Getting started with Programmable Power Control Accelerator (PPCA) and peripheral configurations

About this document

Scope and purpose

This application note introduces the Programmable Power Control Accelerator (PPCA) available in the PSOC™ Control C3M/P8 (PSC3M/P8), PSOC™ Control C3M/P7 (PSC3M/P7) MCUs, which is a powerful peripheral with two Arm® Cortex®-M33 CPU cores and feature-rich analog and digital peripherals designed for complex power conversion and motor control applications. This application note helps you to explore the PPCA peripheral and software support available for using it for implementing a few power conversion solutions.

This document covers the following subfamilies of PSC3M/P8, PSC3M/P7 MCUs:

- PSOC™ Control C3 P (Power Control) series: PSC3P7xx, PSC3P8xx
- PSOC™ Control C3 M (Motor Control) series: PSC3M7xx, PSC3M8xx

For an overview of the PSOC™ Control C3M/P8, C3M/P7 MCU product family, see the datasheet.

***Note**: The configurations in this document are based on the ModusToolbox™ Device Configurator and Peripheral Driver Library (PDL).*

Intended audience

This document is intended for users who are new to PSC3M/P8, PSC3M/P7 MCUs.

Associated part family

All PSC3M/P8, PSC3M/P7 devices.

Table of contents

About this document . . . 1
Table of contents . . . 2
1 Introduction . . . 4
1.1 Introduction to PSC3M/P8, PSC3M/P7 MCUs . . . 4
1.1.1 PSC3M/P8, PSC3M/P7 MCUs architecture . . . 5
1.1.2 Product variants . . . 6
1.2 Introduction to PSC3M/P8, PSC3M/P7 PPCA . . . 6
1.2.1 PSC3M/P8, PSC3M/P7 PPCA architecture . . . 7
2 Getting started with PSC3M/P8, PSC3M/P7 PPCA . . . 9
2.1 Development kits (list of development kits available) . . . 9
2.2 Software support . . . 9
2.2.1 ModusToolbox™ . . . 9
2.2.2 Supported IDEs . . . 9
2.2.3 Code examples and documentation . . . 10
2.2.4 Configurators and runtime APIs for PPCA resources . . . 12
2.2.4.1 PPCA CPUs . . . 12
2.2.4.2 PPCA Configuration (PPCA CNFG) . . . 13
2.2.4.2.1 PPCA general configuration . . . 14
2.2.4.2.2 PPCA configuration . . . 15
2.2.4.2.3 PPCA input configuration . . . 16
2.2.4.2.4 PPCA input trigger configuration . . . 17
2.2.4.2.5 PPCA output configuration . . . 18
2.2.4.3 PPCA ATOP . . . 20
2.2.4.3.1 ATOP analog reference (AREF) . . . 21
2.2.4.3.2 ATOP ADC . . . 22
2.2.4.3.3 ATOP ADC filter . . . 24
2.2.4.3.4 ATOP R2R DAC . . . 26
2.2.4.3.5 ATOP DCMP . . . 28
2.2.4.3.6 ATOP DCSG slope generator . . . 30
2.2.4.4 PPCA TCPWM . . . 33
2.2.4.5 PPCA custom logic block (CLB) . . . 35
2.2.4.6 PPCA event processing unit (EPU) . . . 38
2.2.4.6.1 EPU combiner . . . 39
2.2.4.6.2 EPU combiner feedback . . . 40
2.2.4.6.3 EPU Processing Unit 1 (PU1) . . . 41
2.2.4.6.4 EPU Processing Unit 2 (PU2) . . . 42
2.2.4.7 Hardware 3-pole, 3-zero (3P3Z) filter . . . 44
2.2.4.8 PPCA CORDIC . . . 46

