if not explicitly defined. ModusToolbox also automatically routes analog connections defined for each analog resource. There are two key areas to address when transitioning analog routing from PSoC Creator to ModusToolbox.

For static analog connections in ModusToolbox, choose the specific source or destination from each analog resource personality's Connection Parameters. When the design is built, the route will be configured all the way from the source to the destination. This includes source and destination analog resource switches, AMUXBUS switches, and connection to pins if required. The specific

register settings required to implement the routing can be seen in the *cycfg\_connectivity.c/.h* files. If there is a routing resource conflict because two or more routes requires the same resource the user must deconflict the resource usage.

The analog routing can also be edited manually in ModusToolbox using the **Analog Route Editor** available in the Device Configurator. It also enables you to lock-down all or some of the resource routing. If in case any change that you make should encounter an error the routes are automatically rolled back to the previous state. You can refer to the Analog-Routing Tab section in the [Device Configurator Guide](#) for more information.

Dynamically changing analog connections in PSoC Creator are supported with Analog Mux Components. Analog Mux Components support multiple analog routes dynamically changeable at runtime with PSoC Creator generated function calls. ModusToolbox does not currently support analog multiplexers but user firmware can dynamically change analog routing switches to duplicate the functionality. Refer to the device TRM and analog driver documentation for detailed routing and switch control information. Analog resource switches are supported in their respective PDL drivers. AMUXBUS and GPIO analog switches are supported in the Cy\_GPIO PDL driver.



## Migrate Interrupts

There is no “interrupts” customizer available in ModusToolbox 2.1. All the interrupts used in your PSoC Creator application must be configured through code in ModusToolbox. Please follow these guidelines for porting interrupts.

### Interrupts Internal to Component Code (BLE, CapSense, SCB, etc.)

Refer to the respective Component-specific migration section. Porting involves creating the interrupt structures, proxy interrupt handler, and initializing the interrupt as part of main code. The proxy interrupt handler should call the peripheral driver’s interrupt handler as per the guidelines recommended in the Component specific migration section.

For example, an interrupt handler for I2C SCB calls the SCB driver’s interrupt handler:

```
void I2C_Interrupt(void)
{
    Cy_SCB_I2C_Interrupt(I2C_HW, &I2C_context);
}
```

### Interrupts External to Component Code (GPIO, MCWDT, etc.)

1. Create the interrupt structure in the source files where you initialize the interrupt.

**Note** The interrupt structure generated by the PSoC Creator code can be copied from the *Generated\_Source/PSoC6/Pins and Interrupts/cyfitter\_sysint\_cfg.c* file as well.

2. Use this structure to initialize the interrupt along with the interrupt handler. See the following table for an example (MCWDT\_STRUCT0 interrupt)
3. For the configuration of the interrupt source, refer to the Component migration section.

PSoC Creator	ModusToolbox 2.1
<pre> #include "cyfitter_sysint_cfg.h" #include "mcwdt/cy_mcwdt.h" #include "cy_device_headers.h"  void MCWDT_Interrupt_Handler(void) {     /* Clear the MCWDT peripheral interrupt */      /* Clear the CM4 NVIC pending interrupt for MCWDT */     NVIC_ClearPendingIRQ(MCWDT_isr_cfg.intrSrc);      /* Interrupt processing */ }  void MCWDT_Interrupt_Init(void) {     /* Configure ISR connected to MCWDT interrupt signal*/     /* MCWDT_isr_cfg structure is defined by the SYSINT_PDL component based on parameters entered in the customizer. */     Cy_SysInt_Init(&amp;MCWDT_isr_cfg, &amp;MCWDT_Interrupt_Handler);     /* Clear CM4 NVIC pending interrupt for MCWDT */     NVIC_ClearPendingIRQ(MCWDT_isr_cfg.intrSrc);     /* Enable CM4 NVIC MCWDT interrupt */     NVIC_EnableIRQ(MCWDT_isr_cfg.intrSrc); } </pre>	<pre> #include "cy_pdl.h"  /* MCWDT_isr */ const cy_stc_sysint_t MCWDT_isr_cfg = {     /* IRQn_Type can be found in the device specific (cy8c6xxxxxx_xxxxx.h) header file */     .intrSrc =     (IRQn_Type)srss_interrupt_mcwdt_0_IRQn,     .intrPriority = 7u };  void MCWDT_Interrupt_Handler(void) {     /* Clear the MCWDT peripheral interrupt */      /* Clear the CM4 NVIC pending interrupt for MCWDT */     NVIC_ClearPendingIRQ(MCWDT_isr_cfg.intrSrc);      /* Interrupt processing */ }  void MCWDT_Interrupt_Init(void) {     Cy_SysInt_Init(&amp;MCWDT_isr_cfg, &amp;MCWDT_Interrupt_Handler);     /* Clear CM4 NVIC pending interrupt for MCWDT */     NVIC_ClearPendingIRQ(MCWDT_isr_cfg.intrSrc);     /* Enable CM4 NVIC MCWDT interrupt */     NVIC_EnableIRQ(MCWDT_isr_cfg.intrSrc); } </pre>

## Deprecated Functions

Note that some of the functions used as part of sysint PDL driver (v1.10 or older) have been deprecated and replaced with new functions in the sysint v1.20 (or later). Code that uses the deprecated functions will still build and function. The deprecated functions have been supplied as macros that call their new counterparts. It is recommended that you update your code to use the new functions.

Deprecated functions (SysInt v1.10 or older)	New functions (SysInt v1.20 or later)
Cy_SysInt_GetState	N/A – Invokes call to NVIC_GetEnableIRQ
Cy_SysInt_SetIntSource	Cy_SysInt_SetInterruptSource
Cy_SysInt_GetIntSource	Cy_SysInt_GetInterruptSource
Cy_SysInt_SetIntSourceNMI	Cy_SysInt_SetNmiSource
Cy_SysInt_GetIntSourceNMI	Cy_SysInt_GetNmiSource

## 4. Porting Components



This chapter covers various PSoC Creator Components that can be ported to ModusToolbox, as well as common Component settings. If a Component is not contained in this chapter, your PSoC Creator design cannot be completely ported. This chapter covers the following:

- [Common Component Settings](#)
- [Voltage DAC \(12-bit\)](#)
- [Scanning SAR ADC](#)
- [Low-Power Comparator](#)
- [Analog Reference \(AREF\)](#)
- [Programmable Analog](#)
- [I2S](#)
- [PDM-PCM](#)
- [SCB](#)
- [SMIF \(QSPI\)](#)
- [TCPWM](#)
- [MCWDT](#)
- [RTC](#)
- [BLE](#)
- [Emulated EEPROM](#)
- [DFU](#)
- [USB](#)
- [CapSense](#)

## Common Component Settings

### Enable Component Clocks

Typically, just pick an unused clock in the Device Configurator. The correct clock rate should be set automatically the first time it is enabled. If you make changes to a peripheral, you may need to update the clock settings. In some cases, a warning will be generated if you use an 8-bit clock when you need a 32-bit clock to achieve the required bit rate. In those cases, change the clock used for the hardware block.

### Assign Pins

The Device Configurator has parameters for various connections. In some cases, you'll have check boxes for both input and output connections on a pin. You only need to select one input. When you do, a new notice may be added telling you how you need to configure the pin. Resolve these errors as needed.

### Resolve Configurator Errors, Warnings and Tasks

You can save the file, but code is not generated until you resolve all errors. You should also resolve any warnings/tasks in the notice list of the Device Configurator. The resulting code is generated automatically.

### Aliases

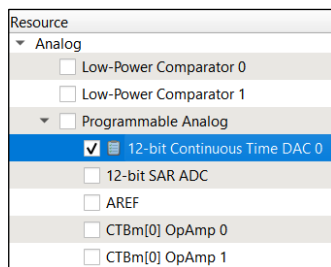
Just like pins, set up the alias for your peripherals so that the names match your PSoC Creator Components.



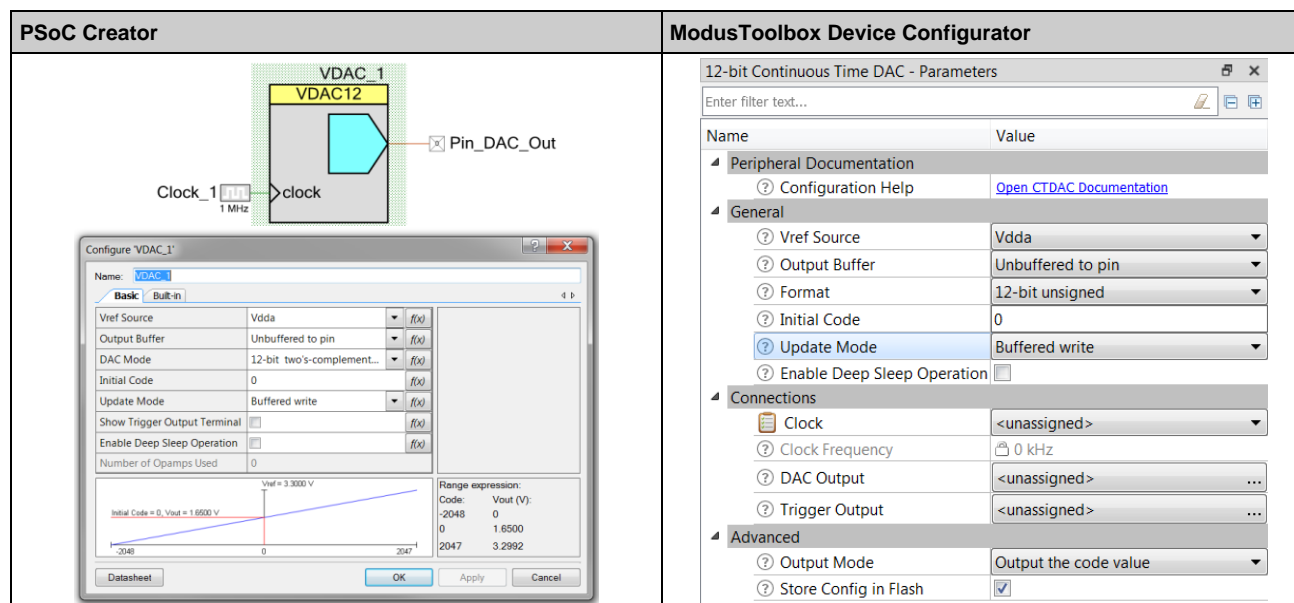
## Voltage DAC (12-bit)

Follow these steps to transition the PSoC Creator Voltage DAC (12-bit) to ModusToolbox:

1. Enable the **12-bit Continuous Time DAC** in the ModusToolbox Device Configurator.



2. Transition the PSoC Creator Component parameters to the ModusToolbox 12-bit Continuous Time DAC parameters as follows:



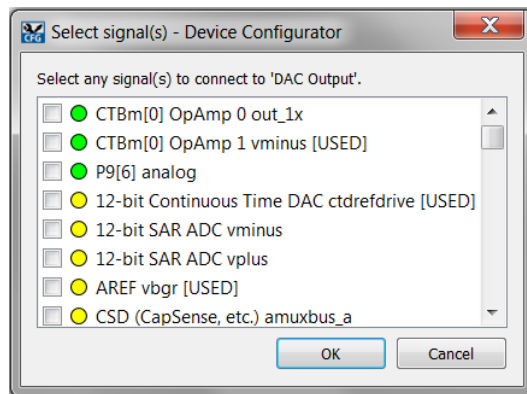
- a. The **Vref Source** value [**Analog reference**] requires two parameter changes in the 12-bit Continuous Time DAC as well as configuration of CTBm[0] OpAmp 1 to route and buffer the internal reference.

