PSoC™ Creator to ModusToolbox™ porting guide

ModusToolbox™ tools package 3.x

About this document

Scope and purpose

This document is a guide to the porting process from PSoC™ Creator to ModusToolbox™ and points you towards the right documentation based on your project's needs.

Table of Contents

Table of Contents

1 Introduction

1.1 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*.

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 Infineon Developer Community. We constantly strive to make ModusToolbox™ better to enhance the overall user experience. Your feedback is both welcome and appreciated.

1.2 Before You Begin

This guide expects that you have read the ModusToolbox™ tools package quick start guide. If not, please do that first.

ModusToolbox™ is a different product than PSoC™ Creator. Infineon 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™, Infineon recommends that you continue to develop with PSoC™ Creator, which remains a fully supported development tool. If you must use ModusToolbox™, reach out to out support team for assistance.

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™ 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™** (formerly 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 Infineon Developer Community. Infineon wants to hear about your issues, advise where possible, and prioritize features for future releases of ModusToolbox™.

1.3 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 of 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™. You can also use VS Code, IAR Embedded Workbench, or µVision MDK.

1.4 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.

2 Getting Started

2.1 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 ModusToolbox™ tools package quick start guide for details about the IDE, and refer to the ModusToolbox™ Project Creator user guide for details about the Project Creator tool.

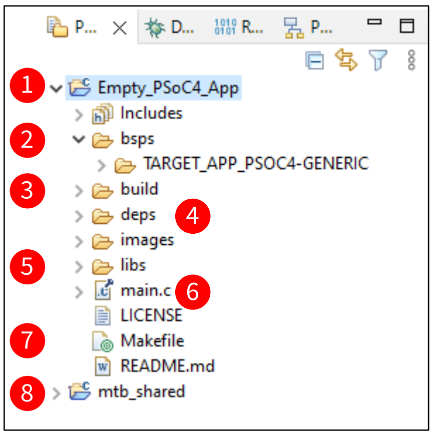
When creating the new application:

- Select the BSP for your device. If you are using a custom board, select the Create from MPN button to open the BSP Assistant and select the part number(s) for your device. Refer to the BSP Assistant user guide for more details.
- Select the application that you want to start with. If you want an empty project, you can select the **Empty App** for most BSPs. You can also rename your application. Once this is done, click **Create**.

Note: Many BSPs have a "GENERIC" option that provide minimum features for a cleaner starting point.

2.1.1 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:

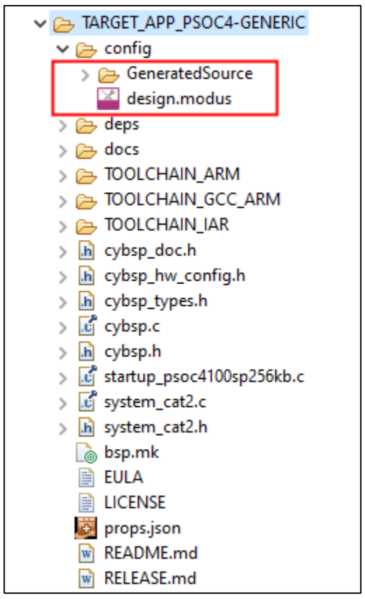


1. This is the application directory that contains all the source code and other files required to successfully build your application. You can have more than one application in a workspace.
2. The **bsps** directory contains all the code, configuration files, and toolchain information for the device(s) used for your application. See BSP Directory later in this section.
3. The **build** directory contains the build artifacts generated after the application is built. The build artifacts for each target are put into separate subdirectories.
4. The **deps** directory contains dependency files used to create your application.
5. The **libs** directory stores dependency files used to create your application.
6. 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.

7. 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™ tools package user guide. You can also refer to the Managing the Makefile for ModusToolbox™ software Knowledge Based Article.
8. The **mtb_shared** directory stores all shared libraries for your application that were downloaded fro GitHub.

2.1.2 BSP Directory

Inside the bsps 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 *config*, as shown in the following image:

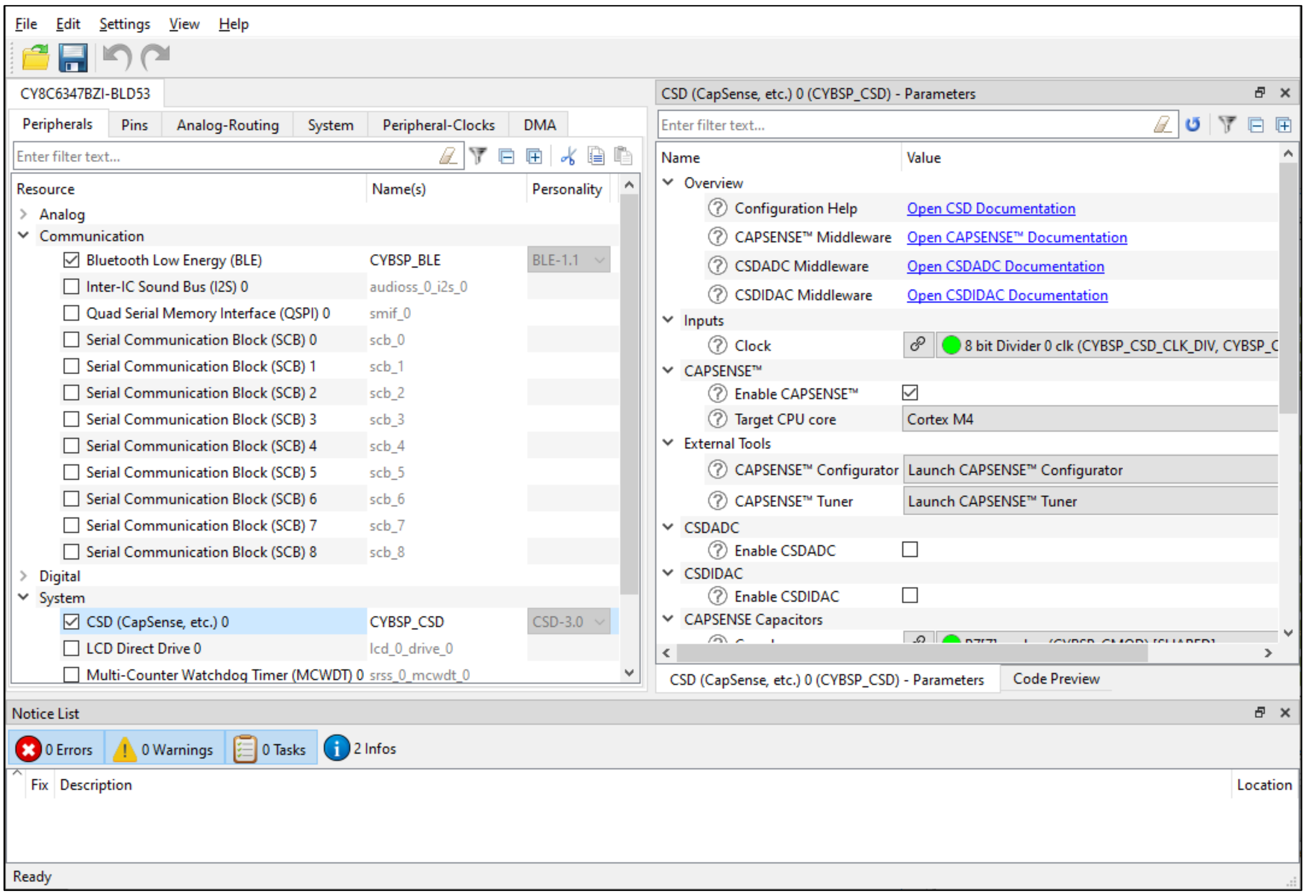


```
TARGET_APP_PSOC4-GENERIC
  config
    GeneratedSource
    design.modus
  deps
  docs
  TOOLCHAIN_ARM
  TOOLCHAIN_GCC_ARM
  TOOLCHAIN_IAR
  cybsp_doc.h
  cybsp_hw_config.h
  cybsp_types.h
  cybsp.c
  cybsp.h
  startup_psoc4100sp256kb.c
  system_cat2.c
  system_cat2.h
  bsp.mk
  EULA
  LICENSE
  props.json
  README.md
  RELEASE.md
```

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

2.1.3 Device Configurator

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


File Edit Settings View Help
CY8C6347BZI-BLD53
Peripherals Pins Analog-Routing System Peripheral-Clocks DMA
Enter filter text...
CSD (CapSense, etc.) 0 (CYBSP_CSD) - Parameters
Enter filter text...
CSD (CapSense, etc.) 0 (CYBSP_CSD) - Parameters Code Preview
Notice List
0 Errors 0 Warnings 0 Tasks 2 Infos
Fix Description Location
Ready

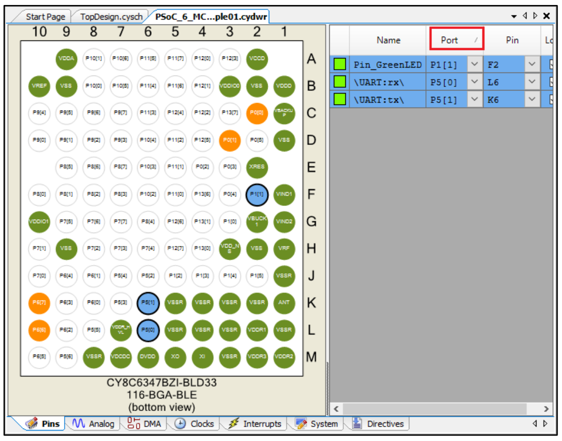
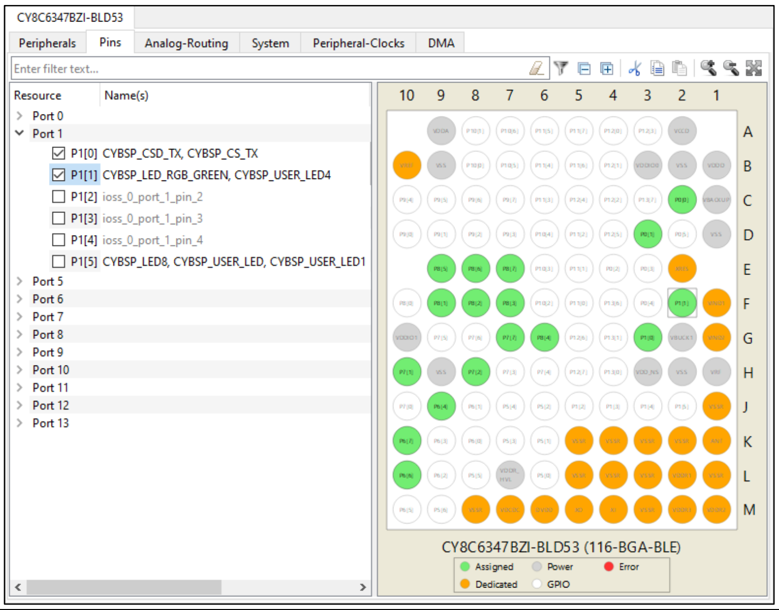
Resource	Name(s)	Personality
Analog		
Communication		
✓ Bluetooth Low Energy (BLE)	CYBSP_BLE	BLE-1.1
Inter-IC Sound Bus (I2S) 0	audioss_0_i2s_0	
Quad Serial Memory Interface (QSPI) 0	smif_0	
Serial Communication Block (SCB) 0	scb_0	
Serial Communication Block (SCB) 1	scb_1	
Serial Communication Block (SCB) 2	scb_2	
Serial Communication Block (SCB) 3	scb_3	
Serial Communication Block (SCB) 4	scb_4	
Serial Communication Block (SCB) 5	scb_5	
Serial Communication Block (SCB) 6	scb_6	
Serial Communication Block (SCB) 7	scb_7	
Serial Communication Block (SCB) 8	scb_8	
Digital		
System		
✓ CSD (CapSense, etc.) 0	CYBSP_CSD	CSD-3.0
LCD Direct Drive 0	lcd_0_drive_0	
Multi-Counter Watchdog Timer (MCWDT) 0	srss_0_mcwdt_0	

Name	Value
Overview	
Configuration Help	Open CSD Documentation
CAPSENSE™ Middleware	Open CAPSENSE™ Documentation
CSDADC Middleware	Open CSDADC Documentation
CSDIDAC Middleware	Open CSDIDAC Documentation
Inputs	
Clock	8 bit Divider 0 clk (CYBSP_CSD_CLK_DIV, CYBSP_C
CAPSENSE™	
Enable CAPSENSE™	✓
Target CPU core	Cortex M4
External Tools	
CAPSENSE™ Configurator	Launch CAPSENSE™ Configurator
CAPSENSE™ Tuner	Launch CAPSENSE™ Tuner
CSDADC	
Enable CSDADC	☐
CSDIDAC	
Enable CSDIDAC	☐
CAPSENSE Capacitors	

You can access the Device Configurator in the Tools section of Quick Panel in the Eclipse IDE for ModusToolbox™. Refer to the Device Configurator user guide for more details.

2.2 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.