Table of contents

3 Solutions using PPCA 47
3.1 Average Current Mode Control (ACMC) strategy using PPCA 47
3.1.1 Introduction to ACMC strategy 47
3.1.2 PPCA resources for ACMC strategy 48
3.1.3 Hardware architecture of the ACMC using PPCA 48
3.1.4 Firmware co-design 49
3.1.5 Benefits compared to legacy implementation 50
3.2 Interleaved totem-pole PFC implementation using PPCA 50
3.2.1 Introduction to interleaved totem-pole PFC 51
3.2.2 PPCA resources for interleaved totem-pole PFC 51
3.2.3 Hardware architecture of the interleaved totem-pole PFC using PPCA 52
3.2.3.1 Voltage loop dedicated to CPU from PPCA 0 52
3.2.3.2 Current loop dedicated to CPU from PPCA 1 52
3.2.3.3 Fast protections 52
3.2.4 Firmware design 53
3.2.5 Benefits compared to legacy implementation 54
3.3 Half-bridge LLC implementation using PPCA 55
3.3.1 Introduction to half-bridge LLC 55
3.3.2 PPCA resources for half-bridge LLC 56
3.3.3 Hardware architecture of the half-bridge LLC using PPCA 56
3.3.3.1 Voltage loop dedicated to CPU from PPCA 56
3.3.3.2 Dead-time and synchronous rectification control dedicated to CPU from PPCA 1 57
3.3.3.3 Fast protections 57
3.3.4 Firmware design 57
3.3.5 Benefits compared to legacy implementation 58
4 Summary 59
References 60
Revision history 61
Trademarks 62
Disclaimer 63

1 Introduction

This section helps you to understand the features and architecture of PSC3M/P8, PSC3M/P7 MCUs and the PPCA subsystem.

1.1 Introduction to PSC3M/P8, PSC3M/P7 MCUs

PSC3M/P8, PSC3M/P7 MCUs are mainly targeted for implementation of complex power conversion solutions, such as totem-pole power factor correction (PFC), half-bridge LLC, full-bridge LLC, phase-shift full-bridge as well as their interleaved versions, and many other topologies that use wide bandgap (WBG) devices and motor control solutions involving multiple motors in various configurations.

The PSOC™ Control C3 MCU integrates the following features on a single chip:

- 180 MHz 32-bit Arm® Cortex®-M33 CPU with
  - Floating-point unit (FPU)
  - Digital signal processor (DSP)
  - Memory protection unit (MPU)
  - TrustZone® framework support
  - 16 KB instruction cache (I-cache)
- Programmable power control accelerator (PPCA) subsystem with
  - Two 200 MHz 32-bit Arm® Cortex®-M33 CPUs
  - 112 KB SRAM with SRAM
  - Four 12-bit, 25 Msps SAR ADC modules, 24 direct channels plus 16 multiplexed channels
  - Nine differential comparator slope generators and fast analog comparators with <10 ns built-in 12-bit DAC supporting up to 50 MHz refresh rate
  - Two 11-bit buffered R2R DAC supporting up to 2 MHz refresh rate
  - Two coordinate rotation digital computer (CORDIC) blocks
  - 12 x 32-bit high-resolution PWMs (HRPWM), <100 ps, with complementary outputs
  - 14 x 16-bit PWM running at CPU clock resolution with complementary output
  - Configurable and programmable events processing unit (EPU) block for handling signals
  - Two independent instances of 3P3Z hardware filters total with six channels
  - Custom logic block unit (CLB) implementing digital current observer and matrix converter
- Up to 512 KB Read-While-Write flash with ECC support
- 64 KB ROM for boot code
- 128 KB SRAM with ECC support
- 2 x 32-bit and 2 x 16-bit pulse-width modulator (TCPWM)
- Seven serial communication blocks (SCB) supporting UART, I2C, and SPI protocols, two with extended PMBus® features and two CAN FD (one supporting 8 Mbps)
- Low-power operation: Sleep, Deep Sleep (three modes) below 12 µA, and Hibernate below 1 µA
- Up to 60 GPIOs with programmable drive modes, strengths, and slew rates, and 26 analog pins
- Security: Platform Security Architecture (PSA) L3 certified; configurable flash partitioning and protection
- Class B and SIL 2 compliant safety test libraries are available
- ModusToolbox™ development environment with installable SDKs and libraries, industry-standard Arm® tools, and RTOS support