First set **Vref Source** to **External** and then select **Reference Source** as **CTBm[0] OpAmp 1 out\_1x**. Enable **CTBm[0] OpAmp 1** and set Personality to OpAmp-[version].

Let all parameters be default with the exception of the following changes:

- Set **Vplus Input** to **AREF vref**,
  - **Vminus Input** to **CTBm[0] OpAmp 1 out\_1x**,
  - **Output (internal only)** to **12-bit Continuous Time DAC ctdrefdrive**.
- b. The **Output Buffer** values [**Buffered to pin**, **Buffered internal**] convert to **Output Buffer[Buffered]**. The differences between **Buffered to pin** and **Buffered internal** are handled in the DAC output parameter based on where the output is routed.
- c. **DAC Mode** has been renamed to **Format**.

- d. **Sample and Hold** (if visible)
  - e. **Initial Code**
  - f. The **Update Mode** values [**Strobe edge sync**, **Strobe edge immediate**, **Strobe level**] are not supported as they require DSI routing to connect to the strobe source. ModusToolbox does not currently support DSI routing. One workaround for a fixed frequency strobe is to set **Update Mode** to **Buffered write** and select a clock connection source configured for the required strobe rate.
  - g. **Show Trigger Output Terminal** is not transferred because it only controls wire terminal visibility in PSoC Creators schematic view.
  - h. **Enable Deep Sleep Operation**
3. PSoC Creator automatically routes the output using schematic entry. In ModusToolbox, the output is routed using **Connections** parameters. From the **DAC Output** drop down connection list, select the direct pin CTBm OpAmp input connections (green circles) or another connection routed through other resources (yellow circles).



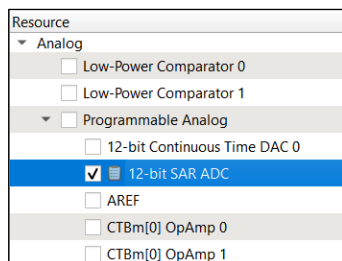
If the **Output Buffer** parameter is set to **Buffered**, **CTBm[0] OpAmp 0 Vplus** must be selected as the output destination. The dedicated output pin is device specific and can be referenced in the device datasheet.

4. The **trigger** output in PSoC Creator is automatically routed using schematic entry. In ModusToolbox, all connections other than interrupts are selected from the **Trigger Output** parameter drop-down. Interrupt connections are fixed and already made to the CPU interrupt controller. See the [Interrupts section](#) for details on transitioning the interrupt, if present.

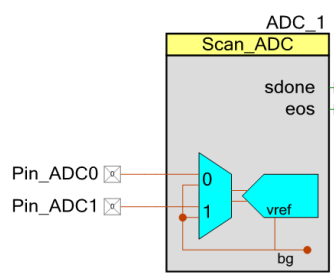
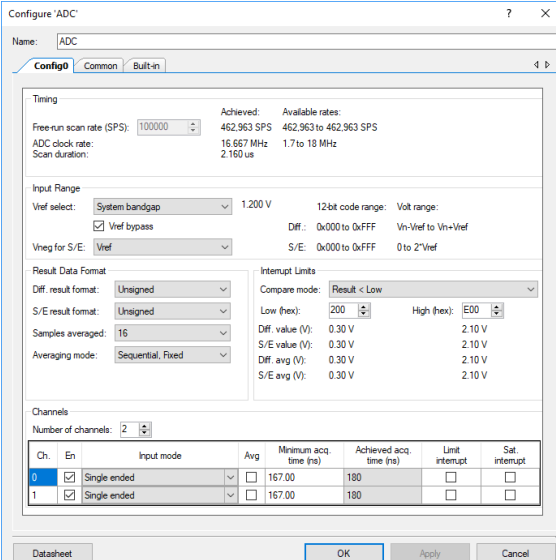
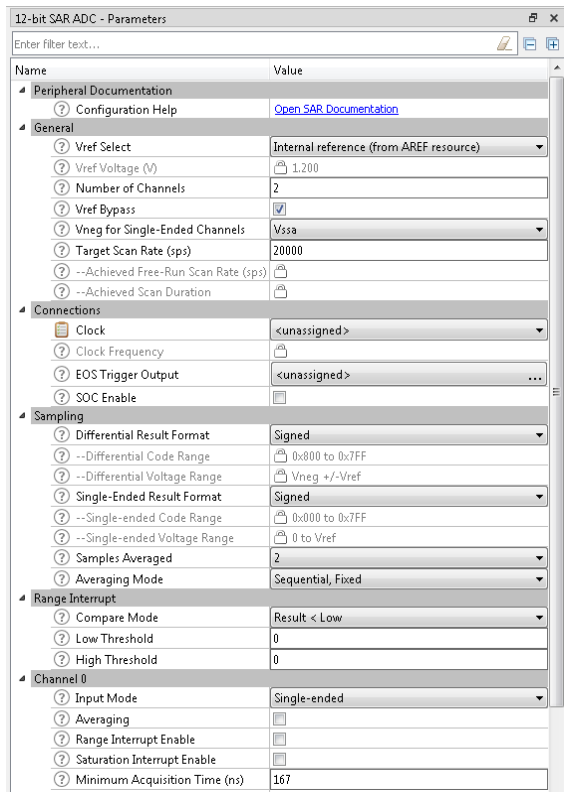
## Scanning SAR ADC

Follow these steps to transition the PSoC Creator Scanning SAR ADC to ModusToolbox:

1. Enable the **12-bit SAR ADC** in the ModusToolbox Device Configurator:



2. Transition the PSoC Creator **Config0** tab Component parameters to the ModusToolbox 12-bit SAR ADC parameters as follows:

PSoC Creator	ModusToolbox Device Configurator
 	

- a. **Vref Select**
- b. **Number of Channels**
- c. **Vref Bypass**
- d. The **Vneg for Single-Ended Channels** (**Vneg for S/E** in PSoC Creator) parameter directly transfers values **Vssa** and **Vref** to ModusToolbox. The PSoC Creator **External** option for **Vneg for S/E** converts to **External P1/P3/P5/P7** options for **Vneg for Single-Ended Channels** parameter in ModusToolbox depending on the specific pin chosen for your design.

PSoC Creator determines the specific pin automatically based on the schematic view. Refer to the Pin Editor in PSoC Creator to determine the I/O pin used.

- e. PSoC Creator **Free-run scan rate (SPS)** converts to ModusToolbox **Target Scan Rate (sps)**. PSoC Creator automatically selects and configures a clock source to generate the scan rate. In ModusToolbox you must select a Clock Connection and configure its source and divider to generate the frequency required for the scan rate. See the [SAR PDL documentation](#) for details on determining the correct clock frequency.
- f. The ModusToolbox parameter **Hardware Trigger Mode**, which is enabled once the **SOC Enable** parameter is checked, is derived from the PSoC Creator parameter **Sample Mode** located on the **Common** tab. If **Sample Mode = Continuous** then **Hardware Trigger Mode = Level sensitive**. If **Sample Mode = Single Shot** then **Hardware Trigger Mode = Edge sensitive**.
- g. **Differential Result Format**
- h. **Single-Ended Result Format**
- i. **Samples Averaged**
- j. **Averaging Mode**
- k. **Compare Mode**
- l. **Low Threshold**
- m. **High Threshold**

3. For each ADC input channel, the following parameters are transferred from PSoC Creator to ModusToolbox:

- ☐ **Input mode -> Input Mode**
- ☐ **Avg -> Averaging**
- ☐ **Limit Interrupt -> Range Interrupt Enable**
- ☐ **Sat. interrupt -> Saturation Interrupt Enable**
- ☐ **Minimum acq. Time (ns) -> Minimum Acquisition Time (ns)**

The **En** check box in PSoC Creator is not supported in ModusToolbox as all channels are automatically enabled. Dynamic control of channel enables can be performed using PDL driver supplied APIs.

4. Transition the PSoC Creator **Common** tab Component parameters.

- a. The **Show analog clock (aclk) terminal** check box does not transfer to ModusToolbox because the analog clock must always be manually configured.
- b. The Parameter **Number of configs** is not supported in the ModusToolbox version of the SAR ADC. If more than one configuration is required, they must be configured in firmware. Multiple configurations are created by manually generating multiple PDL initialization structures as detailed in the PDL driver library documentation.
- c. The Parameter **Sample Mode** is not in the ModusToolbox parameters list. This setting has been moved to an argument passed when calling the ADC start function provided in the PDL library.
- d. The PSoC Creator **Use soc terminal** parameter converts to ModusToolbox **SOC Enable**. The **SOC Input** parameter can then select the source of the SOC signal.

- PSoC Creator automatically routes the output using schematic entry. In ModusToolbox the output is routed using Connections Parameters. The **Ch[n] Vplus**, and **Ch[n] Vminus** (if **Input Mode** is **Differential**) Parameters in ModusToolbox must select the desired analog pin connection for the channel. The dedicated input pins are device specific and can be referenced in the device datasheet.

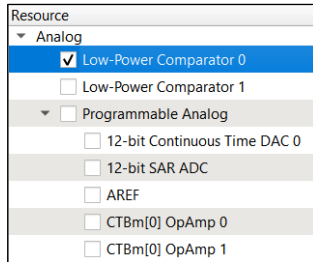
Ch0 Vplus	<unassigned>
Channel 1	P10[0] analog
Input Mode	P10[1] analog
Averaging	P10[2] analog
Range Interrupt Enable	P10[3] analog
Saturation Interrupt Enable	P10[4] analog
Minimum Acquisition Time (ns)	

- The **eos** output in PSoC Creator is automatically routed using schematic entry. In ModusToolbox all connections other than interrupts are selected from the **EOS Trigger Output** parameter dropdown. Interrupt connections are fixed and already made to the CPU interrupt controller. Please see the Interrupts section for details on transitioning the interrupt if present.
- The PSoC Creator **sdone** signal is not supported as it requires DSI routing to connect to the sdone source. ModusToolbox does not currently support DSI routing.