PSoC™ Creator Pin Editor	ModusToolbox™ Device Configurator Pins Tab
	

	Name	Port	Pin
🟩	Pin_GreenLED	P1[1]	F2
🟩	\UART:rx\	P5[0]	L6
🟩	\UART:tx\	P5[1]	K6

CY8C6347BZI-BLD33
116-BGA-BLE
(bottom view)

CY8C6347BZI-BLD53

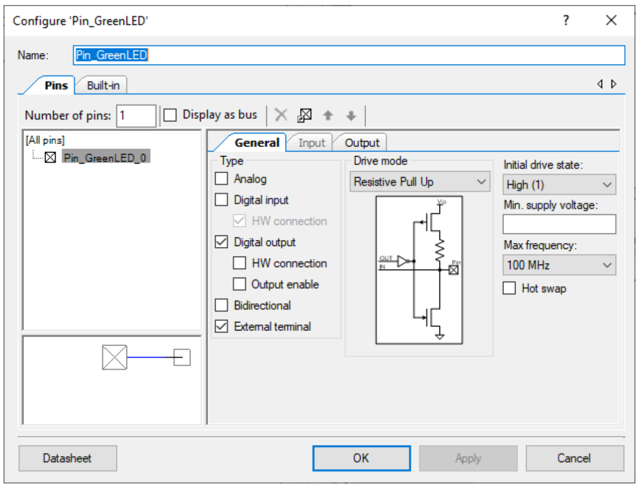
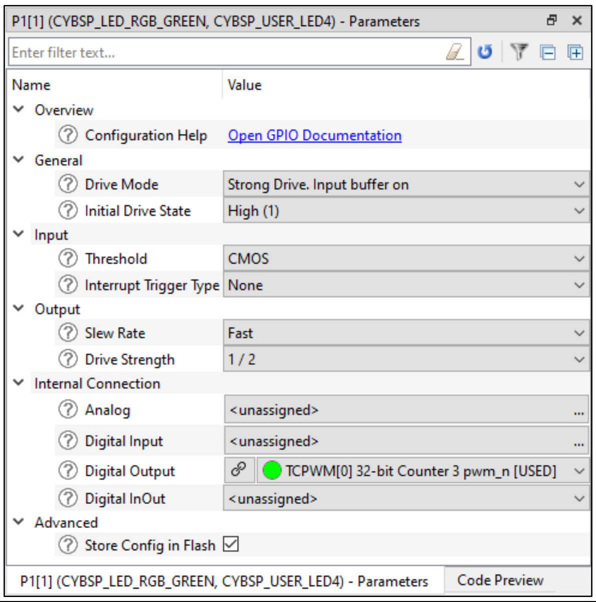
Resource	Name(s)
Port 0	
Port 1	
☑ P1[0]	CYBSP_CSD_TX, CYBSP_CS_TX
☑ P1[1]	CYBSP_LED_RGB_GREEN, CYBSP_USER_LED4
☐ P1[2]	ioss_0_port_1_pin_2
☐ P1[3]	ioss_0_port_1_pin_3
☐ P1[4]	ioss_0_port_1_pin_4
☐ P1[5]	CYBSP_LED8, CYBSP_USER_LED, CYBSP_USER_LED1
Port 5	
Port 6	
Port 7	
Port 8	
Port 9	
Port 10	
Port 11	
Port 12	
Port 13	

CY8C6347BZI-BLD53 (116-BGA-BLE)

9. Use the PSoC™ Creator list as a guide for which pins to enable in the Device Configurator.
10. 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.
11. 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.

12. 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.
13. 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
	

Configure 'Pin_GreenLED'

Name: Pin_GreenLED

Pins | Built-in

Number of pins: 1 ☐ Display as bus

[All pins]
Pin_GreenLED_0

General | Input | Output

Type
- ☐ Analog
- ☐ Digital input
  - ☑ HW connection
- ☑ Digital output
  - ☐ HW connection
  - ☐ Output enable
- ☐ Bidirectional
- ☑ External terminal

Drive mode: Resistive Pull Up

Initial drive state: High (1)

Min. supply voltage:

Max frequency: 100 MHz

☐ Hot swap

Datasheet | OK | Apply | Cancel

P1[1] (CYBSP_LED_RGB_GREEN, CYBSP_USER_LED4) - Parameters

Enter filter text...

Name	Value
Overview	
Configuration Help	Open GPIO Documentation
General	
Drive Mode	Strong Drive. Input buffer on
Initial Drive State	High (1)
Input	
Threshold	CMOS
Interrupt Trigger Type	None
Output	
Slew Rate	Fast
Drive Strength	1 / 2
Internal Connection	
Analog	<unassigned>
Digital Input	<unassigned>
Digital Output	TCPWM[0] 32-bit Counter 3 pwm_n [USED]
Digital InOut	<unassigned>
Advanced	
Store Config in Flash	☑

P1[1] (CYBSP_LED_RGB_GREEN, CYBSP_USER_LED4) - Parameters | Code Preview

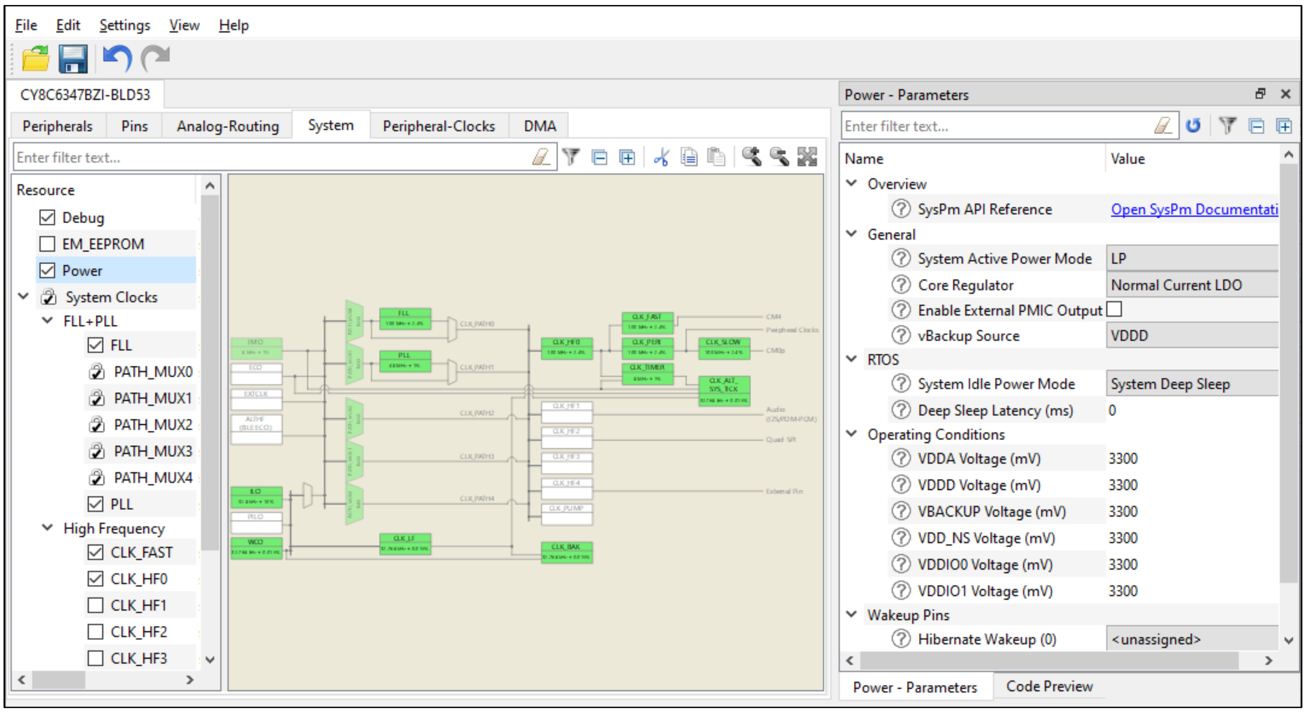
14. Go back to the PSoC™ Creator DWR Pin Editor and repeat the process with every pin.

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.

3.1 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:



Resource
☑ Debug
☐ EM_EEPROM
☑ Power
System Clocks
FLL+PLL
☑ FLL
PATH_MUX0
PATH_MUX1
PATH_MUX2
PATH_MUX3
PATH_MUX4
☑ PLL
High Frequency
☑ CLK_FAST
☑ CLK_HF0
☐ CLK_HF1
☐ CLK_HF2
☐ CLK_HF3

Name	Value
Overview	
SysPm API Reference	Open SysPm Documentati
General	
System Active Power Mode	LP
Core Regulator	Normal Current LDO
Enable External PMIC Output	☐
vBackup Source	VDDD
RTOS	
System Idle Power Mode	System Deep Sleep
Deep Sleep Latency (ms)	0
Operating Conditions	
VDDA Voltage (mV)	3300
VDDD Voltage (mV)	3300
VBACKUP Voltage (mV)	3300
VDD_NS Voltage (mV)	3300
VDDIO0 Voltage (mV)	3300
VDDIO1 Voltage (mV)	3300
Wakeup Pins	
Hibernate Wakeup (0)	<unassigned>

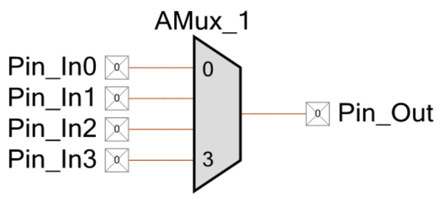
3.2 Analog Routing

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.



3.3 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.

3.4 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.

3.4.1 Interrupts Internal to Component Code (Bluetooth® LE, 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);
}
```

3.4.2 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/PSoC™6/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

```
#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.in
trSrc);

    /* 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(&MCWDT_isr_cfg,
&MCWDT_Interrupt_Handler);
    /* Clear CM4 NVIC pending
interrupt for MCWDT */

NVIC_ClearPendingIRQ(MCWDT_isr_cfg.in
trSrc);
    /* Enable CM4 NVIC MCWDT
interrupt */

NVIC_EnableIRQ(MCWDT_isr_cfg.intrSrc)
;
}
```

ModusToolbox™ 3.x

```
#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_IRQ
n,
        .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.in
trSrc);

    /* Interrupt processing */
}

void MCWDT_Interrupt_Init(void)
{
    Cy_SysInt_Init(&MCWDT_isr_cfg,
&MCWDT_Interrupt_Handler);
    /* Clear CM4 NVIC pending
interrupt for MCWDT */

NVIC_ClearPendingIRQ(MCWDT_isr_cfg.in
trSrc);
    /* Enable CM4 NVIC MCWDT
interrupt */