1.1.1 PSC3M/P8, PSC3M/P7 MCUs architecture

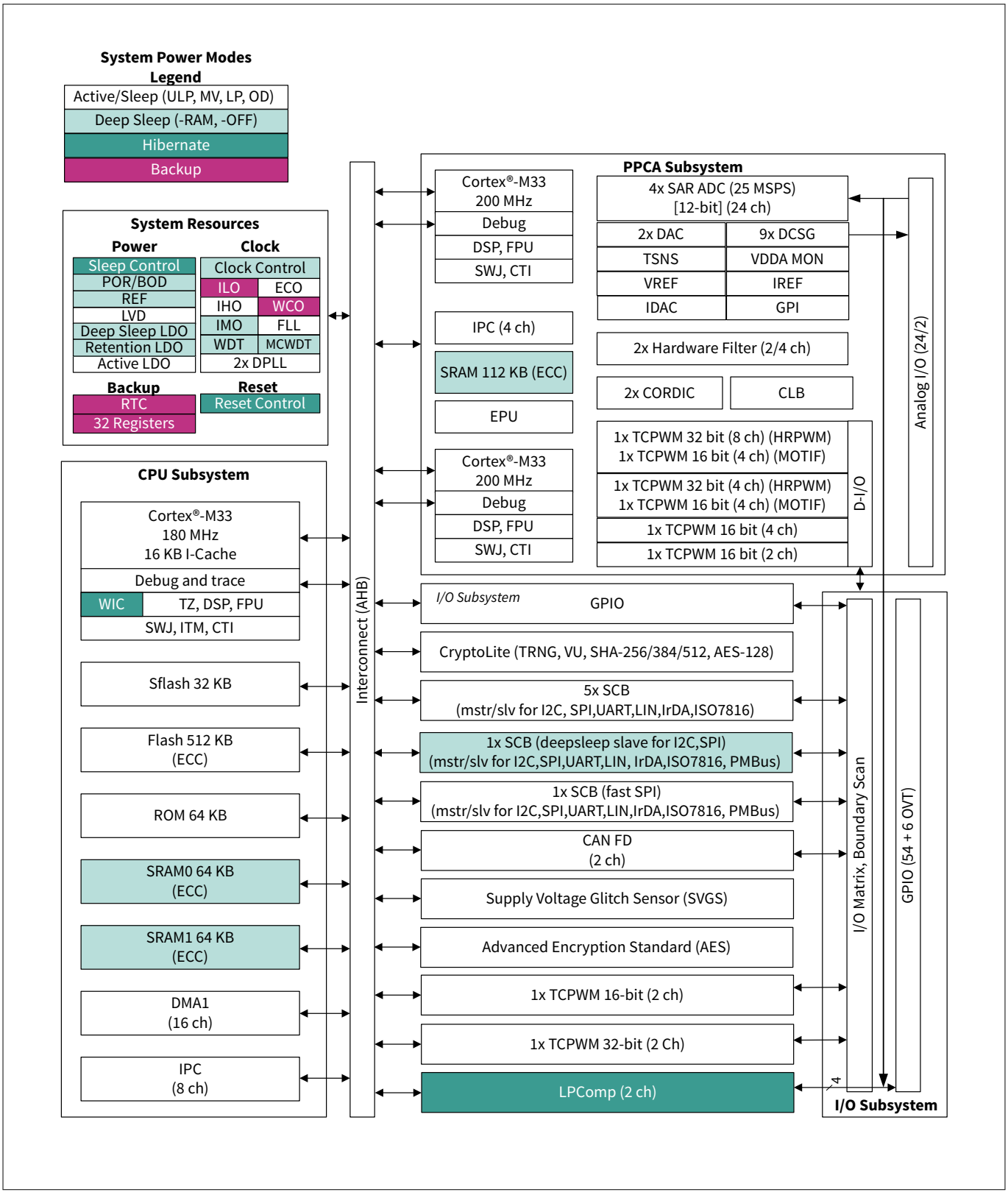


Figure 1 PSC3M/P8, PSC3M/P7 MCUs architecture

1.1.2 Product variants

There are four variants in the PSC3M/P8, PSC3M/P7 MCUs. Table 1 provides an overview of different components available in the product line.