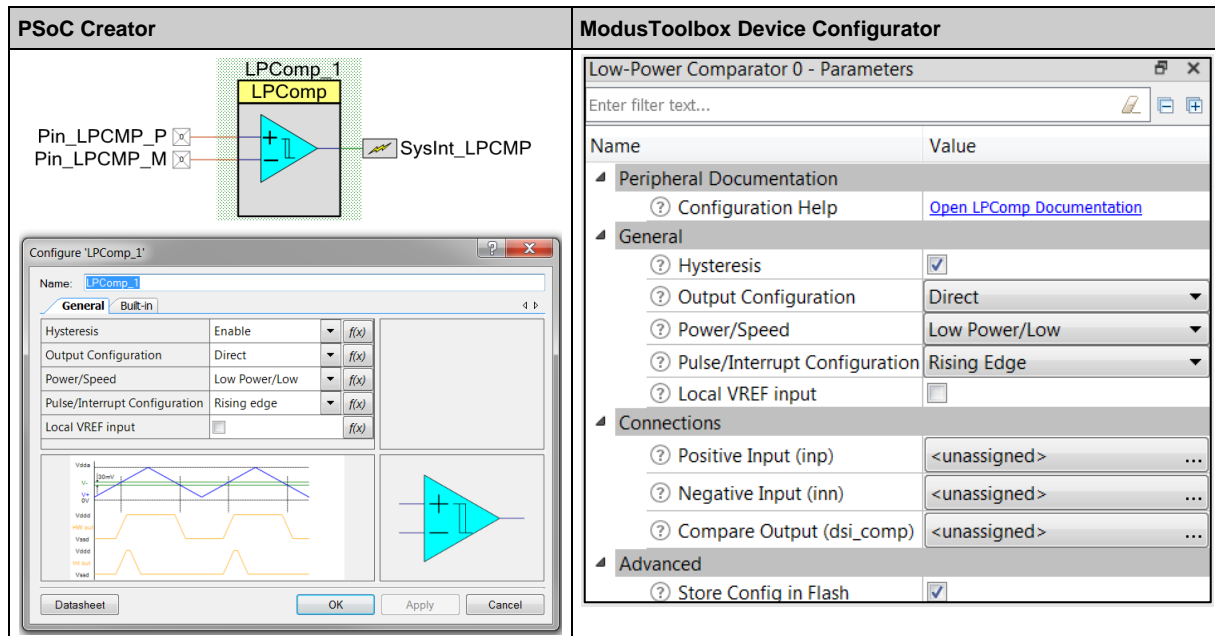
## Low-Power Comparator

Follow these steps to transition the PSoC Creator Low Power Comparator to ModusToolbox:

1. Enable the **Low-Power Comparator 0/1** in the ModusToolbox Device Configurator as follows:

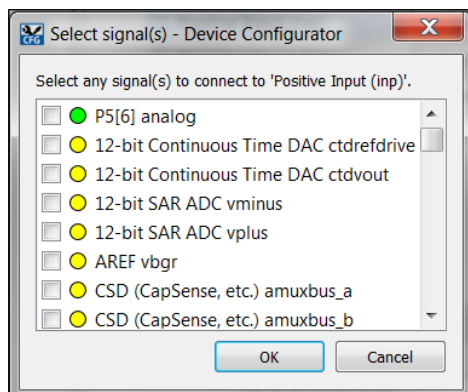


2. Transition the PSoC Creator Component parameters to the ModusToolbox Low-Power Comparator Parameters as follows:



- a. Hysteresis
- b. Output Configuration
- c. Power/Speed
- d. Pulse/Interrupt Configuration
- e. Local VREF input

3. PSoC Creator automatically routes the inputs using schematic entry. In ModusToolbox the **Positive Input** and **Negative Inputs** are routed using **Connections** Parameters. From the **Positive** or **Negative Input** drop-down connection list, select the direct pin connections (green circles) or another connection routed through AMUXBUSA or AMUXBUSB (yellow circles). Input pins are device specific and can be referenced in the device datasheet.

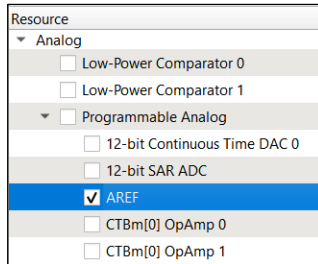


4. The comparator digital output in PSoC Creator is automatically routed using schematic entry. In ModusToolbox all connections other than interrupts are selected from the **Compare Output (dsi\_comp)** parameter dropdown. Interrupt connections are fixed and already made to the CPU interrupt controller. Please see the Interrupts section for details on transitioning the interrupt if present.

## Analog Reference (AREF)

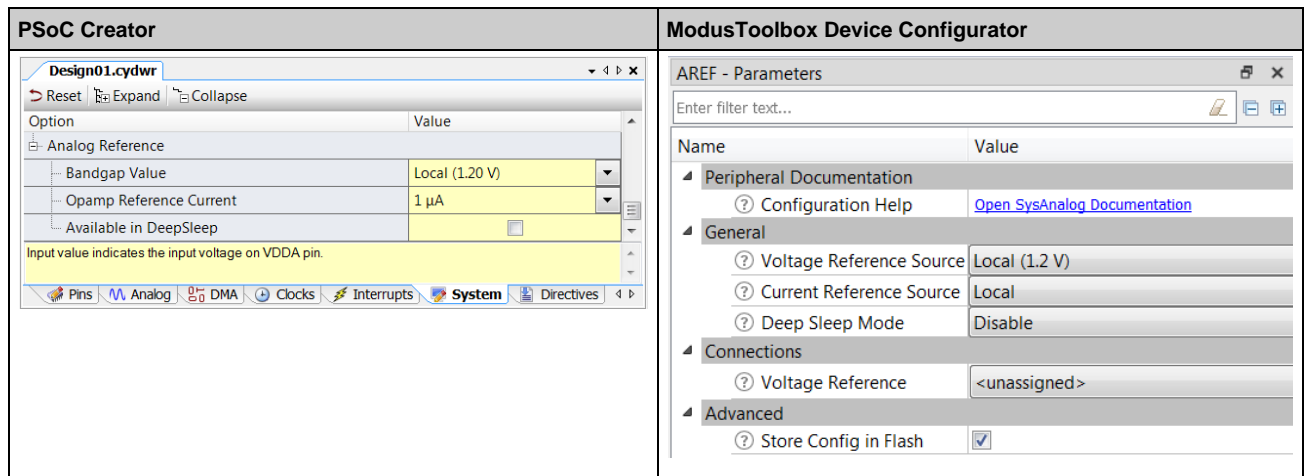
Follow these steps to transition the PSoC Creator Programmable Analog settings to ModusToolbox:

1. All designs that use any analog resource must enable the Analog Reference resource.



2. In a PSoC Creator design, the analog reference settings are in the Design-Wide Resources (DWR) **System** tab, in the **Analog Reference** section. The PSoC Creator **Bandgap Value** transitions to ModusToolbox **Voltage Reference Source**.

**Current Reference Source**, **Deep Sleep Mode**, and **Voltage Reference** ModusToolbox parameters are not provided in PSoC Creator designs as they default to the values shown in ModusToolbox. These settings should typically remain in their default state.

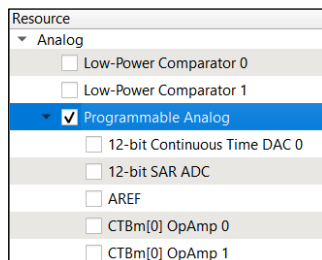




## Programmable Analog

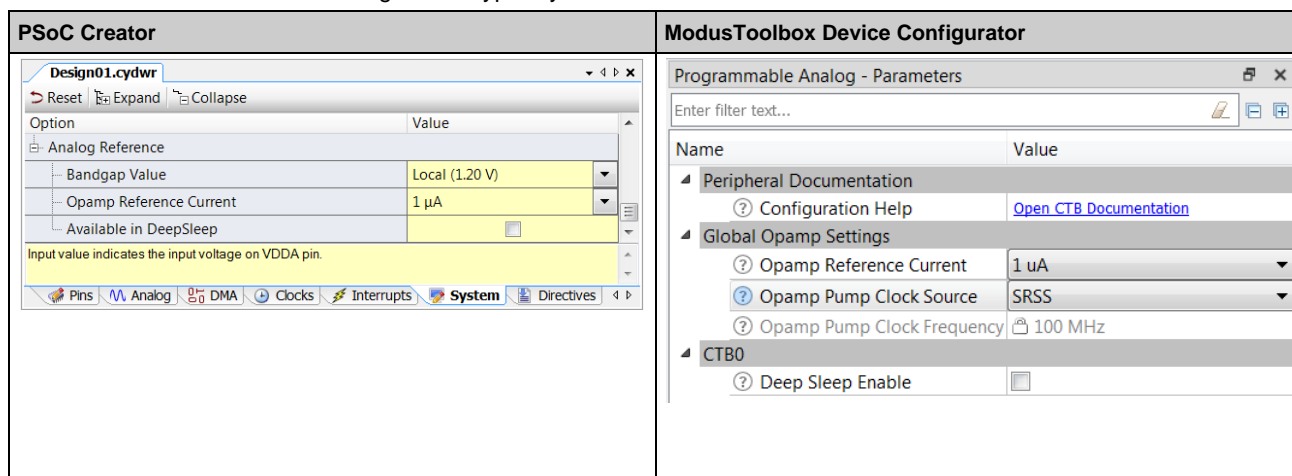
Follow these steps to transition the PSoC Creator Programmable Analog settings to ModusToolbox:

1. All designs that use any analog resource under Programmable Analog section must enable the Programmable Analog resource.



2. In a PSoC Creator design, the analog reference settings are in the DWR **System** tab, in the **Analog References** section. The PSoC Creator **Opamp Reference Current** transitions to ModusToolbox **Opamp Reference Current**. **Available in Deep Sleep** parameter transitions to **Deep Sleep Enable** in ModusToolbox Toolbox.

**Opamp Pump Clock Source** ModusToolbox parameter is not provided in PSoC Creator designs as it defaults to the value shown in ModusToolbox. This setting should typically remain in its default state.



## I2S

Replicate the settings as appropriate.