NVIC_EnableIRQ(MCWDT_isr_cfg.intrSrc)
;
}
```

3.4.3 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™

4.1 Common Component Settings

4.1.1 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.

4.1.2 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.

4.1.3 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.

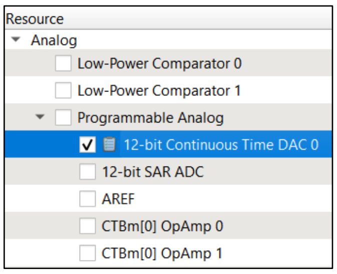
4.1.4 Aliases

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

4.2 Voltage DAC (12-bit)

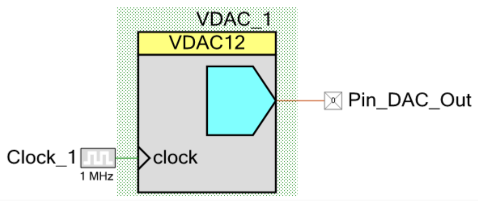
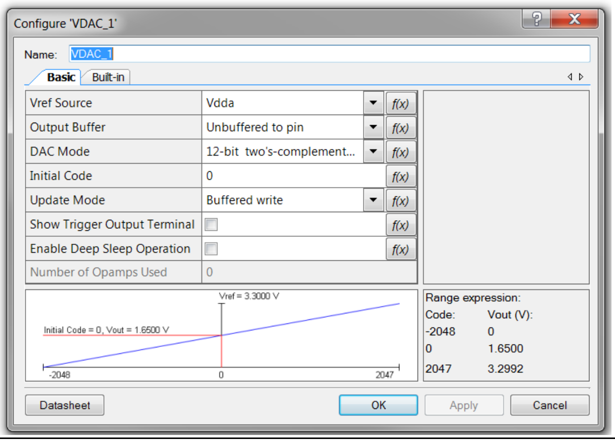
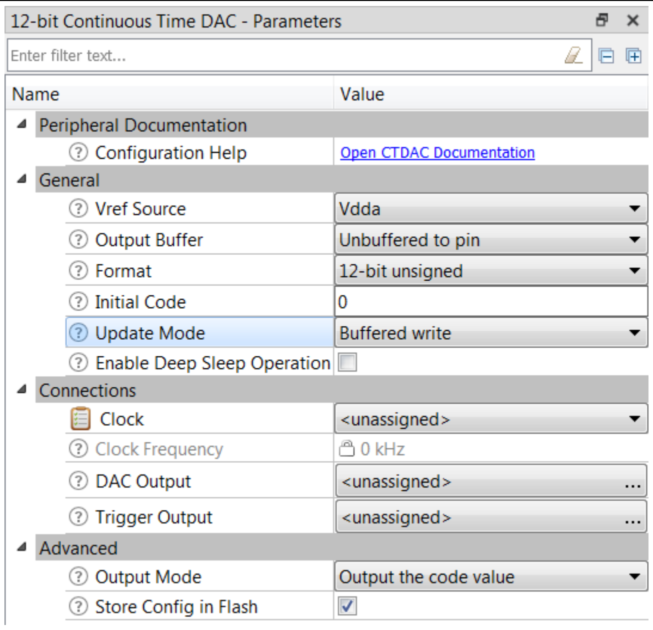
Follow these steps to transition the PSoC™ Creator Voltage DAC (12-bit) to ModusToolbox™:

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



Resource
- Analog
  - ☐ Low-Power Comparator 0
  - ☐ Low-Power Comparator 1
  - ☐ Programmable Analog
    - ☑ 12-bit Continuous Time DAC 0
    - ☐ 12-bit SAR ADC
    - ☐ AREF
    - ☐ CTBm[0] OpAmp 0
    - ☐ CTBm[0] OpAmp 1

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

PSoC™ Creator	ModusToolbox™ Device Configurator
 	

Configure 'VDAC_1'

Name: VDAC_1

Basic | Built-in

Vref Source	Vdda
Output Buffer	Unbuffered to pin
DAC Mode	12-bit two's-complement...
Initial Code	0
Update Mode	Buffered write
Show Trigger Output Terminal	☐
Enable Deep Sleep Operation	☐
Number of Opamps Used	0

Range expression:

Code:	Vout (V):
-2048	0
0	1.6500
2047	3.2992

12-bit Continuous Time DAC - Parameters

Enter filter text...

Name	Value
Peripheral Documentation	
Configuration Help	Open CTDAC Documentation
General	
Vref Source	Vdda
Output Buffer	Unbuffered to pin
Format	12-bit unsigned
Initial Code	0
Update Mode	Buffered write
Enable Deep Sleep Operation	☐
Connections	
Clock	<unassigned>
Clock Frequency	0 kHz
DAC Output	<unassigned>
Trigger Output	<unassigned>
Advanced	
Output Mode	Output the code value
Store Config in Flash	☑

– 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.

– 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.

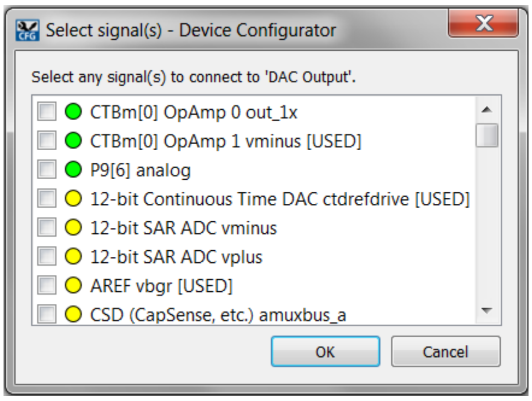
– **DAC Mode** has been renamed to **Format**.

– **Sample and Hold** (if visible)

– **Initial Code**

- 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.
- **Show Trigger Output Terminal** is not transferred because it only controls wire terminal visibility in PSoC™ Creators schematic view.
- **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).


Select signal(s) - Device Configurator

Select any signal(s) to connect to 'DAC Output'.

- ☐ 🟢 CTBm[0] OpAmp 0 out_1x
- ☐ 🟢 CTBm[0] OpAmp 1 vminus [USED]
- ☐ 🟢 P9[6] analog
- ☐ 🟡 12-bit Continuous Time DAC ctdrefdrive [USED]
- ☐ 🟡 12-bit SAR ADC vminus
- ☐ 🟡 12-bit SAR ADC vplus
- ☐ 🟡 AREF vbgr [USED]
- ☐ 🟡 CSD (CapSense, etc.) amuxbus_a

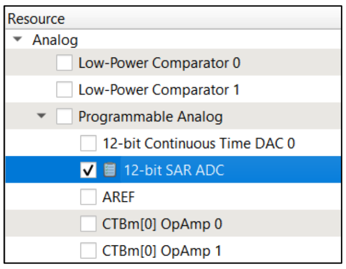
If the **Output Buffer** parameter is set to **Buffered**, **CTBm[0] OpAmp 0 Vplus** must be selected as the output destination. The dedicated output pin is device specific and can be referenced in the device datasheet.

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.

4.3 Scanning SAR ADC

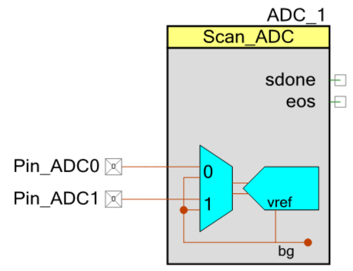
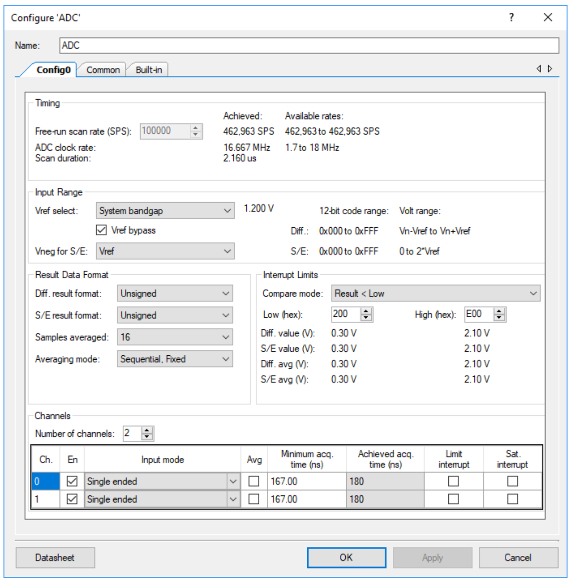
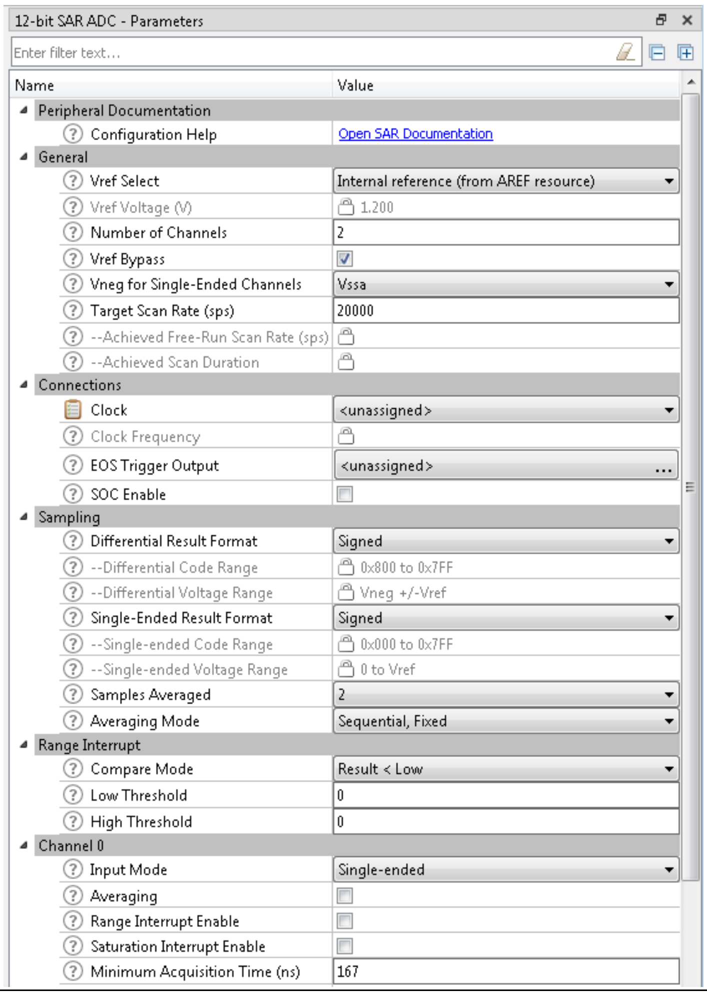
Follow these steps to transition the PSoC™ Creator Scanning SAR ADC to ModusToolbox™:

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



Resource
- Analog
  - ☐ Low-Power Comparator 0
  - ☐ Low-Power Comparator 1
  - ☐ Programmable Analog
    - ☐ 12-bit Continuous Time DAC 0
    - ☑ 12-bit SAR ADC
    - ☐ AREF
    - ☐ CTBm[0] OpAmp 0
    - ☐ CTBm[0] OpAmp 1

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

PSoC™ Creator	ModusToolbox™ Device Configurator
 	

Configure 'ADC'

Name: ADC

Config0 | Common | Built-in

Timing

		Achieved:	Available rates:
Free-run scan rate (SPS):	100000	462,963 SPS	462,963 to 462,963 SPS
ADC clock rate:		16.667 MHz	1.7 to 18 MHz
Scan duration:		2.160 us	

Input Range

				12-bit code range:	Volt range:
Vref select:	System bandgap	1.200 V			
	☑ Vref bypass		Diff.:	0x000 to 0xFFF	Vn-Vref to Vn+Vref
Vneg for S/E:	Vref		S/E:	0x000 to 0xFFF	0 to 2*Vref

Result Data Format

Diff. result format:	Unsigned
S/E result format:	Unsigned
Samples averaged:	16
Averaging mode:	Sequential, Fixed

Interrupt Limits

Compare mode: Result < Low

Low (hex):	200	High (hex):	E00
Diff. value (V):	0.30 V		2.10 V
S/E value (V):	0.30 V		2.10 V
Diff. avg (V):	0.30 V		2.10 V
S/E avg (V):	0.30 V		2.10 V

Channels

Number of channels: 2

Ch.	En	Input mode	Avg	Minimum acq. time (ns)	Achieved acq. time (ns)	Limit interrupt	Sat. interrupt
0	☑	Single ended	☐	167.00	180	☐	☐
1	☑	Single ended	☐	167.00	180	☐	☐

Datasheet | OK | Apply | Cancel

12-bit SAR ADC - Parameters

Enter filter text...

Name	Value
Peripheral Documentation	
Configuration Help	Open SAR Documentation
General	
Vref Select	Internal reference (from AREF resource)
Vref Voltage (V)	1.200
Number of Channels	2
Vref Bypass	☑
Vneg for Single-Ended Channels	Vssa
Target Scan Rate (sps)	20000
--Achieved Free-Run Scan Rate (sps)	
--Achieved Scan Duration	
Connections	
Clock	<unassigned>
Clock Frequency	
EOS Trigger Output	<unassigned>
SOC Enable	☐