Table 1 Product variants

PSC3P8x	PSC3M8x	PSC3P7x	PSC3M7x
• 180 MHz Arm® Cortex®-M33 main core, up to 512 KB flash and 256 KB RAM • Dual core 200 MHz PPCA Arm® Cortex®-M33, 25 Msps ADC, HRPWM	• 180 MHz Arm® Cortex®-M33 main core, up to 512 KB flash and 256 KB RAM • Dual core 200 MHz PPCA Arm® Cortex®-M33, 25 Msps ADC, HRPWM, CORDIC, MOTIF	• 180 MHz Arm® Cortex®-M33 main core, up to 512 KB flash and 256 KB RAM • Single core 200 MHz PPCA Arm® Cortex®-M33, 25 Msps ADC, HRPWM	• 180 MHz Arm® Cortex®-M33 main core, up to 512 KB flash and 256 KB RAM • Single core 200 MHz PPCA Arm® Cortex®-M33, 25 Msps ADC, HRPWM, CORDIC, MOTIF

As shown in Introduction to PSC3M/P8, PSC3M/P7 MCUs, the MCU contains PPCA and other peripherals. The PPCA contains peripherals that help implementing power conversion solutions easily and efficiently. As shown in Table 1, there are differences in the resources available in PPCA subsystem on the MCU variants. The MCUs with name containing M has PPCA with MOTIF and CORDIC instances in it. MOTIF is a feature in TCPWM for sensored motor control without any CPU intervention. CORDIC helps in solving circular (trigonometric) and hyperbolic functions. PPCA has two instances of it.

There are differences in the PPCA subsystem between the MCU with name "7x" and "8x". The ones with name ending with "7x" has only one Cortex®-M33 CPU in the PPCA subsystem, whereas the one with name ending with "8x" has two CPUs.

1.2 Introduction to PSC3M/P8, PSC3M/P7 PPCA

Programmable Power Control Accelerator (PPCA) is a programmable subsystem mainly targeted towards the implementation of AC-DC, DC-AC, DC-DC, switch mode power supply (SMPS) converters, like totem-pole PFC, a half-bridge LLC, a full-bridge LLC, phase-shift full-bridge, dual active bridge dual stage, CLLC, etc. by interfacing with wide bandgap (WBG) devices that switch at high frequencies.

PPCA subsystem has the following features:

- Two 200 MHz 32-bit Arm® Cortex®-M33 CPUs
- 112 KB SRAM
- Four 12-bit, 25-Msps, SAR ADC modules with 24 direct channels plus 16 multiplexed channels
- Nine differential comparator slope generators and fast analog comparators with <10 ns built-in 12-bit DAC supporting a refresh rate of maximum 50 MHz
- Two 11-bit buffered R2R DAC with refresh rate up to 2 MHz
- Two coordinate rotation digital computer (CORDIC) blocks
- 14 x 16-bit and 12 x 32-bit timer/counter pulse-width modulator (TCPWM)
- 12 x 32-bit high-resolution PWMs (HRPWM), <100ps, with complementary outputs
- 14 x 16-bit PWM CPU clock resolution with complementary output
- Configurable and programmable events processing unit (EPU) block for handling signals
- Two independent instances of 3P3Z hardware filters total with six channels
- Custom logic block unit (CLB) implementing digital current observer and matrix converter