### PSoC Creator

Configure 'I2S\_PDL\_v2\_0'

Name: I2S

**Basic** Built-in

☐ Clock from terminal Input clock frequency (kHz): Unknown

Clock divider: 16

☒ TX

Mode: Master

Bit rate (kbps): Unknown

Alignment: I2S mode

Channels: 2

Channel length: 16

Word length: 16

Frame rate (kps): Unknown

Overhead value: 0

WS pulse width: 1 channel length

SDO latching time: Normal

Output clock polarity: Normal

FIFO trigger level: 0

☐ DMA trigger

Interrupts:

☐ Trigger

☐ Not full

☐ Empty

☐ Overflow

☐ Underflow

☐ Watchdog: FFFFFFFF

☐ RX

Mode: Master

Bit rate (kbps): Unknown

Alignment: I2S mode

Channels: 2

Channel length: 16

Word length: 16

Frame rate (kps): Unknown

Sign extension: 0

WS pulse width: 1 channel length

SDI latching time: Normal

Output clock polarity: Normal

FIFO trigger level: 0

☐ DMA trigger

Interrupts:

☐ Trigger

☐ Not empty

☐ Full

☐ Overflow

☐ Underflow

☐ Watchdog: FFFFFFFF

Datasheet OK Apply Cancel

### ModusToolbox Device Configurator

Inter-IC Sound Bus (I2S) 0 - Parameters

Enter filter text...

Name	Value
Peripheral Documentation	
Configuration Help	<a href="#">Open I2S Documentation</a>
Timing	
Clock	CLK_HF1 root_clk [USED]
External Interface Clock	<unassigned>
Input Clock Frequency (kHz)	100000.000
Clock Divider	16
Bit Rate (kbps)	781.250
TX	
Enable	<input checked="" type="checkbox"/>
Mode	Master
Alignment	I2S mode
Channels	2
Channel Length	16
Word Length	16
Frame Rate (kps)	24.414
Overhead Value	0
WS Pulse Width	1 channel length
SDO Latching Time	Normal
Input Serial Clock Polarity	Normal
Output Serial Clock Polarity	Normal
FIFO Trigger Level	0
DMA Trigger	<input type="checkbox"/>
Watchdog Enable	<input type="checkbox"/>
RX	
Enable	<input type="checkbox"/>
Outputs	
Advanced	

## PDM-PCM

Replicate the settings as appropriate.

### PSoC Creator

Configure 'PDM\_PCM'

Name: PDM\_PCM

**Basic** Built-in

<b>Channels</b>		
Channel Recording Swap	<input type="checkbox"/>	fix
Left Channel Gain	0dB	fix
Right Channel Gain	0dB	fix
Stereo / Mono Mode Select	Stereo	fix
<b>Filter</b>		
Disable High Pass Filter	<input type="checkbox"/>	fix
High Pass Filter Gain	8	fix
<b>Interrupts</b>		
RX FIFO is Not Empty Interrupt	<input type="checkbox"/>	fix
RX FIFO Overflow Interrupts	<input type="checkbox"/>	fix
RX FIFO Trigger Interrupts	<input type="checkbox"/>	fix
RX FIFO Underflow Interrupts	<input type="checkbox"/>	fix
<b>Output Data</b>		
Output Data Sign Extension	<input type="checkbox"/>	fix
Output Data Word Length, in Bits	24	fix
<b>Output FIFO</b>		
DMA Trigger Enable	<input checked="" type="checkbox"/>	fix
Output FIFO Trigger Level	0	fix
<b>Soft Mute</b>		
Enable Soft Mute	<input type="checkbox"/>	fix
Select Soft Mute Fine Gain	0.26dB	fix
Soft Mute Cycles	96	fix
<b>Timing</b>		
1st Clock Divisor	1/4	fix
2nd Clock Divisor	1/1	fix
3rd Clock Divisor	3	fix
Number of PDM_CLK Periods	0	fix
Sinc Decimation Rate	32	fix

Datasheet OK Apply Cancel

### ModusToolbox Device Configurator

PDM-PCM Converter 0 - Parameters

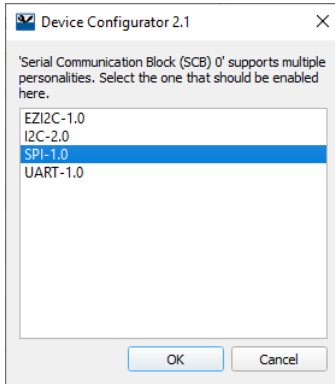
Enter filter text...

Name	Value
<b>Peripheral Documentation</b>	
① Configuration Help	<a href="#">Open PDM_PCM Documentation</a>
<b>Channels</b>	
① Channel Recording Swap	<input type="checkbox"/>
① Left Channel Gain	0dB
① Right Channel Gain	0dB
① Stereo / Mono Mode Select	Stereo
<b>Filter</b>	
① Disable High Pass Filter	<input type="checkbox"/>
① High Pass Filter Gain	0
<b>Interrupts</b>	
① Not Empty	<input type="checkbox"/>
① Overflow	<input type="checkbox"/>
① Trigger	<input type="checkbox"/>
① Underflow	<input type="checkbox"/>
<b>Output Data</b>	
① Output Data Sign Extension	<input type="checkbox"/>
① Output Data Word Length, in Bits	16
<b>Output FIFO</b>	
① DMA Trigger Enable	<input type="checkbox"/>
① Output FIFO Trigger Level	0
<b>Soft Mute</b>	
① Enable Soft Mute	<input type="checkbox"/>
① Select Soft Mute Fine Gain	0.26dB
① Soft Mute Cycles	96
<b>Timing</b>	
① Clock	CLK_HF1 root_clk [USED]
① 1st Clock Divisor	1/1
① 2nd Clock Divisor	1/1
① 3rd Clock Divisor	3
① Number of PDM_CLK Periods	0
① Sinc Decimation Rate	32
<b>Inputs</b>	
① PDM Data	<unassigned>
<b>Outputs</b>	
<b>Advanced</b>	

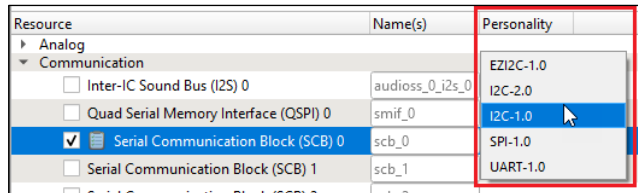
## SCB

### Personalities

For the SCB block, there are several personalities available in the Device Configurator: EZI2C, I2C, SPI, and UART. When you enable an SCB block, a dialog displays to select the appropriate personality:



Once enabled, you can change the personality from the drop-down next to the SCB name.



### How Do I Choose the Correct SCB?

SCBs have fixed connectivity to specific pins. In PSoC Creator, the tool selects the SCB automatically based on pin selection. In ModusToolbox, you must select the correct SCB manually, based on your pins. To determine the correct SCB, find the pin and click on the **Digital InOut** parameter under Internal Connection in the parameters window.

Internal Connection	
ⓘ Analog	<unassigned> ...
ⓘ Digital Input	<unassigned> ...
ⓘ Digital Output	<unassigned> ▼
ⓘ Digital InOut	<unassigned>
Advanced	
ⓘ Store Config in Flash	<ul style="list-style-type: none"> <li>Serial Communication Block (SCB) 1 I2C.scl [ENABLED]</li> <li>Serial Communication Block (SCB) 1 SPI.mosi [ENABLED]</li> <li>Serial Communication Block (SCB) 1 UART.rx [ENABLED]</li> </ul>

This shows you which SCB block(s) connect to this pin. Look at all the pins you've assigned and start with the most constrained pin. If that pin connects to only one SCB, it is obvious which one you should enable. If that pin connects to more than one SCB, you must create a list of the SCBs to which it connects and go to the next most constrained pin. Doing so, you should be able to reduce your list to a definitive list of SCBs that work for your design.

In PSoC Creator, connecting and configuring clocks was automatic. In ModusToolbox, you specify clock and configuration manually. Refer to the [Device Configurator Guide](#) for more details regarding peripheral clock settings.

## EZI2C

Replicate the settings as appropriate.

PSoC Creator	ModusToolbox Device Configurator																														
<p>Configure 'EZI2C_1'</p> <p>Name: <input type="text" value="EZI2C_1"/></p> <p><b>Basic</b> Pins Built-in</p> <p><b>Clock Source</b></p> <p>Enable Clock from Terminal <input type="checkbox"/> f(x)</p> <p><b>General</b></p> <p>Data Rate (kbps) 100 f(x)</p> <p>Number of Addresses 1 f(x)</p> <p>Primary Slave Address (7-bit) 0x08 f(x)</p> <p>Sub-Address Size 8 bits f(x)</p> <p>Enable Wakeup from Deep Sleep Mode <input type="checkbox"/> f(x)</p> <p>Actual data rate (kbps): 400</p> <p>SCB clock (kHz): 12500</p> <p>Datasheet OK Apply Cancel</p>	<p>Serial Communication Block (SCB) 0 - Parameters</p> <p>Enter filter text...</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td colspan="2"><b>Peripheral Documentation</b></td> </tr> <tr> <td>Configuration Help</td> <td><a href="#">Open EZI2C (SCB) Documentation</a></td> </tr> <tr> <td colspan="2"><b>General</b></td> </tr> <tr> <td>Data Rate (kbps)</td> <td>100</td> </tr> <tr> <td>Number of Addresses</td> <td>1</td> </tr> <tr> <td>Primary Slave Address (7-bit)</td> <td>8</td> </tr> <tr> <td>Sub-Address Size</td> <td>8 bits</td> </tr> <tr> <td colspan="2"><b>Connections</b></td> </tr> <tr> <td>Clock</td> <td>&lt;unassigned&gt;</td> </tr> <tr> <td>Clock Frequency</td> <td></td> </tr> <tr> <td>SCL</td> <td>P0[2] digital_inout [USED]</td> </tr> <tr> <td>SDA</td> <td>P0[3] digital_inout [USED]</td> </tr> <tr> <td colspan="2"><b>Advanced</b></td> </tr> <tr> <td>Store Config in Flash</td> <td><input checked="" type="checkbox"/></td> </tr> </tbody> </table>	Name	Value	<b>Peripheral Documentation</b>		Configuration Help	<a href="#">Open EZI2C (SCB) Documentation</a>	<b>General</b>		Data Rate (kbps)	100	Number of Addresses	1	Primary Slave Address (7-bit)	8	Sub-Address Size	8 bits	<b>Connections</b>		Clock	<unassigned>	Clock Frequency		SCL	P0[2] digital_inout [USED]	SDA	P0[3] digital_inout [USED]	<b>Advanced</b>		Store Config in Flash	<input checked="" type="checkbox"/>
Name	Value																														
<b>Peripheral Documentation</b>																															
Configuration Help	<a href="#">Open EZI2C (SCB) Documentation</a>																														
<b>General</b>																															
Data Rate (kbps)	100																														
Number of Addresses	1																														
Primary Slave Address (7-bit)	8																														
Sub-Address Size	8 bits																														
<b>Connections</b>																															
Clock	<unassigned>																														
Clock Frequency																															
SCL	P0[2] digital_inout [USED]																														
SDA	P0[3] digital_inout [USED]																														
<b>Advanced</b>																															
Store Config in Flash	<input checked="" type="checkbox"/>																														

## I2C

Set the mode as needed to match PSoC Creator and replicate the settings as appropriate.