Sampling	
Differential Result Format	Signed
--Differential Code Range	0x800 to 0x7FF
--Differential Voltage Range	Vneg +/-Vref
Single-Ended Result Format	Signed
--Single-ended Code Range	0x000 to 0x7FF
--Single-ended Voltage Range	0 to Vref
Samples Averaged	2
Averaging Mode	Sequential, Fixed
Range Interrupt	
Compare Mode	Result < Low
Low Threshold	0
High Threshold	0
Channel 0	
Input Mode	Single-ended
Averaging	☐
Range Interrupt Enable	☐
Saturation Interrupt Enable	☐
Minimum Acquisition Time (ns)	167

- **Vref Select**
- **Number of Channels**
- **Vref Bypass**
- 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.

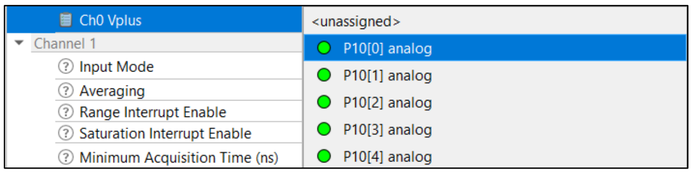
- PSoC™ Creator **Free-run scan rate (SPS)** converts to ModusToolbox™ **Target Scan Rate (sps)**. PSoC™ Creator automatically selects and configures a clock source to generate the scan rate. In ModusToolbox™ you must select a Clock Connection and configure its source and divider to generate the frequency required for the scan rate. See the SAR PDL documentation for details on determining the correct clock frequency.
- 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**.
- **Differential Result Format**
- **Single-Ended Result Format**
- **Samples Averaged**
- **Averaging Mode**
- **Compare Mode**
- **Low Threshold**
- **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.
   - The **Show analog clock (aclk) terminal** check box does not transfer to ModusToolbox™ because the analog clock must always be manually configured.
   - 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.
   - 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.
   - 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.

5. 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.



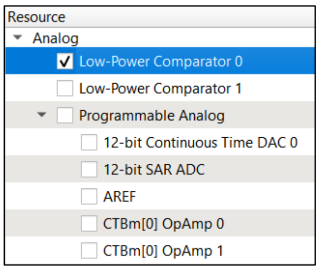
Ch0 Vplus	<unassigned>
Channel 1	● P10[0] analog
Input Mode	● P10[1] analog
Averaging	● P10[2] analog
Range Interrupt Enable	● P10[3] analog
Saturation Interrupt Enable	● P10[4] analog
Minimum Acquisition Time (ns)	

6. 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.
7. 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.

4.4 Low-Power Comparator

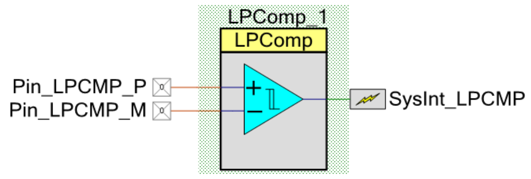
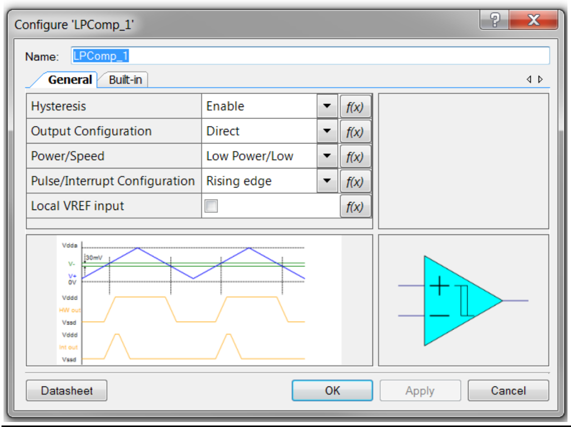
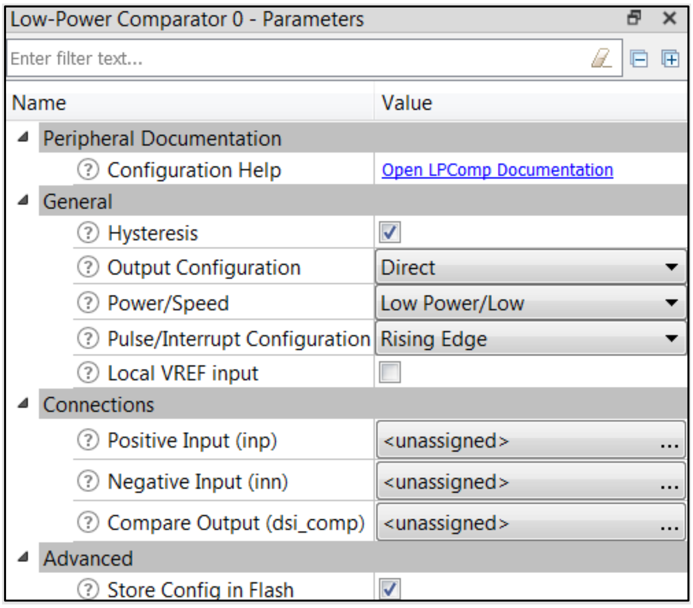
Follow these steps to transition the PSoC™ Creator Low Power Comparator to ModusToolbox™:

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



Resource
▾ Analog
☑ Low-Power Comparator 0
☐ Low-Power Comparator 1
▾ ☐ Programmable Analog
☐ 12-bit Continuous Time DAC 0
☐ 12-bit SAR ADC
☐ AREF
☐ CTBm[0] OpAmp 0
☐ CTBm[0] OpAmp 1

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

PSoC™ Creator	ModusToolbox™ Device Configurator
 	

Configure 'LPComp_1'

Name: LPComp_1

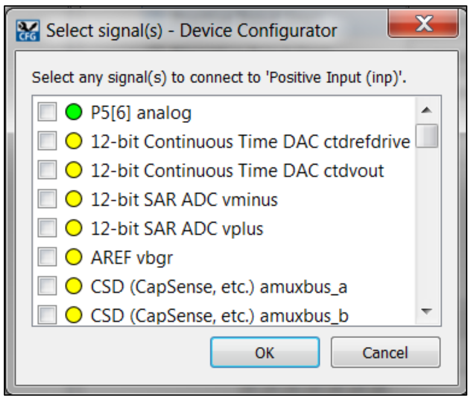
Hysteresis	Enable
Output Configuration	Direct
Power/Speed	Low Power/Low
Pulse/Interrupt Configuration	Rising edge
Local VREF input	☐

Low-Power Comparator 0 - Parameters

Name	Value
Peripheral Documentation	
Configuration Help	Open LPComp Documentation
General	
Hysteresis	☑
Output Configuration	Direct
Power/Speed	Low Power/Low
Pulse/Interrupt Configuration	Rising Edge
Local VREF input	☐
Connections	
Positive Input (inp)	<unassigned>
Negative Input (inn)	<unassigned>
Compare Output (dsi_comp)	<unassigned>
Advanced	
Store Config in Flash	☑

– Hysteresis
– Output Configuration
– Power/Speed
– Pulse/Interrupt Configuration
– 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.



Select signal(s) - Device Configurator

Select any signal(s) to connect to 'Positive Input (inp)'.

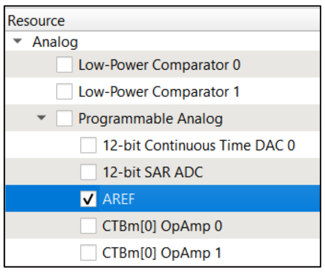
- ☐ 🟢 P5[6] analog
- ☐ 🟡 12-bit Continuous Time DAC ctdrefdrive
- ☐ 🟡 12-bit Continuous Time DAC ctdvout
- ☐ 🟡 12-bit SAR ADC vminus
- ☐ 🟡 12-bit SAR ADC vplus
- ☐ 🟡 AREF vbgr
- ☐ 🟡 CSD (CapSense, etc.) amuxbus_a
- ☐ 🟡 CSD (CapSense, etc.) amuxbus_b

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.

4.5 Analog Reference (AREF)

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

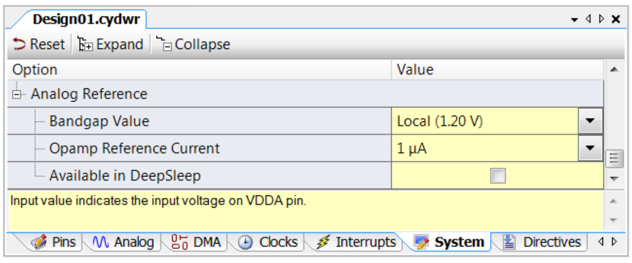
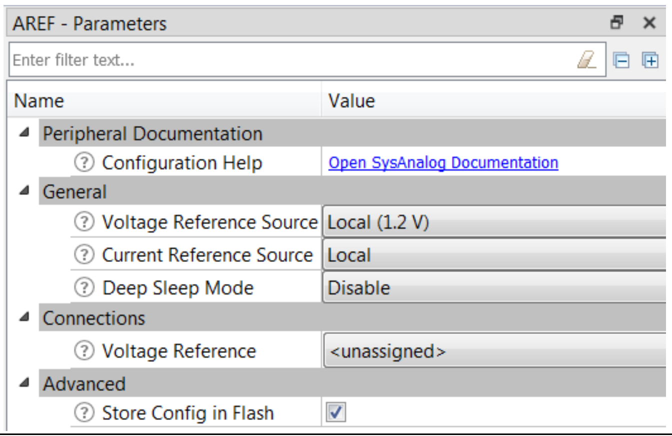
1. All designs that use any analog resource must enable the Analog Reference resource.



Resource
- Analog
  - [ ] Low-Power Comparator 0
  - [ ] Low-Power Comparator 1
  - [ ] Programmable Analog
    - [ ] 12-bit Continuous Time DAC 0
    - [ ] 12-bit SAR ADC
    - [✓] AREF
    - [ ] CTBm[0] OpAmp 0
    - [ ] CTBm[0] OpAmp 1

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.

PSoC™ Creator	ModusToolbox™ Device Configurator
	

PSoC™ Creator (Design01.cydwr):

Option	Value
Analog Reference	
Bandgap Value	Local (1.20 V)
Opamp Reference Current	1 µA
Available in DeepSleep	☐

Input value indicates the input voltage on VDDA pin.

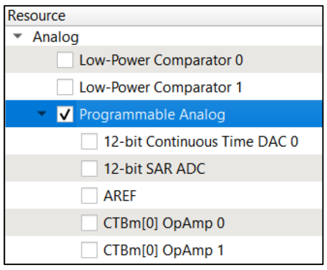
ModusToolbox™ Device Configurator (AREF - Parameters):

Name	Value
Peripheral Documentation	
Configuration Help	Open SysAnalog Documentation
General	
Voltage Reference Source	Local (1.2 V)
Current Reference Source	Local
Deep Sleep Mode	Disable
Connections	
Voltage Reference	<unassigned>
Advanced	
Store Config in Flash	✓

4.6 Programmable Analog