1.2.1 PSC3M/P8, PSC3M/P7 PPCA architecture

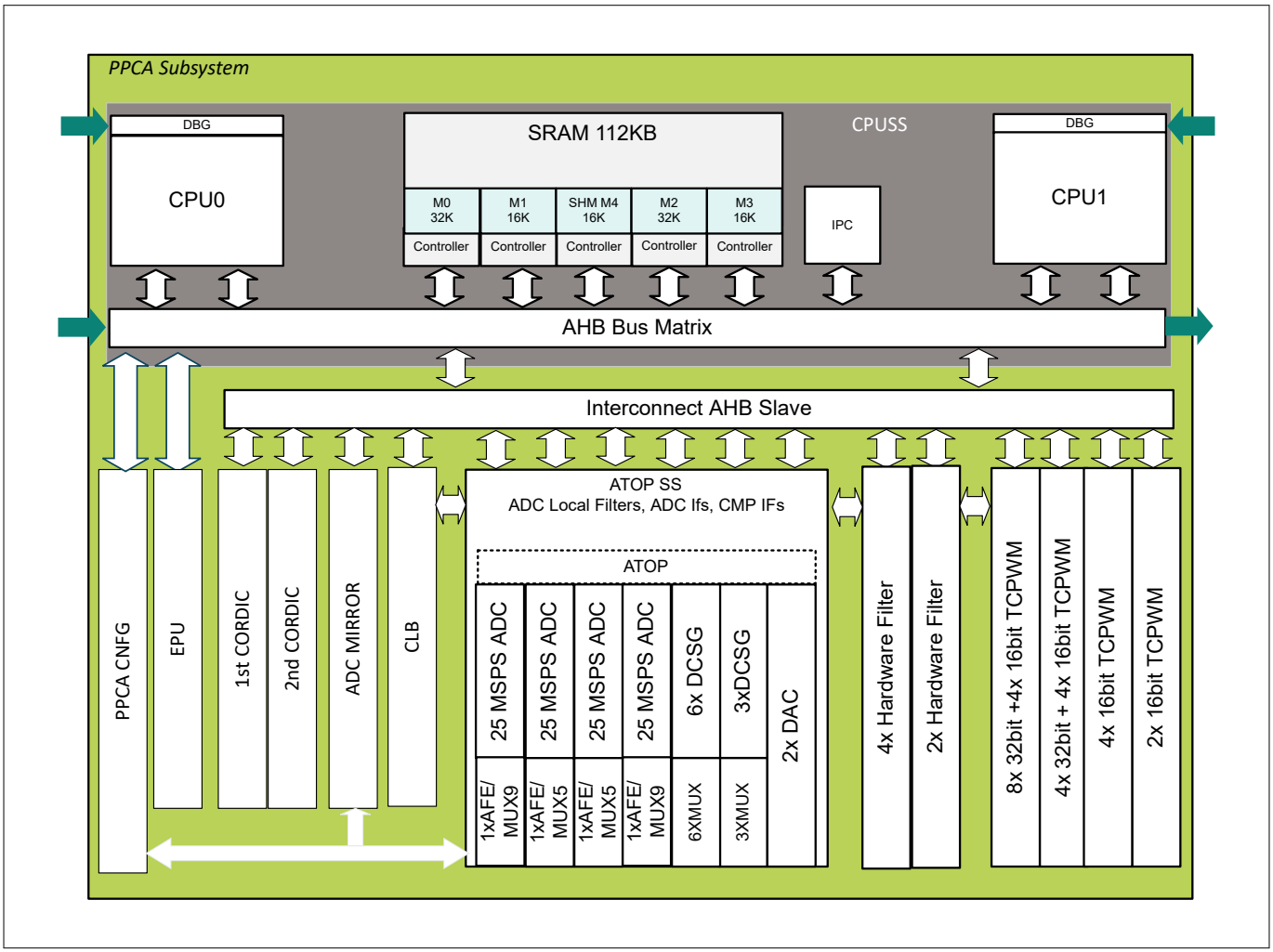


Figure 2 PSC3M/P8, PSC3M/P7 PPCA architecture

The PPCA block is partitioned into two main sections:

- PPCA CPU subsystem
- PPCA peripherals subsystem

The CPU subsystem is based on the AMBA® bus infrastructure. It embeds two dedicated Cortex®-M33 processors with Harvard architecture, local memory and shared memory, and the bus infrastructure to manage the access to available embedded peripherals. The peripherals section embeds all the peripherals aimed to support the implementation of SMPS converters.