PSoC Creator	ModusToolbox Device Configurator																																														
<p>Configure 'I2Cm1'</p> <p>Name: <input type="text" value="I2Cm1"/></p> <p><b>Basic</b> Pins Built-in</p> <p><b>Clock Source</b></p> <p>Enable Clock from Terminal <input type="checkbox"/> f(x)</p> <p><b>General</b></p> <p>Mode Master</p> <p>Data Rate (kbps) 100 f(x)</p> <p>Use TX FIFO <input type="checkbox"/> f(x)</p> <p>Use RX FIFO <input type="checkbox"/> f(x)</p> <p><b>Master</b></p> <p>Enable Manual SCL Control <input type="checkbox"/> f(x)</p> <p>Actual data rate (kbps): 97</p> <p>SCB clock (kHz): 1562.5</p> <p>tLow (us): 5120</p> <p>tHigh (us): 5120</p> <p>Datasheet OK Apply Cancel</p>	<p>Serial Communication Block (SCB) 2 - Parameters</p> <p>Enter filter text...</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td colspan="2"><b>Peripheral Documentation</b></td> </tr> <tr> <td>Configuration Help</td> <td><a href="#">Open I2C (SCB) Documentation</a></td> </tr> <tr> <td colspan="2"><b>General</b></td> </tr> <tr> <td>Mode</td> <td>Slave</td> </tr> <tr> <td>Data Rate (kbps)</td> <td>100</td> </tr> <tr> <td>Use TX FIFO</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>Accept Matching Address in RX FIFO</td> <td><input type="checkbox"/></td> </tr> <tr> <td>Use RX FIFO</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td colspan="2"><b>Slave</b></td> </tr> <tr> <td>Slave Address (7-bit)</td> <td>8</td> </tr> <tr> <td>Slave Address Mask (8-bit)</td> <td>254</td> </tr> <tr> <td>Accept General Call Address</td> <td><input type="checkbox"/></td> </tr> <tr> <td colspan="2"><b>Connections</b></td> </tr> <tr> <td>Clock</td> <td>&lt;unassigned&gt;</td> </tr> <tr> <td>SCL</td> <td>&lt;unassigned&gt;</td> </tr> <tr> <td>SDA</td> <td>&lt;unassigned&gt;</td> </tr> <tr> <td>SCL Output (scl_trig)</td> <td>&lt;unassigned&gt;</td> </tr> <tr> <td colspan="2"><b>Actual Data Rate</b></td> </tr> <tr> <td>Actual Data Rate (kbps)</td> <td>100</td> </tr> <tr> <td>Clock Frequency</td> <td></td> </tr> <tr> <td colspan="2"><b>Advanced</b></td> </tr> <tr> <td>Store Config in Flash</td> <td><input checked="" type="checkbox"/></td> </tr> </tbody> </table>	Name	Value	<b>Peripheral Documentation</b>		Configuration Help	<a href="#">Open I2C (SCB) Documentation</a>	<b>General</b>		Mode	Slave	Data Rate (kbps)	100	Use TX FIFO	<input checked="" type="checkbox"/>	Accept Matching Address in RX FIFO	<input type="checkbox"/>	Use RX FIFO	<input checked="" type="checkbox"/>	<b>Slave</b>		Slave Address (7-bit)	8	Slave Address Mask (8-bit)	254	Accept General Call Address	<input type="checkbox"/>	<b>Connections</b>		Clock	<unassigned>	SCL	<unassigned>	SDA	<unassigned>	SCL Output (scl_trig)	<unassigned>	<b>Actual Data Rate</b>		Actual Data Rate (kbps)	100	Clock Frequency		<b>Advanced</b>		Store Config in Flash	<input checked="" type="checkbox"/>
Name	Value																																														
<b>Peripheral Documentation</b>																																															
Configuration Help	<a href="#">Open I2C (SCB) Documentation</a>																																														
<b>General</b>																																															
Mode	Slave																																														
Data Rate (kbps)	100																																														
Use TX FIFO	<input checked="" type="checkbox"/>																																														
Accept Matching Address in RX FIFO	<input type="checkbox"/>																																														
Use RX FIFO	<input checked="" type="checkbox"/>																																														
<b>Slave</b>																																															
Slave Address (7-bit)	8																																														
Slave Address Mask (8-bit)	254																																														
Accept General Call Address	<input type="checkbox"/>																																														
<b>Connections</b>																																															
Clock	<unassigned>																																														
SCL	<unassigned>																																														
SDA	<unassigned>																																														
SCL Output (scl_trig)	<unassigned>																																														
<b>Actual Data Rate</b>																																															
Actual Data Rate (kbps)	100																																														
Clock Frequency																																															
<b>Advanced</b>																																															
Store Config in Flash	<input checked="" type="checkbox"/>																																														

## SPI

Replicate the settings as appropriate.

### PSoC Creator

Configure 'SPIm1'

Name: **SPIm1**

Basic | Advanced | Pins | Outlets

**Basic**

☐ Clock Source

Enable Clock from Terminal ☐ f(x)

**General**

Mode: Master f(x)

Sub Mode: Motorola f(x)

SCLK Mode: CPHA = 0, CPOL = 0 f(x)

Data Rate (kbps): 125 f(x)

Oversample: 16 f(x)

Enable Input Glitch Filter ☐ f(x)

Enable MISO Late Sampling ☐ f(x)

SCLK Free Running ☐ f(x)

**Data Configuration**

Bit Order: MSB First f(x)

RX Data Width: 8 f(x)

TX Data Width: 8 f(x)

**Slave Select**

Deassert SS Between Data Elements ☐ f(x)

Number of SS: 0 f(x)

**Waveform**

Actual data rate (kbps): UNKN

Diagram showing SCLK, MOSI, MISO, and Sample signals over 8 data elements (D7 to D0).

Buttons: Datasheet, OK, Apply, Cancel

### ModusToolbox Device Configurator

Serial Communication Block (SCB) 0 - Parameters

Enter filter text...

Name	Value
<b>Peripheral Documentation</b>	
Configuration Help	<a href="#">Open SPI SCB Documentation</a>
<b>General</b>	
Mode	Slave
Sub Mode	Motorola
SCLK Mode	CPHA = 0, CPOL = 0
Data Rate (kbps)	1000
Enable Input Glitch Filter	<input type="checkbox"/>
<b>Data Configuration</b>	
Bit Order	MSB First
RX Data Width	8
TX Data Width	8
<b>Slave Select</b>	
SS0 Polarity	Active Low
SS1 Polarity	Active Low
SS2 Polarity	Active Low
<b>Connections</b>	
Clock	<unassigned>
SCLK	P0[4] digital_inout [USED]
MOSI	<unassigned>
MISO	<unassigned>
SS0	<unassigned>
SS1	<unassigned>
SS2	<unassigned>
RX Trigger Output	<unassigned>
TX Trigger Output	<unassigned>
<b>Data Rate</b>	
Actual Data Rate (kbps)	
Clock Frequency	
<b>Trigger Level</b>	
RX FIFO Level	63
TX FIFO Level	63
<b>API Mode</b>	
API Mode	High Level
<b>Advanced</b>	
Store Config in Flash	<input checked="" type="checkbox"/>

## UART

Replicate the settings as appropriate.

### PSoC Creator

Configure 'UART\_Debug'

Name: UART\_Debug

**Basic** | Advanced | Pins | Built-in

Clock Source	
Enable Clock from Terminal	<input type="checkbox"/> f(x)

General	
Com Mode	Standard f(x)
TX/RX Mode	TX + RX f(x)
Baud Rate (bps)	115200 f(x)
Oversample	12 f(x)
Bit Order	LSB First f(x)
Data Width	8 bits f(x)
Parity	None f(x)
Stop Bits	1 f(x)
Enable Digital Filter	<input type="checkbox"/> f(x)

Actual baud rate (bps): 115741 ⓘ

Datasheet OK Apply Cancel

### ModusToolbox Device Configurator

Serial Communication Block (SCB) 0 - Parameters

Enter filter text...

Name	Value
<b>Peripheral Documentation</b>	
Configuration Help	<a href="#">Open UART (SCB) Documentation</a>
<b>General</b>	
Com Mode	Standard
Baud Rate (bps)	115200
Oversample	8
Bit Order	LSB First
Data Width	8 bits
Parity	None
Stop Bits	1 bit
Enable Digital Filter	<input type="checkbox"/>
<b>Support RS-485</b>	
TX-Enable	<input type="checkbox"/>
<b>Flow Control</b>	
Enable Flow Control	<input type="checkbox"/>
CTS Polarity	Active Low
RTS Polarity	Active Low
RTS Activation Level	63
<b>Connections</b>	
Clock	<unassigned>
RX	<unassigned>
TX	<unassigned>
RX Trigger Output	<unassigned> ...
TX Trigger Output	<unassigned> ...
<b>Actual Baud Rate</b>	
Actual Baud Rate (bps)	
Baud Rate Accuracy (%)	
Clock Frequency	
<b>Trigger Level</b>	
RX FIFO Level	63
TX FIFO Level	63
<b>Multi Processor Mode</b>	
Enable Multi Processor Mode	<input type="checkbox"/>
Address	0

## SMIF (QSPI)

Replicate the settings as appropriate. Refer also to the [ModusToolbox QSPI Configurator Guide](#), as needed.

### PSoC Creator

Configure 'SMIF\_FLASH'
 ? X

Name:

Basic Built-in
 ◀ ▶

DMA Trigger Outputs		
RX FIFO DMA Trigger	<input type="checkbox"/>	fix
TX FIFO DMA Trigger	<input type="checkbox"/>	fix

GPIO Configuration		
SMIF Datalines [0:1]	<input checked="" type="checkbox"/>	fix
SMIF Datalines [2:3]	<input checked="" type="checkbox"/>	fix
SMIF Datalines [4:5]	<input type="checkbox"/>	fix
SMIF Datalines [6:7]	<input type="checkbox"/>	fix
SMIF SPI Slave Select 0	<input checked="" type="checkbox"/>	fix
SMIF SPI Slave Select 1	<input type="checkbox"/>	fix
SMIF SPI Slave Select 2	<input type="checkbox"/>	fix
SMIF SPI Slave Select 3	<input type="checkbox"/>	fix

Interrupt Cause		
Memory Mode Alignment Error	<input type="checkbox"/>	fix
RX Data FIFO Underflow	<input type="checkbox"/>	fix
TX Command FIFO Overflow	<input type="checkbox"/>	fix
TX Data FIFO Overflow	<input type="checkbox"/>	fix

TX and RX FIFO Trigger Levels		
RX FIFO Trigger Level	0	fix
TX FIFO Trigger Level	0	fix

Advanced user: Build configuration		
Generate code from cy_smif.cysmif file	<input type="checkbox"/>	fix

Datasheet OK Apply Cancel

### ModusToolbox Device Configurator

Quad Serial Memory Interface (QSPI) 0 - Parameters
 ⌵ X

Enter filter text...