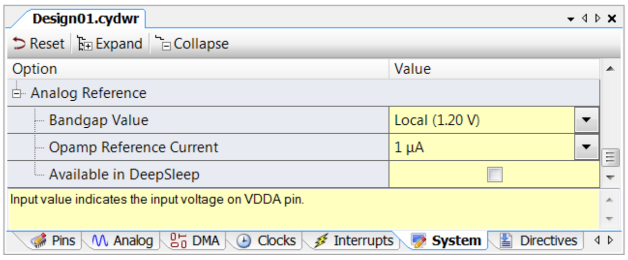
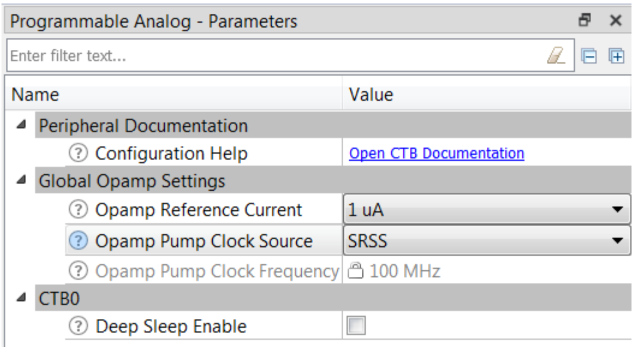
Follow these steps to transition the PSoC™ Creator Programable Analog settings to ModusToolbox™:

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


Resource
- Analog
  - [ ] Low-Power Comparator 0
  - [ ] Low-Power Comparator 1
  - [✓] Programmable Analog
    - [ ] 12-bit Continuous Time DAC 0
    - [ ] 12-bit SAR ADC
    - [ ] AREF
    - [ ] CTBm[0] OpAmp 0
    - [ ] CTBm[0] OpAmp 1

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.

PSoC™ Creator	ModusToolbox™ Device Configurator
	

PSoC™ Creator (Design01.cydwr):

Option	Value
Analog Reference	
Bandgap Value	Local (1.20 V)
Opamp Reference Current	1 µA
Available in DeepSleep	☐

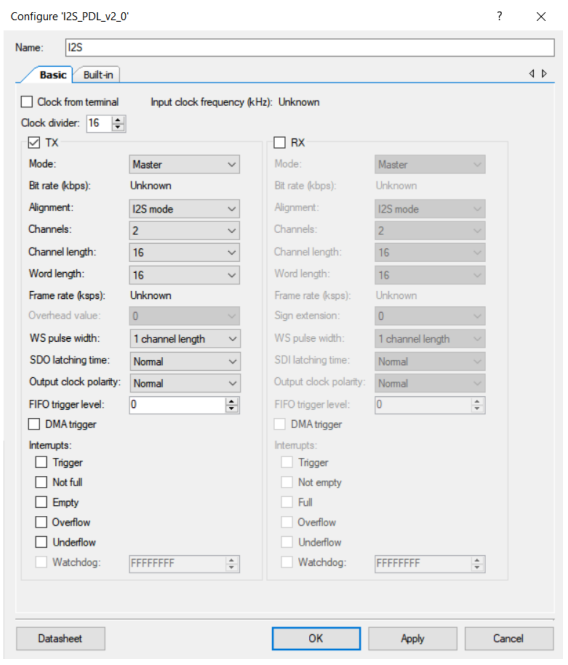
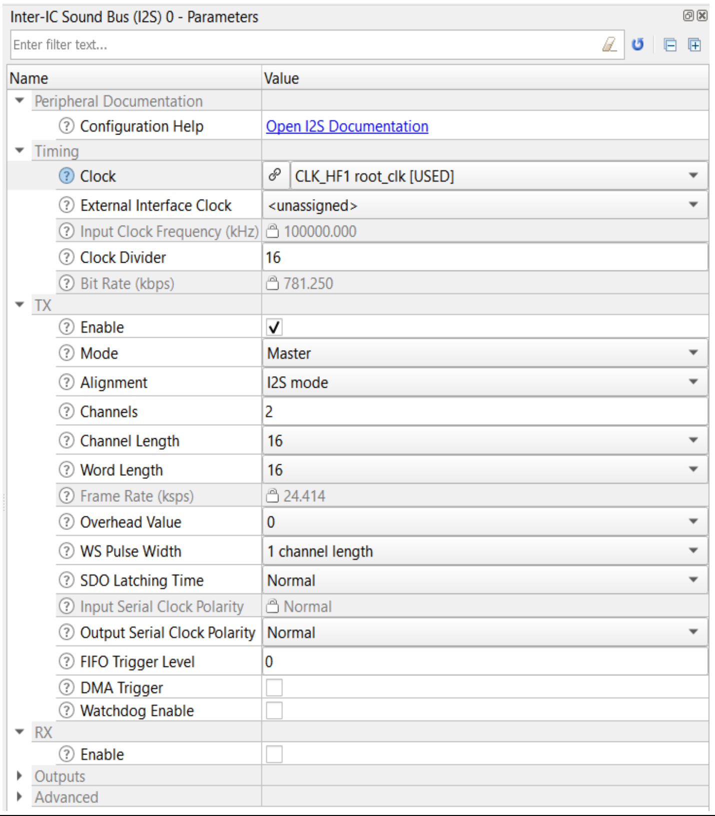
Input value indicates the input voltage on VDDA pin.

ModusToolbox™ Device Configurator (Programmable Analog - Parameters):

Name	Value
Peripheral Documentation	
Configuration Help	Open CTB Documentation
Global Opamp Settings	
Opamp Reference Current	1 uA
Opamp Pump Clock Source	SRSS
Opamp Pump Clock Frequency	100 MHz
CTB0	
Deep Sleep Enable	☐

4.7 I2S

Replicate the settings as appropriate.

PSoC™ Creator	ModusToolbox™ Device Configurator
	

Configure 'I2S_PDL_v2_0'

Name: I2S

Basic | Built-in

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

Clock divider: 16

☑ TX		☐ RX	
Mode:	Master	Mode:	Master
Bit rate (kbps):	Unknown	Bit rate (kbps):	Unknown
Alignment:	I2S mode	Alignment:	I2S mode
Channels:	2	Channels:	2
Channel length:	16	Channel length:	16
Word length:	16	Word length:	16
Frame rate (ksps):	Unknown	Frame rate (ksps):	Unknown
Overhead value:	0	Sign extension:	0
WS pulse width:	1 channel length	WS pulse width:	1 channel length
SDO latching time:	Normal	SDI latching time:	Normal
Output clock polarity:	Normal	Output clock polarity:	Normal
FIFO trigger level:	0	FIFO trigger level:	0
☐ DMA trigger		☐ DMA trigger	
Interrupts:		Interrupts:	
☐ Trigger		☐ Trigger	
☐ Not full		☐ Not empty	
☐ Empty		☐ Full	
☐ Overflow		☐ Overflow	
☐ Underflow		☐ Underflow	
☐ Watchdog:	FFFFFFFF	☐ Watchdog:	FFFFFFFF

Inter-IC Sound Bus (I2S) 0 - Parameters

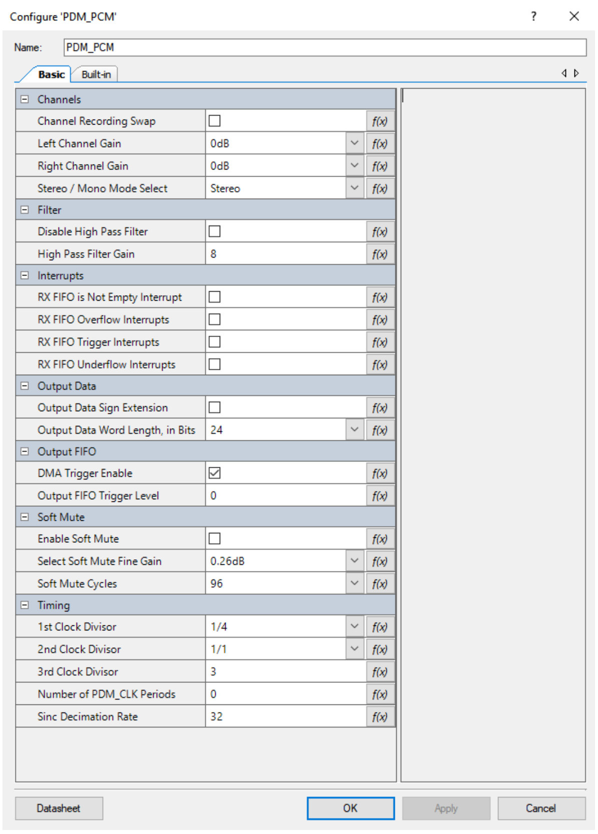
Enter filter text...

Name	Value
Peripheral Documentation	
Configuration Help	Open I2S Documentation
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	✓
Mode	Master
Alignment	I2S mode
Channels	2
Channel Length	16
Word Length	16
Frame Rate (ksps)	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	☐
Watchdog Enable	☐
RX	
Enable	☐
Outputs	
Advanced	

4.8 PDM-PCM

Replicate the settings as appropriate.

PSoC™ Creator



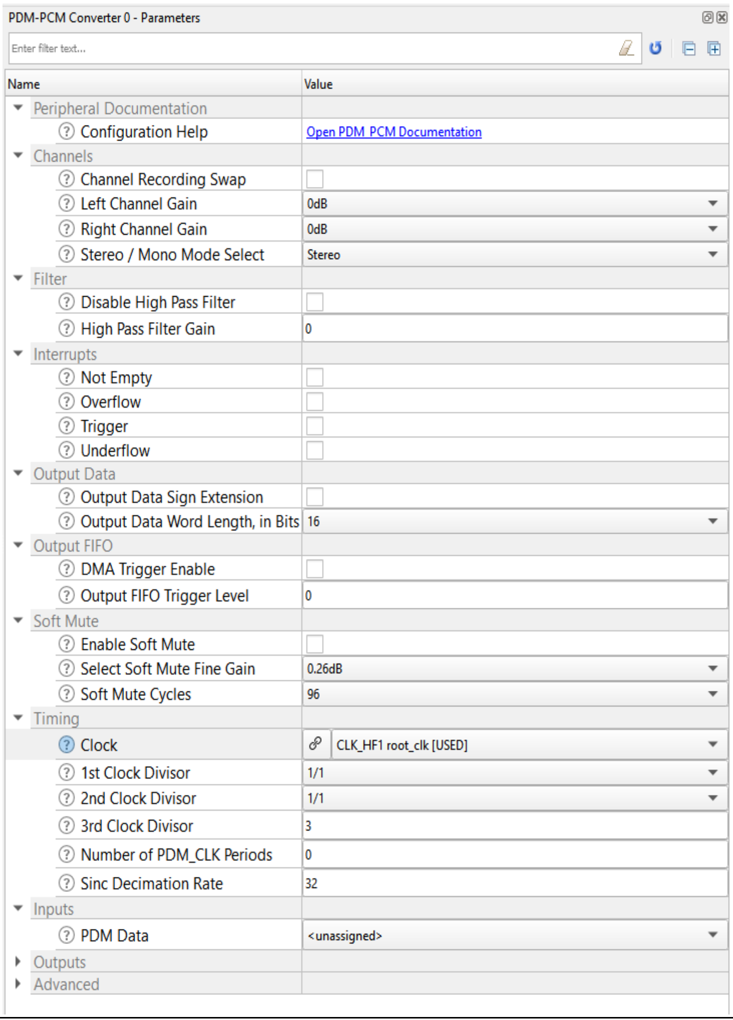
Configure 'PDM_PCM'

Name: PDM_PCM

Basic | Built-in

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

ModusToolbox™ Device Configurator



PDM-PCM Converter 0 - Parameters

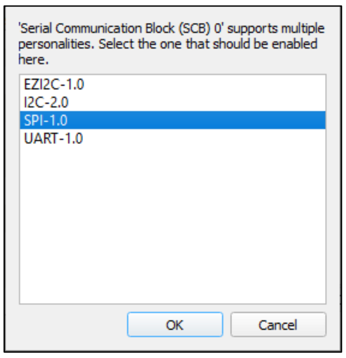
Enter filter text...

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

4.9 SCB

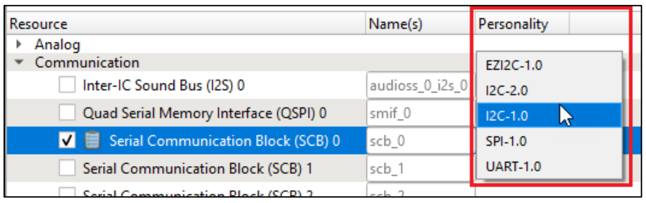
4.9.1 Personalities

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:


'Serial Communication Block (SCB) 0' supports multiple personalities. Select the one that should be enabled here.

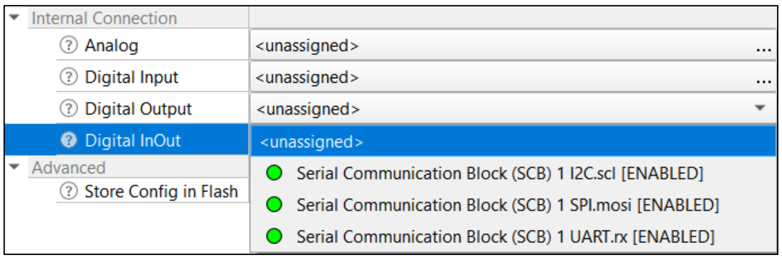
EZI2C-1.0
I2C-2.0
SPI-1.0
UART-1.0

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


Resource	Name(s)	Personality
Analog		
Communication		
Inter-IC Sound Bus (I2S) 0	audioss_0_i2s_0	
Quad Serial Memory Interface (QSPI) 0	smif_0	
✓ Serial Communication Block (SCB) 0	scb_0	
Serial Communication Block (SCB) 1	scb_1	

4.9.2 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	

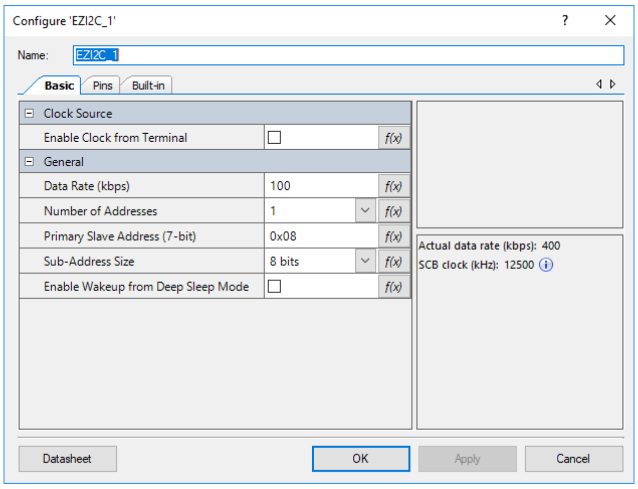
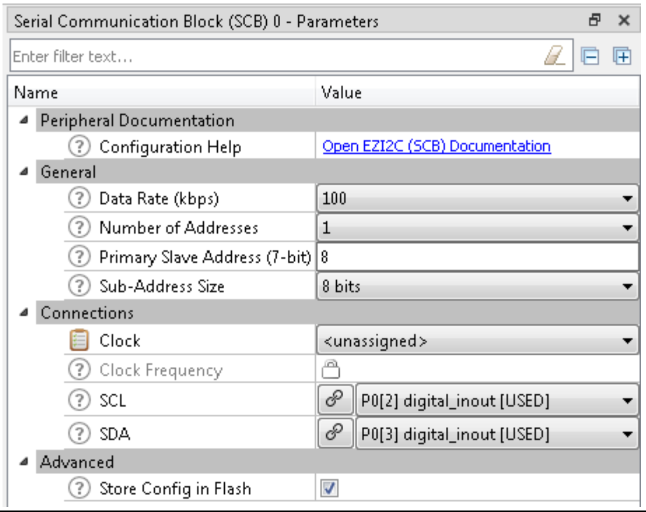
● Serial Communication Block (SCB) 1 I2C.scl [ENABLED]
● Serial Communication Block (SCB) 1 SPI.mosi [ENABLED]
● Serial Communication Block (SCB) 1 UART.rx [ENABLED]

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.

4.9.3 EZI2C

Replicate the settings as appropriate.

PSoC™ Creator	ModusToolbox™ Device Configurator
	

Configure 'EZI2C_1'

Name: EZI2C_1

Basic | Pins | Built-in

Clock Source	
Enable Clock from Terminal	☐
General	
Data Rate (kbps)	100
Number of Addresses	1
Primary Slave Address (7-bit)	0x08
Sub-Address Size	8 bits
Enable Wakeup from Deep Sleep Mode	☐

Actual data rate (kbps): 400
SCB clock (kHz): 12500

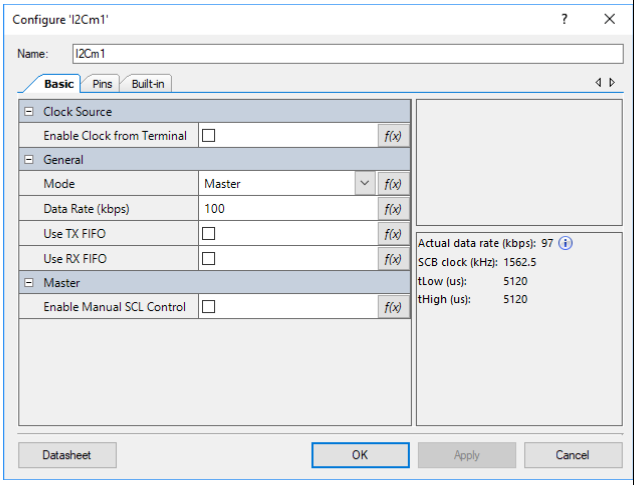
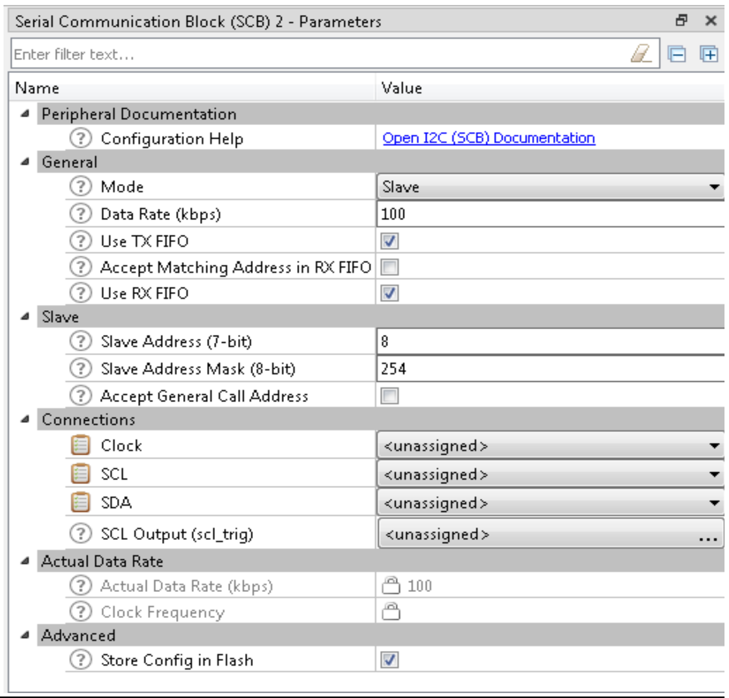
Datasheet | OK | Apply | Cancel

Serial Communication Block (SCB) 0 - Parameters

Name	Value
Peripheral Documentation	
Configuration Help	Open EZI2C (SCB) Documentation
General	
Data Rate (kbps)	100
Number of Addresses	1
Primary Slave Address (7-bit)	8
Sub-Address Size	8 bits
Connections	
Clock	<unassigned>
Clock Frequency	
SCL	P0[2] digital_inout [USED]
SDA	P0[3] digital_inout [USED]
Advanced	
Store Config in Flash	✓

4.9.4 I2C

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

PSoC™ Creator	ModusToolbox™ Device Configurator
	

Configure 'I2Cm1'

Name: I2Cm1

Basic | Pins | Built-in

Clock Source	
Enable Clock from Terminal	☐
General	
Mode	Master
Data Rate (kbps)	100
Use TX FIFO	☐