All peripheral events and triggers in the PPCA subsystem are interconnected through the EPU module, instantiating multiple processing units. Trigger event signals are used by PPCA peripherals to indicate the occurrence of an event or transition to a state and requiring a transition to a new state or triggering a new operation.

The PPCA analog top subsystem (ATOP SS) embeds the analog components in the PPCA block (ADCs, differential comparator slope generators, DACs, analog multiplexers, temperature sensor, voltage reference, current reference, etc.) and the digital logic to interface and manage the instantiated analog components.

Custom logic block (CLB) is another resource in PPCA that contains two components: digital current observer (DICO) and matrix converter (MACO). DICO is for multi-phase and multi-mode totem-pole PFC, while MACO is designated for use with a matrix converter control algorithm supporting soft-start for resonant topologies via trajectory control and a commutation matrix.

PPCA has two independent instances of hardware filter (HWFILT). Each filter provides a configuration hardware filter function, three poles/three zeros.

The timer, counter, and pulse-width modulator (TCPWM) block implements a 16- or 32-bit timer, counter, pulse width modulator (PWM), high-resolution PWM (HRPWM), and motion interface (MOTIF) functionality.

PSC3M/P8, PSC3M/P7 supports two CORDIC modules. CORDIC is a peripheral module to support the CPU with hardware calculations for trigonometric, hyperbolic, and complex numbers, and related functions.

See the architecture reference manual for detailed information.

2 Getting started with PSC3M/P8, PSC3M/P7 PPCA

2.1 Development kits (list of development kits available)

Infineon provides a wide variety of hardware development kits in various form factors and features to enable easy and rapid evaluation and prototyping of applications based on PSC3M/P8, PSC3M/P7 devices. ModusToolbox™ software is the software development platform for creating embedded applications using the development kits.

Table 2 List of development kits

Kit	Product	Kit name	BSP GitHub repo
KIT_PSC3M8_EVK	PSC3M8xD	PSOC™ Control C3M8 Evaluation Kit	BSP
KIT_PSC3M8_CC1	PSC3M8xD	PSOC™ Control C3M8 Digital Power Control Card	BSP

For the complete list of kits for the PSC3M/P8, PSC3M/P7 MCUs along with the shield modules, see the PSOC™ Control C3M/P8, C3M/P7 MCU webpage.

2.2 Software support

2.2.1 ModusToolbox™

Infineon provides the ModusToolbox™ software for firmware/application development on MCUs. ModusToolbox™ software is a modern, extensible development ecosystem supporting a wide range of Infineon microcontroller devices, including PSOC™ Arm® Cortex® Microcontrollers, TRAVEO™ T2G Arm® Cortex® Microcontroller, XMC™ Industrial Microcontrollers, AIROC™ Wi-Fi devices, AIROC™ Bluetooth® devices, and USB-C Power Delivery Microcontrollers.

This software includes configuration tools, low-level drivers, middleware libraries, and other packages that enable you to create MCU and wireless applications. All tools run on Windows, macOS, and Linux. ModusToolbox™ includes an Eclipse IDE, which provides an integrated flow with all the ModusToolbox™ tools. Other IDEs such as Visual Studio Code, IAR Embedded Workbench, and Arm® MDK (µVision) are also supported.

ModusToolbox™ software supports standalone device and middleware configurators. Use the configurators to set the configuration of different blocks in the device and generate code that can be used in firmware development. Libraries and enablement software are available at the GitHub site.