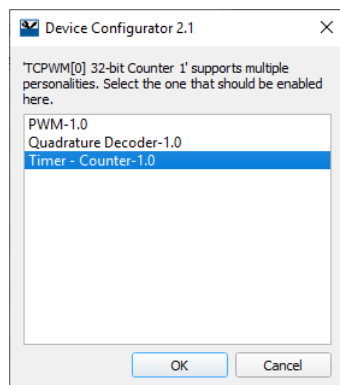
Name	Value
<b>Peripheral Documentation</b>	
Configuration Help	<a href="#">Open SMIF Documentation</a>
<b>Clocks</b>	
HF Clock	CLK_HF0 root_clk [USED]
Interface Clock	CLK_HF2 root_clk [USED]
SPI Clock	P11[7] digital_inout [USED]
<b>External tools</b>	
QSPI Configurator	<a href="#">Launch QSPI Configurator</a> ...
<b>Data</b>	
SPI Data[0]	<unassigned> ...
SPI Data[1]	<unassigned> ...
SPI Data[2]	<unassigned> ...
SPI Data[3]	<unassigned> ...
SPI Data[4]	<unassigned> ...
SPI Data[5]	<unassigned> ...
SPI Data[6]	<unassigned> ...
SPI Data[7]	<unassigned> ...
<b>Slave Select</b>	
SPI Slave Select 0	<unassigned> ...
SPI Slave Select 1	<unassigned> ...
SPI Slave Select 2	<unassigned> ...
SPI Slave Select 3	<unassigned> ...
<b>Interrupt</b>	
Memory Mode Alignment Error	<input type="checkbox"/>
RX Data FIFO Underflow	<input type="checkbox"/>
TX Command FIFO Overflow	<input type="checkbox"/>
TX Data FIFO Overflow	<input type="checkbox"/>
<b>DMA Triggers</b>	
RX Trigger Output	<unassigned> ...
RX FIFO Trigger Level	0
TX Trigger Output	<unassigned> ...
TX FIFO Trigger Level	0
<b>Advanced</b>	
Store Config in Flash	<input checked="" type="checkbox"/>



## TCPWM

### Personalities

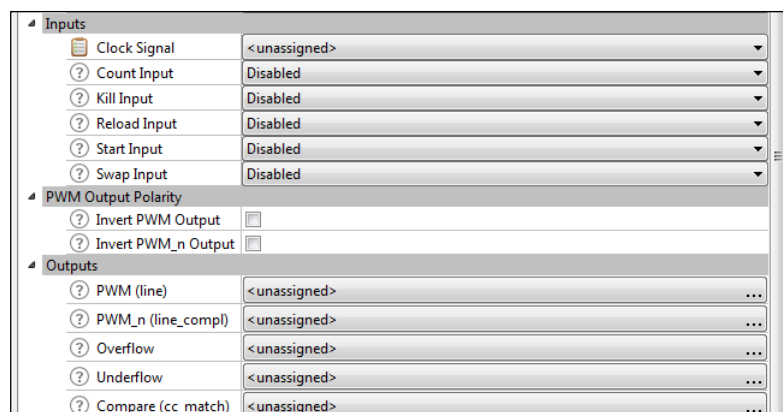
Like the SCB block, the TCPWM block includes several personalities in the Device Configurator: PWM, Quadrature Decoder, and Timer. When you enable a TCPWM block, a dialog displays to select the appropriate personality:




### Assign Inputs and Outputs

For the TCPWM block, the biggest change between PSoC Creator and ModusToolbox is routing into and out of the TCPWM. In PSoC Creator, you could connect wires between the TCPWM inputs and outputs and various other Components on the schematic. ModusToolbox does not have a schematic, so it requires a different mechanism.


1. To assign inputs and outputs in ModusToolbox, go to the Device Configurator Parameters pane for the TCPWM personality and scroll down to **Inputs** and **Outputs** sections, as shown:



2. For **Inputs**, select the type of input required: **Rising Edge**, **Falling Edge**, **Either Edge**, Or **Level** (if available). A new row appears where you can choose the source of the input from a drop-down menu.
3. After choosing the input, click the link  icon to jump to the Parameters pane for the peripheral connected to the TCPWM.
4. You also need to assign a clock input to the TCPWM from one of the many programmable clock dividers.
5. For the TCPWM **Outputs**, use the "..." to select multiple locations to route the TCPWM output.

Each TCPWM has specific outputs to which the **PWM** and **PWM\_n** signals are connected. In PSoC Creator, if you connect a Pin Component to the **PWM** or **PWM\_n** outputs and choose a pin, the appropriate TCPWM is selected automatically.

In ModusToolbox, you must select the correct TCPWM for your application. The easiest way to do this is:

- ☐ Go to the **Pins** tab in the Device Configurator.
- ☐ Enable the pin you want to use for a TCPWM output.
- ☐ Click on the digital output selection choose the TCPWM you want.
- ☐ Then, use the  to open the Parameters pane for that TCPWM.

For each TCPWM personality (PWM, Timer Counter, Quadrature Decoder), copy the configuration information from the PSoC Creator Component to the Device Configurator.

## MCWDT

Replicate the settings as appropriate.

PSoC Creator	ModusToolbox Device Configurator																																		
<div style="border: 1px solid #ccc; padding: 5px;"> <p>Configure 'MCWDT_System' <span style="float: right;">? X</span></p> <p>Name: <span style="border: 1px solid #00aaff; padding: 2px;">MCWDT_System</span></p> <div style="display: flex; border-bottom: 1px solid #ccc; margin-bottom: 5px;"> <span style="border: 1px solid #ccc; padding: 2px 5px;">General</span> <span style="border: 1px solid #ccc; padding: 2px 5px; margin-left: 5px;">Built-in</span> </div> <div style="display: flex;"> <div style="flex: 1; border-right: 1px solid #ccc; padding: 5px;"> <div style="margin-bottom: 5px;"> <div style="border: 1px solid #ccc; padding: 2px;">Cascade</div> <div style="display: flex; justify-content: space-between; align-items: center;"> <div>Cascade C0C1</div> <input type="checkbox"/> <div>f(x)</div> </div> <div style="display: flex; justify-content: space-between; align-items: center;"> <div>Cascade C1C2</div> <input type="checkbox"/> <div>f(x)</div> </div> </div> <div> <div style="border: 1px solid #ccc; padding: 2px;">Counter0</div> <div style="display: flex; justify-content: space-between; align-items: center;"> <div>Enable counter</div> <input type="checkbox"/> <div>f(x)</div> </div> <div style="display: flex; justify-content: space-between; align-items: center;"> <div>Match</div> <div>60000</div> <div>f(x)</div> </div> <div style="display: flex; justify-content: space-between; align-items: center;"> <div>Mode</div> <div>Interrupt</div> <div>f(x)</div> </div> <div style="display: flex; justify-content: space-between; align-items: center;"> <div>Clear on Match</div> <div>Clear on match</div> <div>f(x)</div> </div> </div> <div> <div style="border: 1px solid #ccc; padding: 2px;">Counter1</div> <div style="display: flex; justify-content: space-between; align-items: center;"> <div>Enable counter</div> <input checked="" type="checkbox"/> <div>f(x)</div> </div> <div style="display: flex; justify-content: space-between; align-items: center;"> <div>Match</div> <div>32768</div> <div>f(x)</div> </div> <div style="display: flex; justify-content: space-between; align-items: center;"> <div>Mode</div> <div>No action</div> <div>f(x)</div> </div> <div style="display: flex; justify-content: space-between; align-items: center;"> <div>Clear on Match</div> <div>Free running</div> <div>f(x)</div> </div> </div> <div> <div style="border: 1px solid #ccc; padding: 2px;">Counter2</div> <div style="display: flex; justify-content: space-between; align-items: center;"> <div>Enable counter</div> <input checked="" type="checkbox"/> <div>f(x)</div> </div> <div style="display: flex; justify-content: space-between; align-items: center;"> <div>Period / Toggle Bit</div> <div>Toggle bit 31</div> <div>f(x)</div> </div> <div style="display: flex; justify-content: space-between; align-items: center;"> <div>Mode</div> <div>Interrupt</div> <div>f(x)</div> </div> </div> </div> <div style="flex: 1; padding: 5px;"> <p style="font-size: small;">All counters are clocked by either LFCLK (nominal 32 kHz) or by a cascaded counter.</p> <div style="display: flex; justify-content: space-around; margin-bottom: 5px;"> <div style="border: 1px solid #ccc; padding: 2px; text-align: center;">C2 (32-bit)</div> <div style="border: 1px solid #ccc; padding: 2px; text-align: center;">C1 (16-bit)</div> <div style="border: 1px solid #ccc; padding: 2px; text-align: center;">C0 (16-bit)</div> </div> <div style="display: flex; justify-content: space-around; font-size: x-small;"> <div>           Freq (Hz): 0.00            Period (hr): 18.64         </div> <div>           Freq (Hz): 0.49            Period (s): 2.05         </div> <div>           Freq (Hz): 0.53            Period (s): 1.88         </div> </div> <p style="font-size: x-small;">The interrupts are OR'd together to a single interrupt</p> </div> </div> <div style="display: flex; justify-content: flex-end; margin-top: 5px;"> <span style="border: 1px solid #ccc; padding: 2px 10px; margin-right: 5px;">Datasheet</span> <span style="border: 1px solid #00aaff; padding: 2px 10px; margin-right: 5px;">OK</span> <span style="border: 1px solid #ccc; padding: 2px 10px; margin-right: 5px; background-color: #f0f0f0;">Apply</span> <span style="border: 1px solid #ccc; padding: 2px 10px; background-color: #f0f0f0;">Cancel</span> </div> </div>	<div style="border: 1px solid #ccc; padding: 5px;"> <p>Multi-Counter Watchdog Timer (MCWDT) 0 - Parameters - Modus C... <span style="float: right;">X</span></p> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 5px;">Enter filter text...</div> <table style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #f0f0f0;"> <th style="text-align: left; padding: 5px;">Name</th> <th style="text-align: left; padding: 5px;">Value</th> </tr> </thead> <tbody> <tr style="background-color: #f0f0f0;"> <td colspan="2" style="padding: 5px;">▼ Counter0</td> </tr> <tr> <td style="padding: 5px;">? Clear on Match</td> <td style="padding: 5px;">Free running ▼</td> </tr> <tr> <td style="padding: 5px;">? Match</td> <td style="padding: 5px;">32768</td> </tr> <tr> <td style="padding: 5px;">? Mode</td> <td style="padding: 5px;">No action ▼</td> </tr> <tr style="background-color: #f0f0f0;"> <td colspan="2" style="padding: 5px;">▼ Counter1</td> </tr> <tr> <td style="padding: 5px;">? Clear on Match</td> <td style="padding: 5px;">Free running ▼</td> </tr> <tr> <td style="padding: 5px;">? Match</td> <td style="padding: 5px;">32768</td> </tr> <tr> <td style="padding: 5px;">? Mode</td> <td style="padding: 5px;">No action ▼</td> </tr> <tr style="background-color: #f0f0f0;"> <td colspan="2" style="padding: 5px;">▼ Counter2</td> </tr> <tr> <td style="padding: 5px;">? Mode</td> <td style="padding: 5px;">No action ▼</td> </tr> <tr> <td style="padding: 5px;">? Period / Toggle Bit</td> <td style="padding: 5px;">16</td> </tr> <tr style="background-color: #f0f0f0;"> <td colspan="2" style="padding: 5px;">▼ Cascade</td> </tr> <tr> <td style="padding: 5px;">? Cascade C0C1</td> <td style="padding: 5px;"><input type="checkbox"/></td> </tr> <tr> <td style="padding: 5px;">? Cascade C1C2</td> <td style="padding: 5px;"><input type="checkbox"/></td> </tr> <tr style="background-color: #f0f0f0;"> <td colspan="2" style="padding: 5px;">▼ Advanced</td> </tr> <tr> <td style="padding: 5px;">? Store Config in Flash</td> <td style="padding: 5px;"><input checked="" type="checkbox"/></td> </tr> </tbody> </table> </div>	Name	Value	▼ Counter0		? Clear on Match	Free running ▼	? Match	32768	? Mode	No action ▼	▼ Counter1		? Clear on Match	Free running ▼	? Match	32768	? Mode	No action ▼	▼ Counter2		? Mode	No action ▼	? Period / Toggle Bit	16	▼ Cascade		? Cascade C0C1	<input type="checkbox"/>	? Cascade C1C2	<input type="checkbox"/>	▼ Advanced		? Store Config in Flash	<input checked="" type="checkbox"/>
Name	Value																																		
▼ Counter0																																			
? Clear on Match	Free running ▼																																		
? Match	32768																																		
? Mode	No action ▼																																		
▼ Counter1																																			
? Clear on Match	Free running ▼																																		
? Match	32768																																		
? Mode	No action ▼																																		
▼ Counter2																																			
? Mode	No action ▼																																		
? Period / Toggle Bit	16																																		
▼ Cascade																																			
? Cascade C0C1	<input type="checkbox"/>																																		
? Cascade C1C2	<input type="checkbox"/>																																		
▼ Advanced																																			
? Store Config in Flash	<input checked="" type="checkbox"/>																																		