Use RX FIFO	☐
Master	
Enable Manual SCL Control	☐

Actual data rate (kbps): 97
SCB clock (kHz): 1562.5
tLow (us): 5120
tHigh (us): 5120

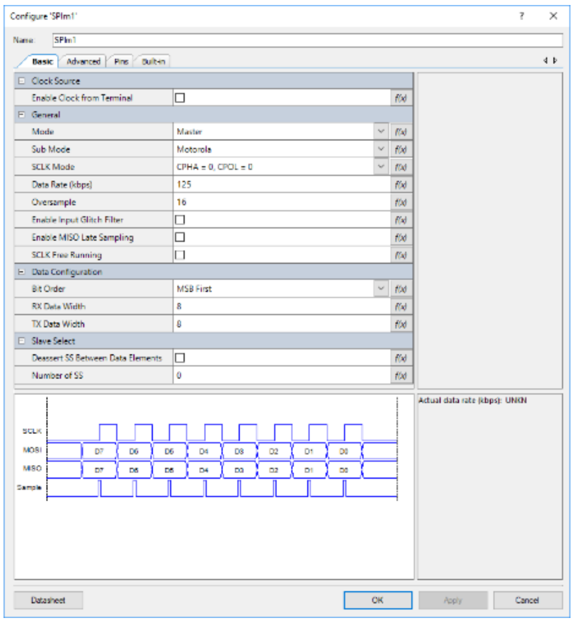
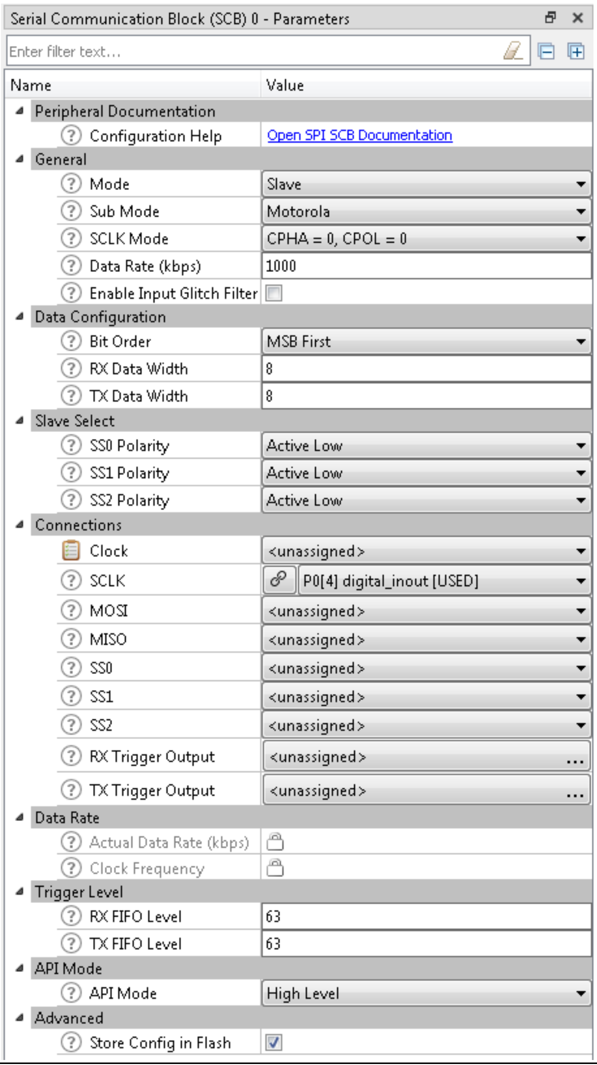
Datasheet | OK | Apply | Cancel

Serial Communication Block (SCB) 2 - Parameters

Name	Value
Peripheral Documentation	
Configuration Help	Open I2C (SCB) Documentation
General	
Mode	Slave
Data Rate (kbps)	100
Use TX FIFO	✓
Accept Matching Address in RX FIFO	☐
Use RX FIFO	✓
Slave	
Slave Address (7-bit)	8
Slave Address Mask (8-bit)	254
Accept General Call Address	☐
Connections	
Clock	<unassigned>
SCL	<unassigned>
SDA	<unassigned>
SCL Output (scl_trig)	<unassigned>
Actual Data Rate	
Actual Data Rate (kbps)	100
Clock Frequency	
Advanced	
Store Config in Flash	✓

4.9.5 SPI

Replicate the settings as appropriate.

PSoC™ Creator	ModusToolbox™ Device Configurator
	

Configure 'SPIm1'

Name: SPIm1

Basic | Advanced | Pins | Built-in

Clock Source	
Enable Clock from Terminal	☐
General	
Mode	Master
Sub Mode	Motorola
SCLK Mode	CPHA = 0, CPOL = 0
Data Rate (kbps)	125
Oversample	16
Enable Input Glitch Filter	☐
Enable MISO Late Sampling	☐
SCLK Free Running	☐
Data Configuration	
Bit Order	MSB First
RX Data Width	8
TX Data Width	8
Slave Select	
Deassert SS Between Data Elements	☐
Number of SS	0

Actual data rate (kbps): UNKN

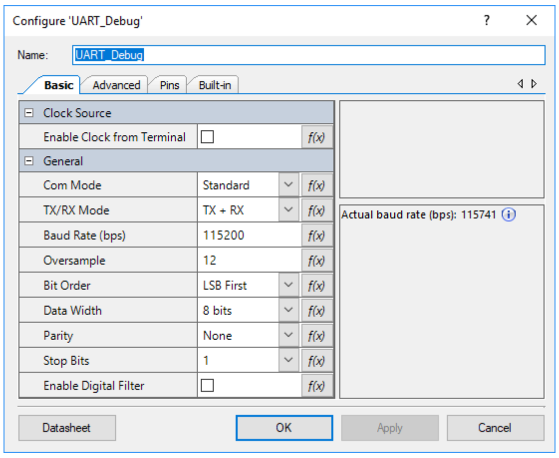
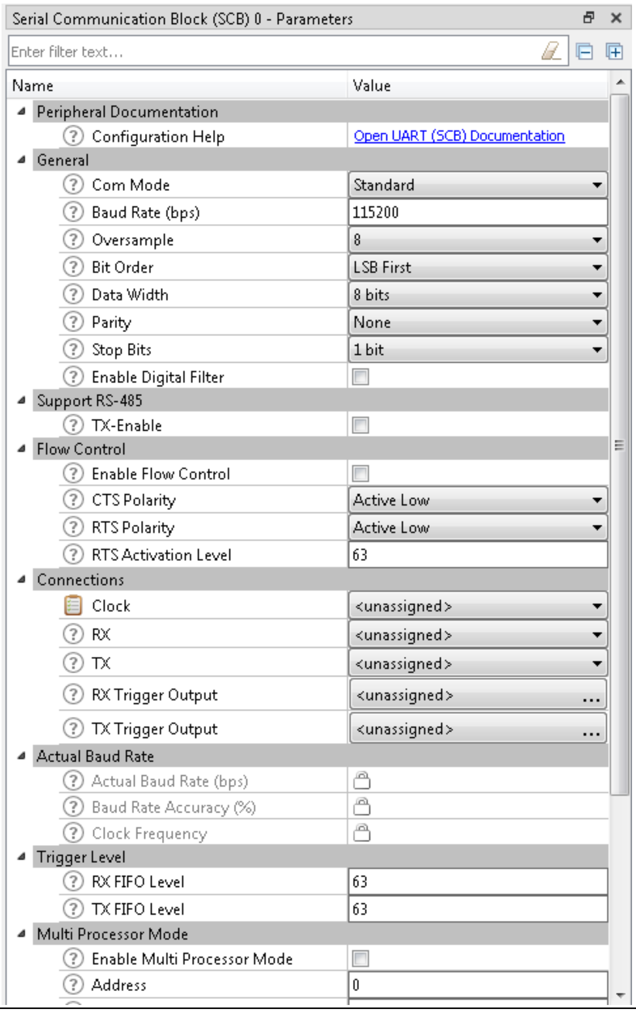
Serial Communication Block (SCB) 0 - Parameters

Enter filter text…

Name	Value
Peripheral Documentation	
Configuration Help	Open SPI SCB Documentation
General	
Mode	Slave
Sub Mode	Motorola
SCLK Mode	CPHA = 0, CPOL = 0
Data Rate (kbps)	1000
Enable Input Glitch Filter	☐
Data Configuration	
Bit Order	MSB First
RX Data Width	8
TX Data Width	8
Slave Select	
SS0 Polarity	Active Low
SS1 Polarity	Active Low
SS2 Polarity	Active Low
Connections	
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>
Data Rate	
Actual Data Rate (kbps)	
Clock Frequency	
Trigger Level	
RX FIFO Level	63
TX FIFO Level	63
API Mode	
API Mode	High Level
Advanced	
Store Config in Flash	☑

4.9.6 UART

Replicate the settings as appropriate.

PSoC™ Creator	ModusToolbox™ Device Configurator
	

Configure 'UART_Debug'

Name: UART_Debug

Basic | Advanced | Pins | Built-in

Clock Source		
Enable Clock from Terminal	☐	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	☐	f(x)

Actual baud rate (bps): 115741

Datasheet | OK | Apply | Cancel

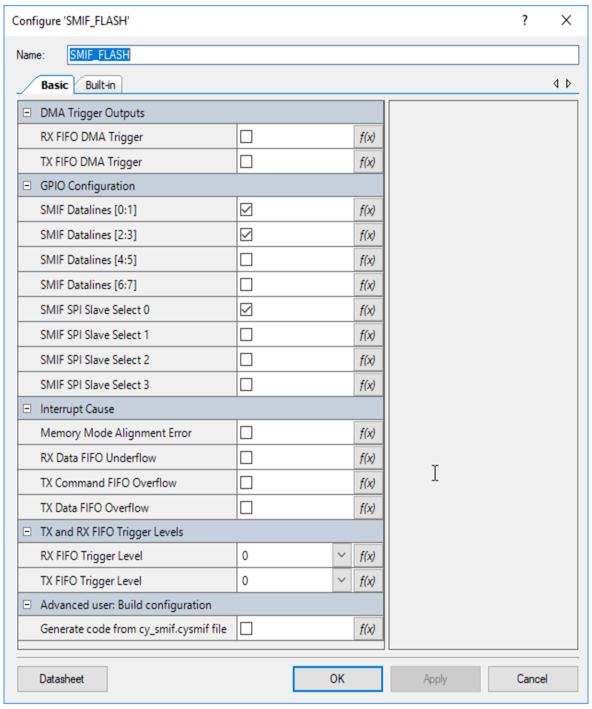
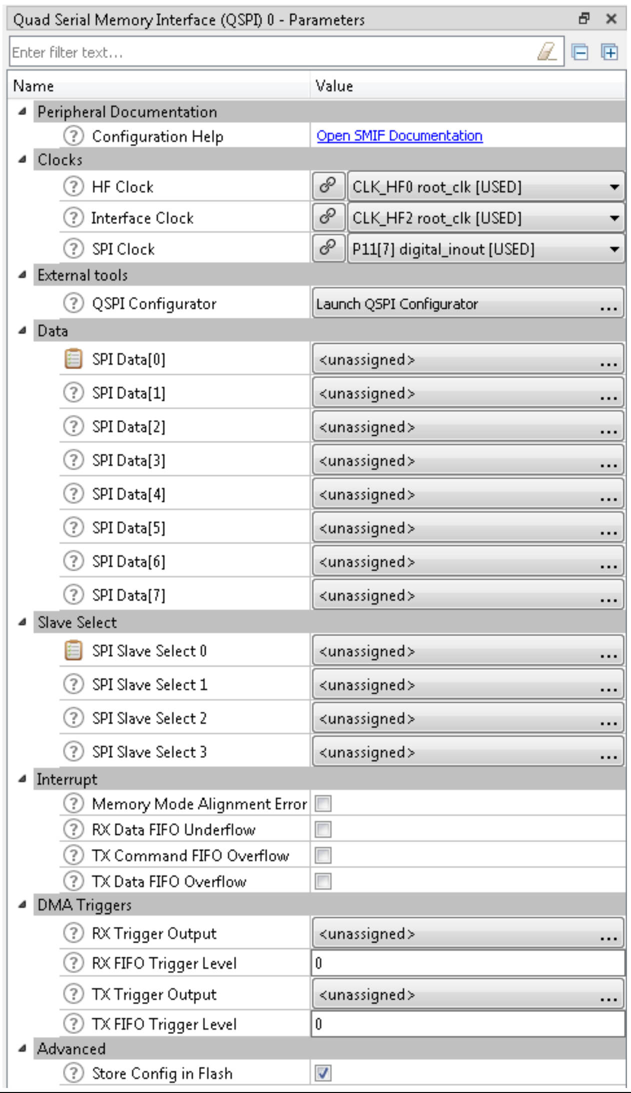
Serial Communication Block (SCB) 0 - Parameters

Enter filter text...

Name	Value
Peripheral Documentation	
Configuration Help	Open UART (SCB) Documentation
General	
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	☐
Support RS-485	
TX-Enable	☐
Flow Control	
Enable Flow Control	☐
CTS Polarity	Active Low
RTS Polarity	Active Low
RTS Activation Level	63
Connections	
Clock	<unassigned>
RX	<unassigned>
TX	<unassigned>
RX Trigger Output	<unassigned>
TX Trigger Output	<unassigned>
Actual Baud Rate	
Actual Baud Rate (bps)	
Baud Rate Accuracy (%)	
Clock Frequency	
Trigger Level	
RX FIFO Level	63
TX FIFO Level	63
Multi Processor Mode	
Enable Multi Processor Mode	☐
Address	0

4.10 SMIF (QSPI)

Replicate the settings as appropriate. Refer also to the ModusToolbox™ QSPI Configurator user guide, as needed.

PSoC™ Creator	ModusToolbox™ Device Configurator
	

Configure 'SMIF_FLASH'

Name: SMIF_FLASH

DMA Trigger Outputs	
RX FIFO DMA Trigger	☐
TX FIFO DMA Trigger	☐
GPIO Configuration	
SMIF Datalines [0:1]	☑
SMIF Datalines [2:3]	☑
SMIF Datalines [4:5]	☐
SMIF Datalines [6:7]	☐
SMIF SPI Slave Select 0	☑
SMIF SPI Slave Select 1	☐
SMIF SPI Slave Select 2	☐
SMIF SPI Slave Select 3	☐
Interrupt Cause	
Memory Mode Alignment Error	☐
RX Data FIFO Underflow	☐
TX Command FIFO Overflow	☐
TX Data FIFO Overflow	☐
TX and RX FIFO Trigger Levels	
RX FIFO Trigger Level	0
TX FIFO Trigger Level	0
Advanced user: Build configuration	
Generate code from cy_smif.cysmif file	☐

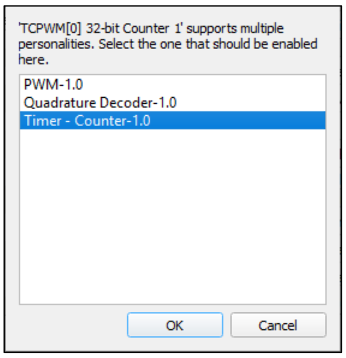
Quad Serial Memory Interface (QSPI) 0 - Parameters

Name	Value
Peripheral Documentation	
Configuration Help	Open SMIF Documentation
Clocks	
HF Clock	CLK_HF0 root_clk [USED]
Interface Clock	CLK_HF2 root_clk [USED]
SPI Clock	P11[7] digital_inout [USED]
External tools	
QSPI Configurator	Launch QSPI Configurator
Data	
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>
Slave Select	
SPI Slave Select 0	<unassigned>
SPI Slave Select 1	<unassigned>
SPI Slave Select 2	<unassigned>
SPI Slave Select 3	<unassigned>
Interrupt	
Memory Mode Alignment Error	☐
RX Data FIFO Underflow	☐
TX Command FIFO Overflow	☐
TX Data FIFO Overflow	☐
DMA Triggers	
RX Trigger Output	<unassigned>
RX FIFO Trigger Level	0
TX Trigger Output	<unassigned>
TX FIFO Trigger Level	0
Advanced	
Store Config in Flash	☑

4.11 TCPWM

4.11.1 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:

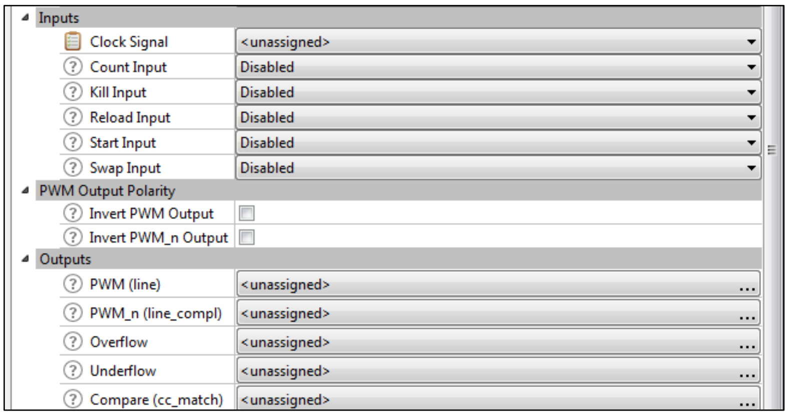

'TCPWM[0] 32-bit Counter 1' supports multiple personalities. Select the one that should be enabled here.

PWM-1.0
Quadrature Decoder-1.0
Timer - Counter-1.0

4.11.2 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:



Inputs	
Clock Signal	<unassigned>
Count Input	Disabled
Kill Input	Disabled
Reload Input	Disabled
Start Input	Disabled
Swap Input	Disabled
PWM Output Polarity	
Invert PWM Output	☐
Invert PWM_n Output	☐
Outputs	
PWM (line)	<unassigned> ...
PWM_n (line_compl)	<unassigned> ...
Overflow	<unassigned> ...
Underflow	<unassigned> ...
Compare (cc_match)	<unassigned> ...

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.

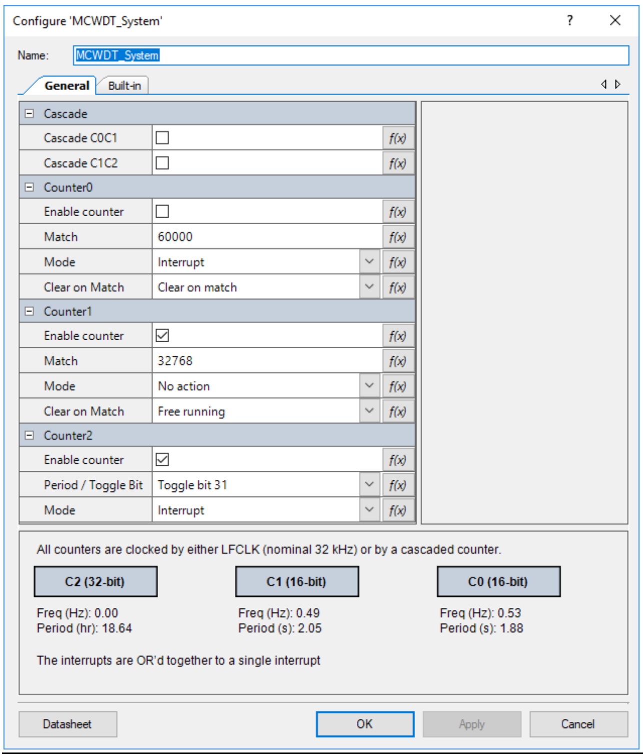
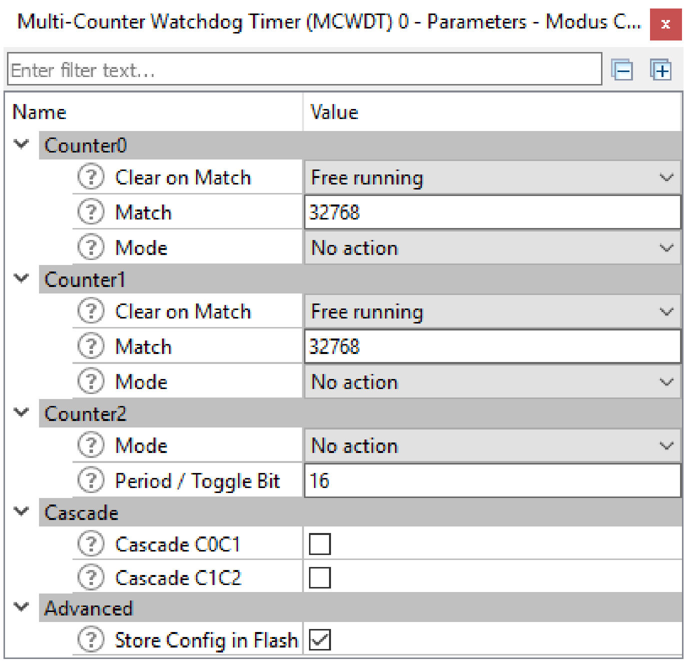
In ModusToolbox™, you must select the correct TCPWM for your application. The easiest way to do this is:

- 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.

4.12 MCWDT

Replicate the settings as appropriate.

PSoC™ Creator	ModusToolbox™ Device Configurator
	

Configure 'MCWDT_System'

Name: MCWDT_System

General | Built-in

Cascade	
Cascade C0C1	☐
Cascade C1C2	☐
Counter0	
Enable counter	☐
Match	60000
Mode	Interrupt
Clear on Match	Clear on match
Counter1	
Enable counter	☑
Match	32768
Mode	No action
Clear on Match	Free running
Counter2	
Enable counter	☑
Period / Toggle Bit	Toggle bit 31
Mode	Interrupt

All counters are clocked by either LFCLK (nominal 32 kHz) or by a cascaded counter.

C2 (32-bit)	C1 (16-bit)	C0 (16-bit)
Freq (Hz): 0.00	Freq (Hz): 0.49	Freq (Hz): 0.53
Period (hr): 18.64	Period (s): 2.05	Period (s): 1.88

The interrupts are OR'd together to a single interrupt

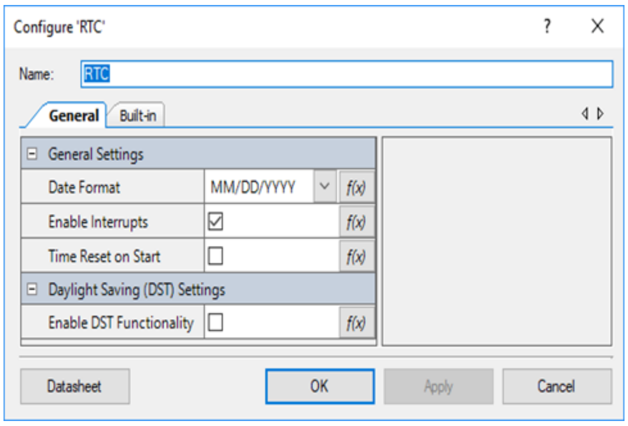
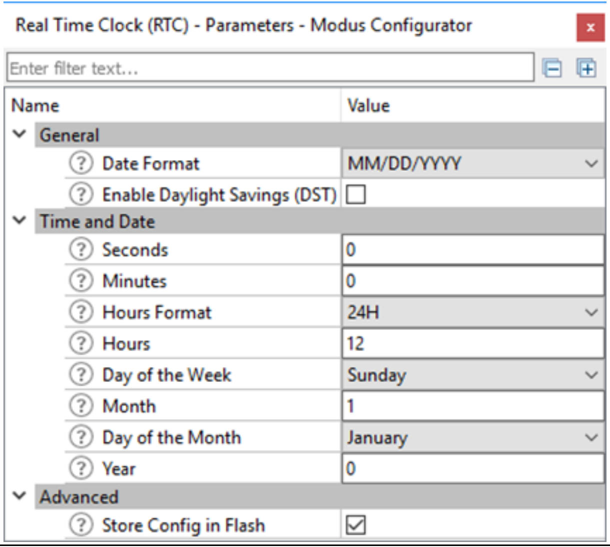
Multi-Counter Watchdog Timer (MCWDT) 0 - Parameters - Modus C...

Enter filter text...

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	☐
Cascade C1C2	☐
Advanced	
Store Config in Flash	☑

4.13 RTC

Replicate the settings as appropriate.

PSoC™ Creator	ModusToolbox™ Device Configurator
	

Configure 'RTC'

Name: RTC

General | Built-in

General Settings		
Date Format	MM/DD/YYYY	f(x)
Enable Interrupts	☑	f(x)
Time Reset on Start	☐	f(x)
Daylight Saving (DST) Settings		
Enable DST Functionality	☐	f(x)

Datasheet | OK | Apply | Cancel

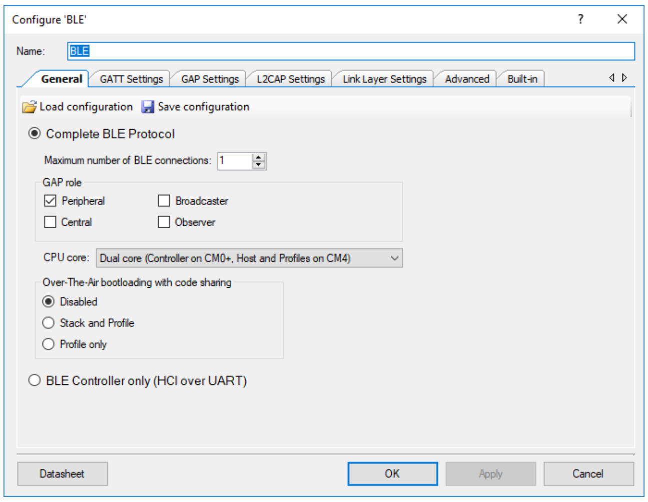
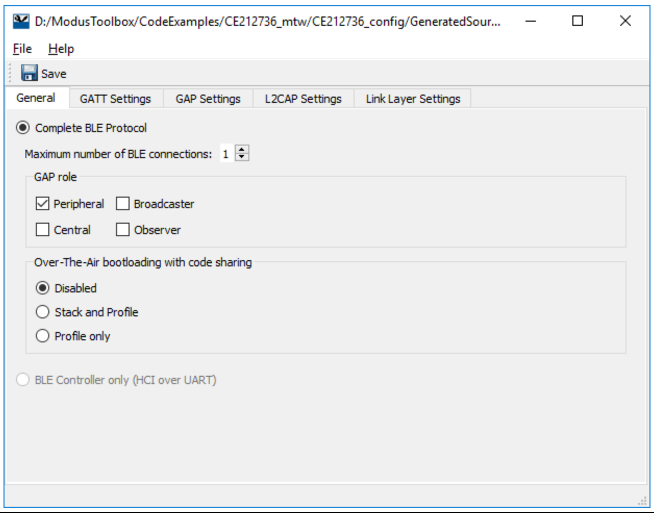
Real Time Clock (RTC) - Parameters - Modus Configurator

Enter filter text...

Name	Value
General	
Date Format	MM/DD/YYYY
Enable Daylight Savings (DST)	☐
Time and Date	
Seconds	0
Minutes	0
Hours Format	24H
Hours	12
Day of the Week	Sunday
Month	1
Day of the Month	January
Year	0
Advanced	
Store Config in Flash	☑

4.14 Bluetooth® LE

Replicate all settings from the PSoC™ Creator Component customizer to the ModusToolbox™ Bluetooth® Configurator. Refer also to the ModusToolbox™ Bluetooth® Configurator user guide, as needed.

PSoC™ Creator	ModusToolbox™ Bluetooth® Configurator
	

Configure 'BLE'

Name: BLE

General | GATT Settings | GAP Settings | L2CAP Settings | Link Layer Settings | Advanced | Built-in

Load configuration | Save configuration

● Complete BLE Protocol

Maximum number of BLE connections: 1

GAP role: ☑ Peripheral ☐ Broadcaster ☐ Central ☐ Observer

CPU core: Dual core (Controller on CM0+, Host and Profiles on CM4)

Over-The-Air bootloading with code sharing: ● Disabled ○ Stack and Profile ○ Profile only

○ BLE Controller only (HCI over UART)

Datasheet | OK | Apply | Cancel

D:/ModusToolbox/CodeExamples/CE212736_mtw/CE212736_config/GeneratedSour...

File | Help

Save

General | GATT Settings | GAP Settings | L2CAP Settings | Link Layer Settings

● Complete BLE Protocol

Maximum number of BLE connections: 1

GAP role: ☑ Peripheral ☐ Broadcaster ☐ Central ☐ Observer

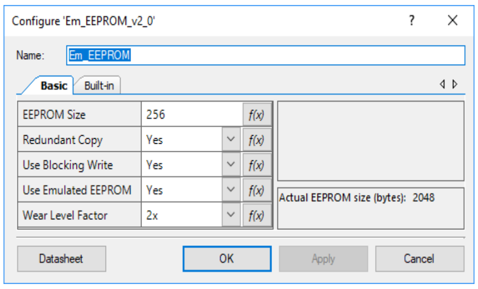
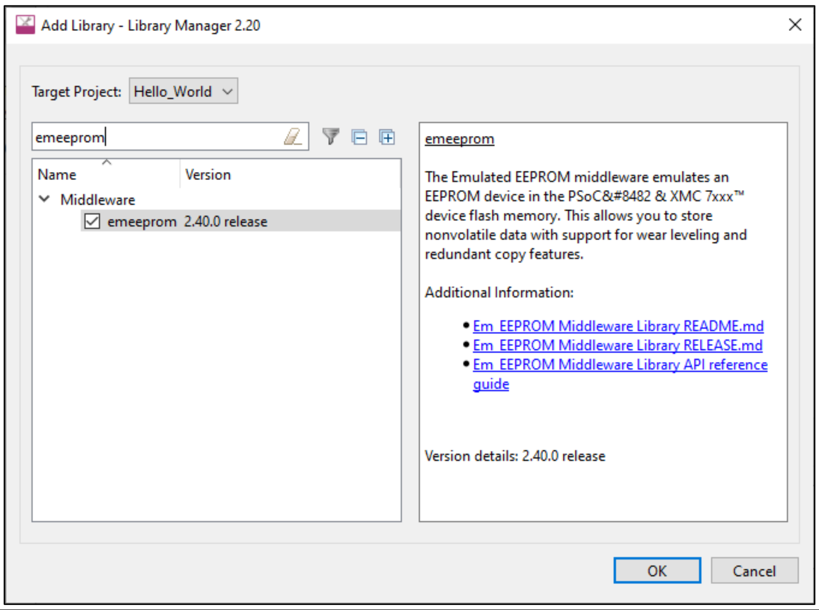
Over-The-Air bootloading with code sharing: ● Disabled ○ Stack and Profile ○ Profile only

○ BLE Controller only (HCI over UART)

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

4.15 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 user guide for more details.

PSoC™ Creator	ModusToolbox™ Library Manager
	

Configure 'Em_EEPROM_v2_0'

Name: Em_EEPROM

Basic | Built-in

EEPROM Size	256
Redundant Copy	Yes
Use Blocking Write	Yes
Use Emulated EEPROM	Yes
Wear Level Factor	2x

Actual EEPROM size (bytes): 2048

Add Library - Library Manager 2.20

Target Project: Hello_World

emeeprom

Name	Version
Middleware	
☑ emeeprom	2.40.0 release

emeeprom

The Emulated EEPROM middleware emulates an EEPROM device in the PSoC&#8482 & XMC 7xxx™ device flash memory. This allows you to store nonvolatile data with support for wear leveling and redundant copy features.

Additional Information:

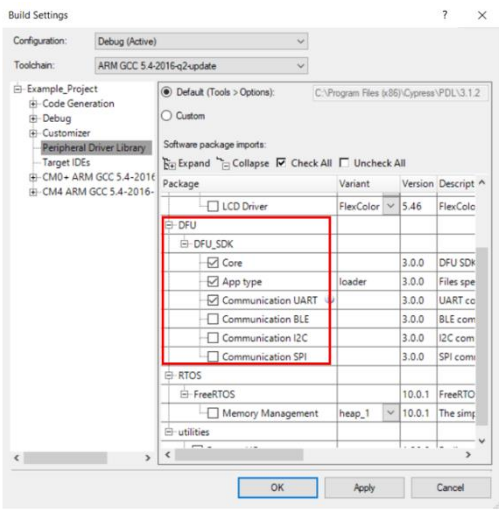
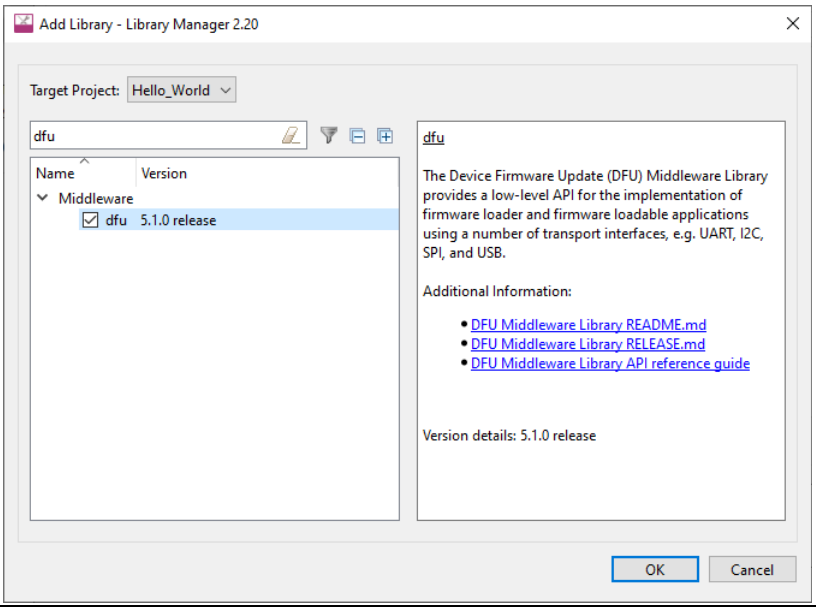
- Em_EEPROM Middleware Library README.md
- Em_EEPROM Middleware Library RELEASE.md
- Em_EEPROM Middleware Library API reference guide

Version details: 2.40.0 release

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 a way to alias it, so all the PSoC™ Creator API calls must be renamed to use the middleware API directly.

4.16 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 user guide for more details.

PSoC™ Creator	ModusToolbox™ Library Manager
	

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
- CM4 ARM GCC 5.4-2016-