## RTC

Replicate the settings as appropriate.

PSoC Creator	ModusToolbox Device Configurator																																										
<p>Configure 'RTC'</p> <p>Name: <input type="text" value="RTC"/></p> <p><b>General</b> Built-in</p> <p><b>General Settings</b></p> <table border="1"> <tr> <td>Date Format</td> <td>MM/DD/YYYY</td> <td>f(x)</td> </tr> <tr> <td>Enable Interrupts</td> <td><input checked="" type="checkbox"/></td> <td>f(x)</td> </tr> <tr> <td>Time Reset on Start</td> <td><input type="checkbox"/></td> <td>f(x)</td> </tr> </table> <p><b>Daylight Saving (DST) Settings</b></p> <table border="1"> <tr> <td>Enable DST Functionality</td> <td><input type="checkbox"/></td> <td>f(x)</td> </tr> </table> <p>Datasheet OK Apply Cancel</p>	Date Format	MM/DD/YYYY	f(x)	Enable Interrupts	<input checked="" type="checkbox"/>	f(x)	Time Reset on Start	<input type="checkbox"/>	f(x)	Enable DST Functionality	<input type="checkbox"/>	f(x)	<p>Real Time Clock (RTC) - Parameters - Modus Configurator</p> <p>Enter filter text...</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td><b>General</b></td> <td></td> </tr> <tr> <td>  Date Format</td> <td>MM/DD/YYYY</td> </tr> <tr> <td>  Enable Daylight Savings (DST)</td> <td><input type="checkbox"/></td> </tr> <tr> <td><b>Time and Date</b></td> <td></td> </tr> <tr> <td>  Seconds</td> <td>0</td> </tr> <tr> <td>  Minutes</td> <td>0</td> </tr> <tr> <td>  Hours Format</td> <td>24H</td> </tr> <tr> <td>  Hours</td> <td>12</td> </tr> <tr> <td>  Day of the Week</td> <td>Sunday</td> </tr> <tr> <td>  Month</td> <td>1</td> </tr> <tr> <td>  Day of the Month</td> <td>January</td> </tr> <tr> <td>  Year</td> <td>0</td> </tr> <tr> <td><b>Advanced</b></td> <td></td> </tr> <tr> <td>  Store Config in Flash</td> <td><input checked="" type="checkbox"/></td> </tr> </tbody> </table>	Name	Value	<b>General</b>		Date Format	MM/DD/YYYY	Enable Daylight Savings (DST)	<input type="checkbox"/>	<b>Time and Date</b>		Seconds	0	Minutes	0	Hours Format	24H	Hours	12	Day of the Week	Sunday	Month	1	Day of the Month	January	Year	0	<b>Advanced</b>		Store Config in Flash	<input checked="" type="checkbox"/>
Date Format	MM/DD/YYYY	f(x)																																									
Enable Interrupts	<input checked="" type="checkbox"/>	f(x)																																									
Time Reset on Start	<input type="checkbox"/>	f(x)																																									
Enable DST Functionality	<input type="checkbox"/>	f(x)																																									
Name	Value																																										
<b>General</b>																																											
Date Format	MM/DD/YYYY																																										
Enable Daylight Savings (DST)	<input type="checkbox"/>																																										
<b>Time and Date</b>																																											
Seconds	0																																										
Minutes	0																																										
Hours Format	24H																																										
Hours	12																																										
Day of the Week	Sunday																																										
Month	1																																										
Day of the Month	January																																										
Year	0																																										
<b>Advanced</b>																																											
Store Config in Flash	<input checked="" type="checkbox"/>																																										

## BLE

Replicate all settings from the PSoC Creator Component customizer to the ModusToolbox Bluetooth Configurator. Refer also to the [ModusToolbox Bluetooth Configurator Guide](#), as needed.

PSoC Creator	ModusToolbox Bluetooth Configurator
<p>Configure 'BLE'</p> <p>Name: <input type="text" value="BLE"/></p> <p><b>General</b> GATT Settings GAP Settings L2CAP Settings Link Layer Settings Advanced Built-in</p> <p>Load configuration Save configuration</p> <p><input checked="" type="radio"/> Complete BLE Protocol</p> <p>Maximum number of BLE connections: 1</p> <p>GAP role</p> <p><input checked="" type="checkbox"/> Peripheral <input type="checkbox"/> Broadcaster</p> <p><input type="checkbox"/> Central <input type="checkbox"/> Observer</p> <p>CPU core: Dual core (Controller on CM0+, Host and Profiles on CM4)</p> <p>Over-The-Air bootloading with code sharing</p> <p><input checked="" type="radio"/> Disabled</p> <p><input type="radio"/> Stack and Profile</p> <p><input type="radio"/> Profile only</p> <p><input type="radio"/> BLE Controller only (HCI over UART)</p> <p>Datasheet OK Apply Cancel</p>	<p>D:/ModusToolbox/CodeExamples/CE212736_mtw/CE212736_config/GeneratedSour...</p> <p>File Help</p> <p>Save</p> <p><b>General</b> GATT Settings GAP Settings L2CAP Settings Link Layer Settings</p> <p><input checked="" type="radio"/> Complete BLE Protocol</p> <p>Maximum number of BLE connections: 1</p> <p>GAP role</p> <p><input checked="" type="checkbox"/> Peripheral <input type="checkbox"/> Broadcaster</p> <p><input type="checkbox"/> Central <input type="checkbox"/> Observer</p> <p>Over-The-Air bootloading with code sharing</p> <p><input checked="" type="radio"/> Disabled</p> <p><input type="radio"/> Stack and Profile</p> <p><input type="radio"/> Profile only</p> <p><input type="radio"/> BLE Controller only (HCI over UART)</p>

For a detailed explanation regarding how to configure the correct BLE middleware appropriate for your project, refer to the [BLE Middleware API Reference Manual](#).BLE

## Emulated EEPROM

Unlike most ModusToolbox peripherals, the Emulated EEPROM is not configured by the Device Configurator. Use the Library Manager to enable the middleware. Refer to the [Library Manager User Guide](#) for more details.

**PSoC Creator**

Configure 'Em\_EEPROM\_v2\_0'

Name:

**Basic** Built-in

EEPROM Size	256	<input type="text" value="f(x)"/>
Redundant Copy	Yes	<input type="text" value="f(x)"/>
Use Blocking Write	Yes	<input type="text" value="f(x)"/>
Use Emulated EEPROM	Yes	<input type="text" value="f(x)"/>
Wear Level Factor	2x	<input type="text" value="f(x)"/>

Actual EEPROM size (bytes): 2048

**ModusToolbox Library Manager**

Library Manager 1.1

Settings Help

Directory:

Project:

Active BSP: CY8CKIT-062-WIFI-BT

Enter filter text

BSPs Libraries

Name	Version
Board Utils	
PSoC 6 Base Libraries	
core-lib	Latest 1.X release
psoc6cm0p	Latest 1.X release
psoc6hal	Latest 1.X release
psoc6make	Latest 1.X release
psoc6pdl	Latest 1.X release
PSoC 6 Middleware	
capsense	Latest 2.X release
clib-support	Latest 1.X release
csdadc	Latest 2.X release
csdidac	Latest 2.X release
dfu	Latest 4.X release
*emEEPROM	Latest 2.X release
emwin	Latest 3.X release
freertos	Latest 10.X release
usbdev	Latest 2.X release
WiFi Middleware libraries	

The Emulated EEPROM does not have a configuration dialog in ModusToolbox. Refer to the [Em EEPROM Middleware API Reference Manual](#) to learn how to configure the Emulated EEPROM middleware. There is no way to alias it, so all the PSoC Creator API calls must be renamed to use the middleware API directly.

## DFU

The Bootloader was renamed to Device Firmware Update (DFU) tool. In PSoC Creator, the DFU SDK is enabled in the project Build Settings under Peripheral Driver Library. In ModusToolbox, use the Library Manager to enable the dfu middleware. Refer to the [Library Manager User Guide](#) for more details.

### PSoC Creator

Build Settings

Configuration: Debug (Active)

Toolchain: ARM GCC 5.4-2016-q2-update

Example\_Project

- Code Generation
- Debug
- Customizer
- Peripheral Driver Library
- Target IDEs
- CM0+ ARM GCC 5.4-2016-q2-update
- CM4 ARM GCC 5.4-2016-q2-update

Default (Tools > Options): C:\Program Files (x86)\Cypress\VPDL\3.1.2

Custom

Software package imports:

Expand Collapse Check All Uncheck All

Package	Variant	Version	Description
LCD Driver	FlexColor	5.46	FlexColor
DFU			
DFU_SDK			
Core		3.0.0	DFU SDK
App type	loader	3.0.0	Files spe
Communication UART		3.0.0	UART co
Communication BLE		3.0.0	BLE com
Communication I2C		3.0.0	I2C com
Communication SPI		3.0.0	SPI com
RTOS			
FreeRTOS		10.0.1	FreeRTO
Memory Management	heap_1	10.0.1	The simp
utilities			

OK Apply Cancel

### ModusToolbox Library Manager

Library Manager 1.1

Settings Help

Directory:

Project:

Active BSP: CY8CKIT-062-BLE

Enter filter text

BSPs Libraries

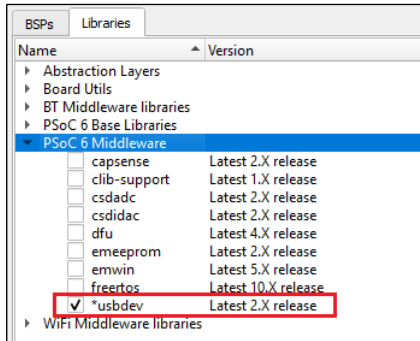
Name Version

- Abstraction Layers
- Board Utils
- PSoC 6 Base Libraries
- PSoC 6 Middleware
  - bless Latest 3.X release
  - capsense Latest 2.X release
  - clib-support Latest 1.X release
  - csdadc Latest 2.X release
  - csdidac Latest 2.X release
  - \*dfu Latest 4.X release**
  - emeeprom Latest 2.X release
  - emwin Latest 5.X release
  - freertos Latest 10.X release

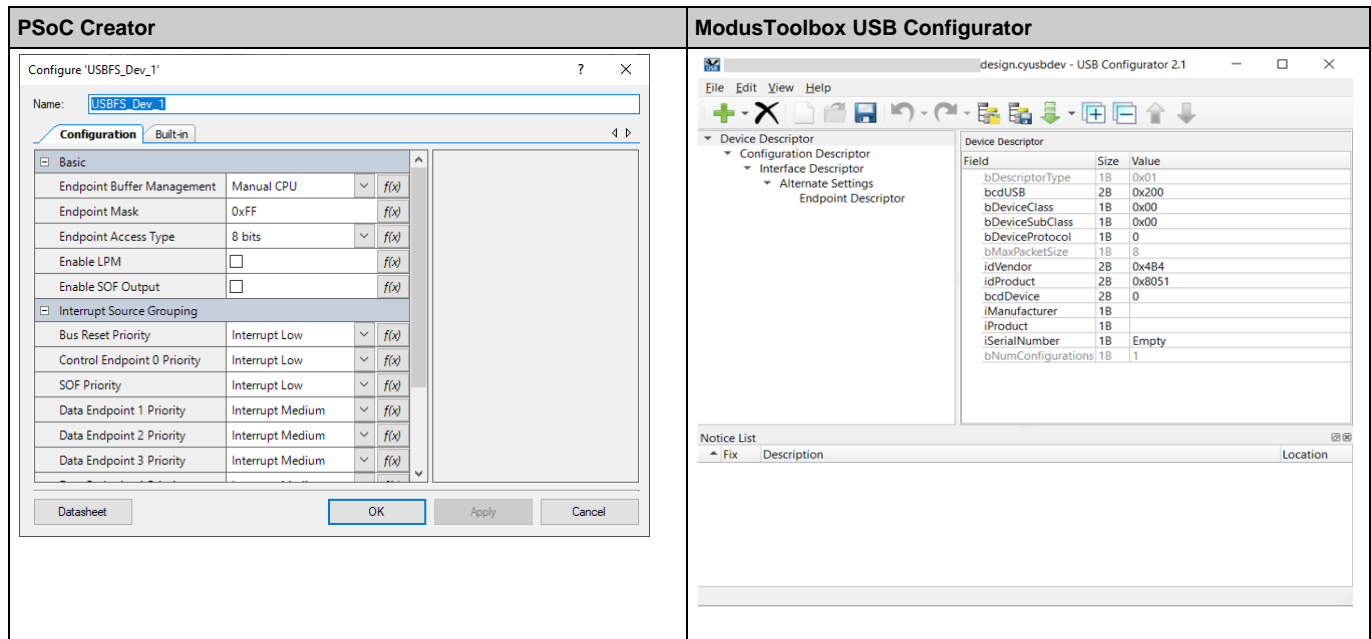
Refer to the [Device Firmware Update Host User Guide](#) and the [DFU Middleware API Reference Manual](#) to learn how to configure the DFU middleware.

## USB

A USB application can use the USB PDL driver directly or the USB middleware in ModusToolbox. Use the Library Manager to enable the middleware. Refer to the [Library Manager User Guide](#) for more details.



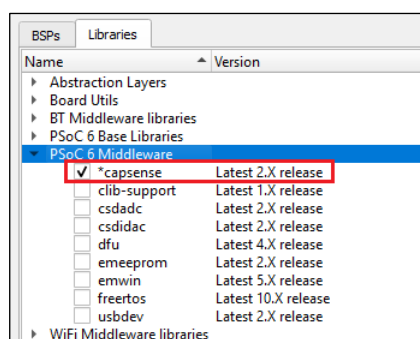
The PSoC Creator USB Configurator settings can be replicated to the ModusToolbox USB Configurator. For USB Configurator details, refer to the [ModusToolbox USB Configurator Guide](#).



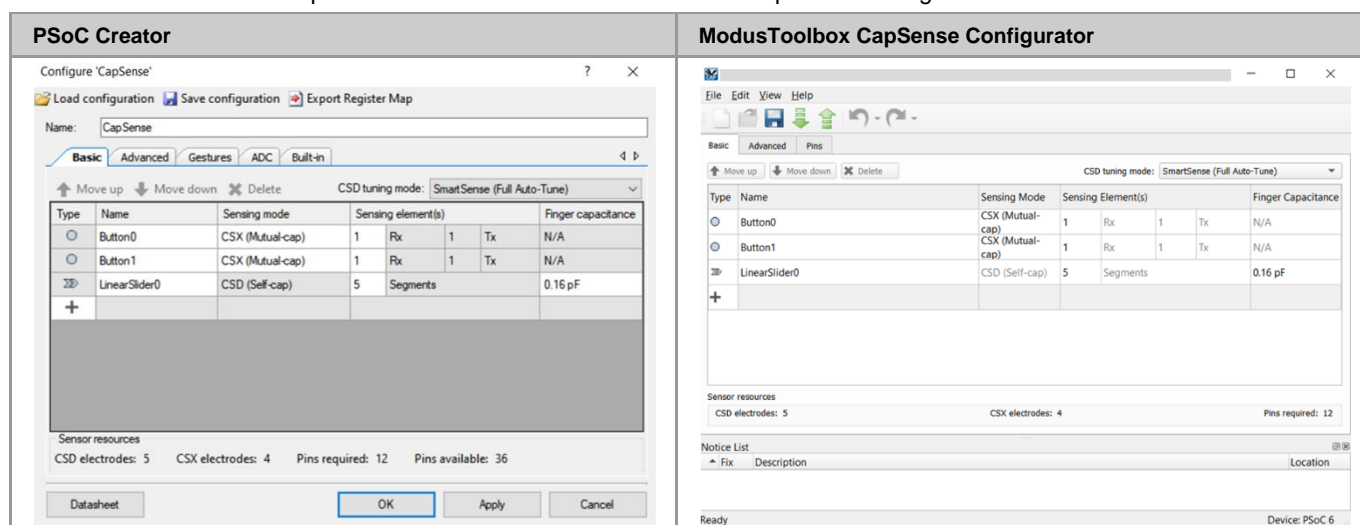
Refer to the [USB Middleware API Reference Manual](#) to learn how to configure the USB peripheral. The USB middleware in turn uses the USB PDL driver. If your PSoC Creator project uses the USB PDL driver, then you should be able to port the project following the guidelines mentioned in this document for a normal PDL driver.

## CapSense

CapSense is delivered as a middleware library in ModusToolbox. Use the Library Manager to enable the middleware. Refer to the [Library Manager User Guide](#) for more details.



CapSense middleware is enabled by default for BSPs that support CapSense. CapSense configuration settings can be replicated from the PSoC Creator Component customizer to the ModusToolbox CapSense Configurator.



Instead of setting the clock frequency, you now set the clock frequency divider in the CapSense Configurator. The **Tx clock source** settings and the **Sense clock source** settings are now in the **Widget Details** subtab under **Advanced** tab. Gestures are also in the **Widget Details** tab for the Linear Sliders.

The CapSense ADC is a middleware library in ModusToolbox called CSDADC. Enable this middleware like any other middleware using the Library Manager Tool.

For information on designing your CapSense applications, refer to the [PSoC 4 and PSoC 6 MCU CapSense Design Guide](#). For CapSense Configurator specific details you can refer to the [CapSense Configurator Guide](#).

## 5. Copying Code



### Importing Your Code

If you've used other files beyond the *main.c* file in PSoC Creator to build your application, you can simply drag and drop those files into the application directory and they will be included in your application. This also works for directories. Build your application and start working on the errors. (The following sections will help with resolving some of these errors.)

### Building *project.h* File

The files you copy from a PSoC Creator project likely reference the *project.h* file to pull in the specific component settings. ModusToolbox doesn't have a *project.h* file. You should create this file as it is a convenient place to put headers, defines, and constants that you might need elsewhere in your project. To get started, put the following in your *project.h* file:

```
#include "cy_pdl.h"
#include "cybsp.h"
```

### Rewriting *main.c* File

If you copy the *main.c* file from your PSoC Creator design, nothing will work. Pins and clocks are not initialized automatically in ModusToolbox software. Add the following calls to your *main.c* file so that everything is initialized correctly:

```
cy_rslt_t result;

/* Initialize the device and board peripherals */
result = cybsp_init() ;
if (result != CY_RSLT_SUCCESS)
{
    CY_ASSERT(0);
}

enable_irq();
```

### Rewriting Component-Specific APIs and Migrating ISRs

There are two approaches to this:

- Inspect your code, find all Component-specific APIs, and look up the correct PDL call to do the same thing.
- Alternatively, you can simply build the application and look at the errors to help find all the component API calls.

Additionally, if you go to the same code in PSoC Creator, you can right-click on the call and use the "Go to Definition" command to find the code. In most cases, you can copy the component-specific API code and replace the API call with it. Or you can copy the code into a file and prototype into a header file to quickly move the call over.

As for ISRs, follow the same techniques described above.