(●) Default (Tools > Options): C:\Program Files (x86)\Cypress\PDL\3.1.2
( ) Custom

Software package imports:

Package	Variant	Version	Descript
☐ LCD Driver	FlexColor	5.46	FlexColo
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			

Add Library - Library Manager 2.20

Target Project: Hello_World

dfu

Name	Version
Middleware	
☑ dfu	5.1.0 release

dfu

The Device Firmware Update (DFU) Middleware Library provides a low-level API for the implementation of firmware loader and firmware loadable applications using a number of transport interfaces, e.g. UART, I2C, SPI, and USB.

Additional Information:

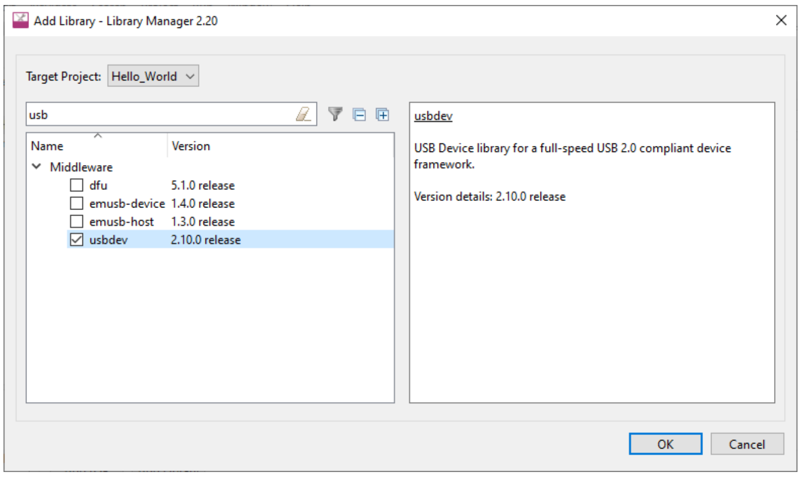
- DFU Middleware Library README.md
- DFU Middleware Library RELEASE.md
- DFU Middleware Library API reference guide

Version details: 5.1.0 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.

4.17 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 user guide for more details.



Add Library - Library Manager 2.20

Target Project: Hello_World

usb

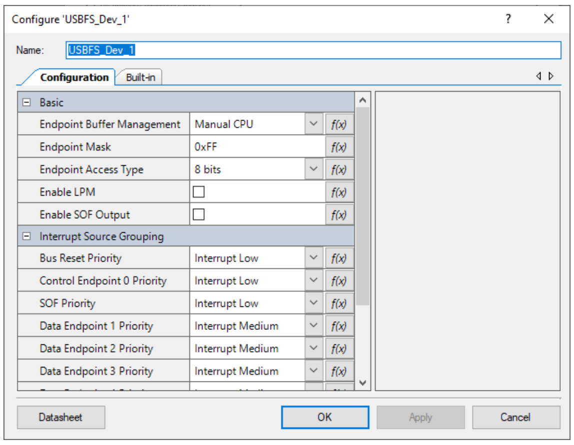
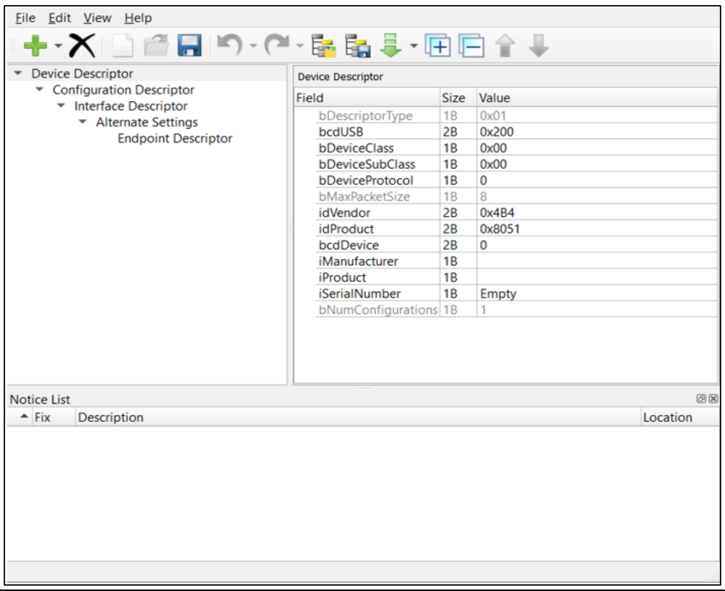
Name	Version
Middleware	
☐ dfu	5.1.0 release
☐ emusb-device	1.4.0 release
☐ emusb-host	1.3.0 release
☑ usbdev	2.10.0 release

usbdev

USB Device library for a full-speed USB 2.0 compliant device framework.

Version details: 2.10.0 release

The PSoC™ Creator USB Configurator settings can be replicated to the ModusToolbox™ USB Configurator. For USB Configurator details, refer to the ModusToolbox™ USB Configurator user guide.

PSoC™ Creator	ModusToolbox™ USB Configurator
	

Configure 'USBFS_Dev_1'

Name: USBFS_Dev_1

Configuration | Built-in

Basic	
Endpoint Buffer Management	Manual CPU
Endpoint Mask	0xFF
Endpoint Access Type	8 bits
Enable LPM	☐
Enable SOF Output	☐
Interrupt Source Grouping	
Bus Reset Priority	Interrupt Low
Control Endpoint 0 Priority	Interrupt Low
SOF Priority	Interrupt Low
Data Endpoint 1 Priority	Interrupt Medium
Data Endpoint 2 Priority	Interrupt Medium
Data Endpoint 3 Priority	Interrupt Medium

- Device Descriptor
  - Configuration Descriptor
    - Interface Descriptor
      - Alternate Settings
        - Endpoint Descriptor

Device Descriptor

Field	Size	Value
bDescriptorType	1B	0x01
bcdUSB	2B	0x200
bDeviceClass	1B	0x00
bDeviceSubClass	1B	0x00
bDeviceProtocol	1B	0
bMaxPacketSize	1B	8
idVendor	2B	0x4B4
idProduct	2B	0x8051
bcdDevice	2B	0
iManufacturer	1B	
iProduct	1B	
iSerialNumber	1B	Empty
bNumConfigurations	1B	1

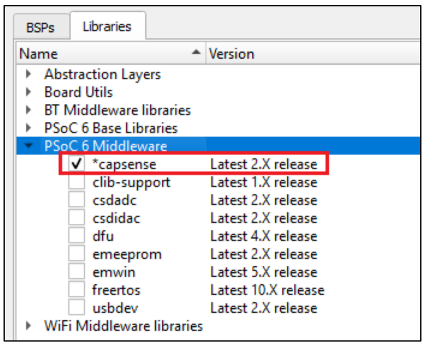
Notice List

Fix	Description	Location

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.

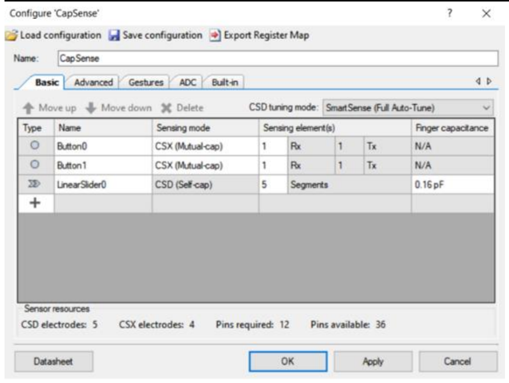
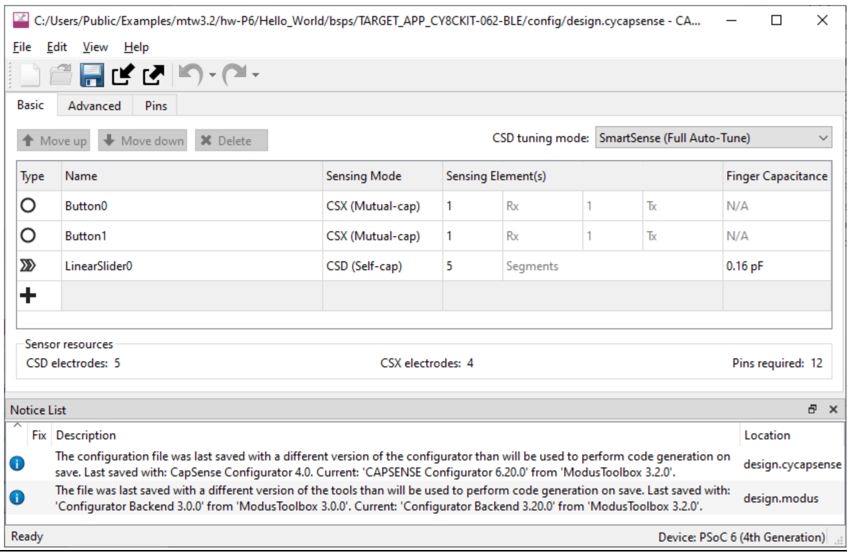
4.18 CAPSENSE™

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.



Name	Version
Abstraction Layers	
Board Utils	
BT Middleware libraries	
PSoC 6 Base Libraries	
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 5.X release
freertos	Latest 10.X release
usbdev	Latest 2.X release
WiFi Middleware libraries	

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.

PSoC™ Creator	ModusToolbox™ CAPSENSE™ Configurator
	

Configure 'CapSense'

Name: CapSense

CSD tuning mode: SmartSense (Full Auto-Tune)

Type	Name	Sensing mode	Sensing element(s)				Finger capacitance
○	Button0	CSX (Mutual-cap)	1	Rx	1	Tx	N/A
○	Button1	CSX (Mutual-cap)	1	Rx	1	Tx	N/A
⪢	LinearSlider0	CSD (Self-cap)	5	Segments			0.16 pF

Sensor resources
CSD electrodes: 5 CSX electrodes: 4 Pins required: 12 Pins available: 36

C:/Users/Public/Examples/mtw3.2/hw-P6/Hello_World/bsps/TARGET_APP_CY8CKIT-062-BLE/config/design.cycapsense - CA...

CSD tuning mode: SmartSense (Full Auto-Tune)

Type	Name	Sensing Mode	Sensing Element(s)				Finger Capacitance
○	Button0	CSX (Mutual-cap)	1	Rx	1	Tx	N/A
○	Button1	CSX (Mutual-cap)	1	Rx	1	Tx	N/A
⪢	LinearSlider0	CSD (Self-cap)	5	Segments			0.16 pF

Sensor resources
CSD electrodes: 5 CSX electrodes: 4 Pins required: 12

Notice List

Fix	Description	Location
	The configuration file was last saved with a different version of the configurator than will be used to perform code generation on save. Last saved with: CapSense Configurator 4.0. Current: 'CAPSENSE Configurator 6.20.0' from 'ModusToolbox 3.2.0'.	design.cycapsense
	The file was last saved with a different version of the tools than will be used to perform code generation on save. Last saved with: 'Configurator Backend 3.0.0' from 'ModusToolbox 3.0.0'. Current: 'Configurator Backend 3.20.0' from 'ModusToolbox 3.2.0'.	design.modus

Ready   Device: PSoC 6 (4th Generation)

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 user guide.

5 Copying Code

5.1 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.)

5.2 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"
```

5.3 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();
```

Revision history

Document revision	Date	Description of changes
**	1/30/19	New document.
*A	4/1/19	Updated for ModusToolbox™ 1.1
*B	6/10/20	Updated for ModusToolbox™ 2.1
*C	9/18/24	Updated for ModusToolbox™ 3.x

Edition 2024-09-18

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email:

erratum@infineon.com

Document reference

002-25408 Rev. *C