ModusToolbox™ tools and resources can also be used on the command line. See the "Build system" chapter in the ModusToolbox™ tools package user guide for detailed documentation.

2.2.2 Supported IDEs

ModusToolbox™ software, the latest-generation toolset, is supported across Windows, Linux, and macOS platforms. ModusToolbox™ software supports 3rd-party IDEs, including the Eclipse IDE, Visual Studio Code, Arm® MDK (µVision), and IAR Embedded Workbench. The tools package includes an implementation for all the supported IDEs. The associated BSP and library configurators also work on all three host operating systems.

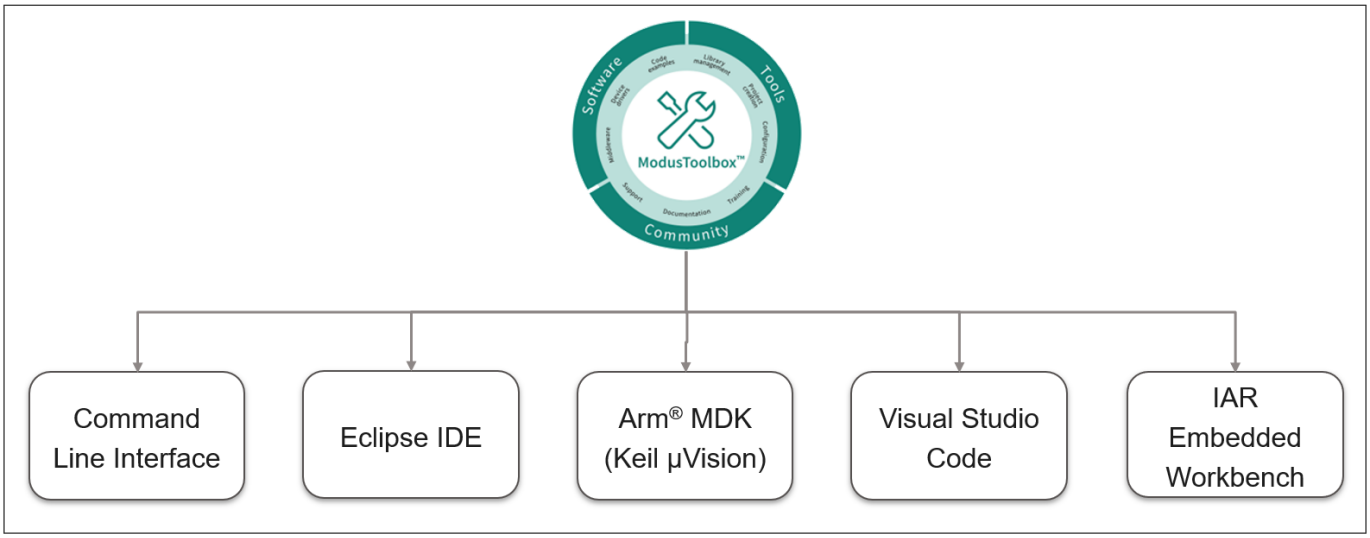


Figure 3 ModusToolbox™supported IDEs

Refer Visual Studio Code for ModusToolbox™ user guide for working with VS Code.

Refer IAR Embedded Workbench for ModusToolbox™ user guide for working with IAR.

Refer Keil µVision for ModusToolbox™ user guide for working with Keil uVision.

2.2.3 Code examples and documentation

A large set of code examples is available with ModusToolbox™ to introduce Infineon MCUs and its peripherals, including PPCA. As the first step, install ModusToolbox™. Refer to the ModusToolbox™ tools package installation guide for details.

After installing ModusToolbox™, open the Project Creator tool and select **KIT_PSC3M8_EVK** as the target and click **Next**. You will be presented with the list of code examples available for the PSC3M/P8, PSC3M/P7 MCUs, as shown in Figure 4.

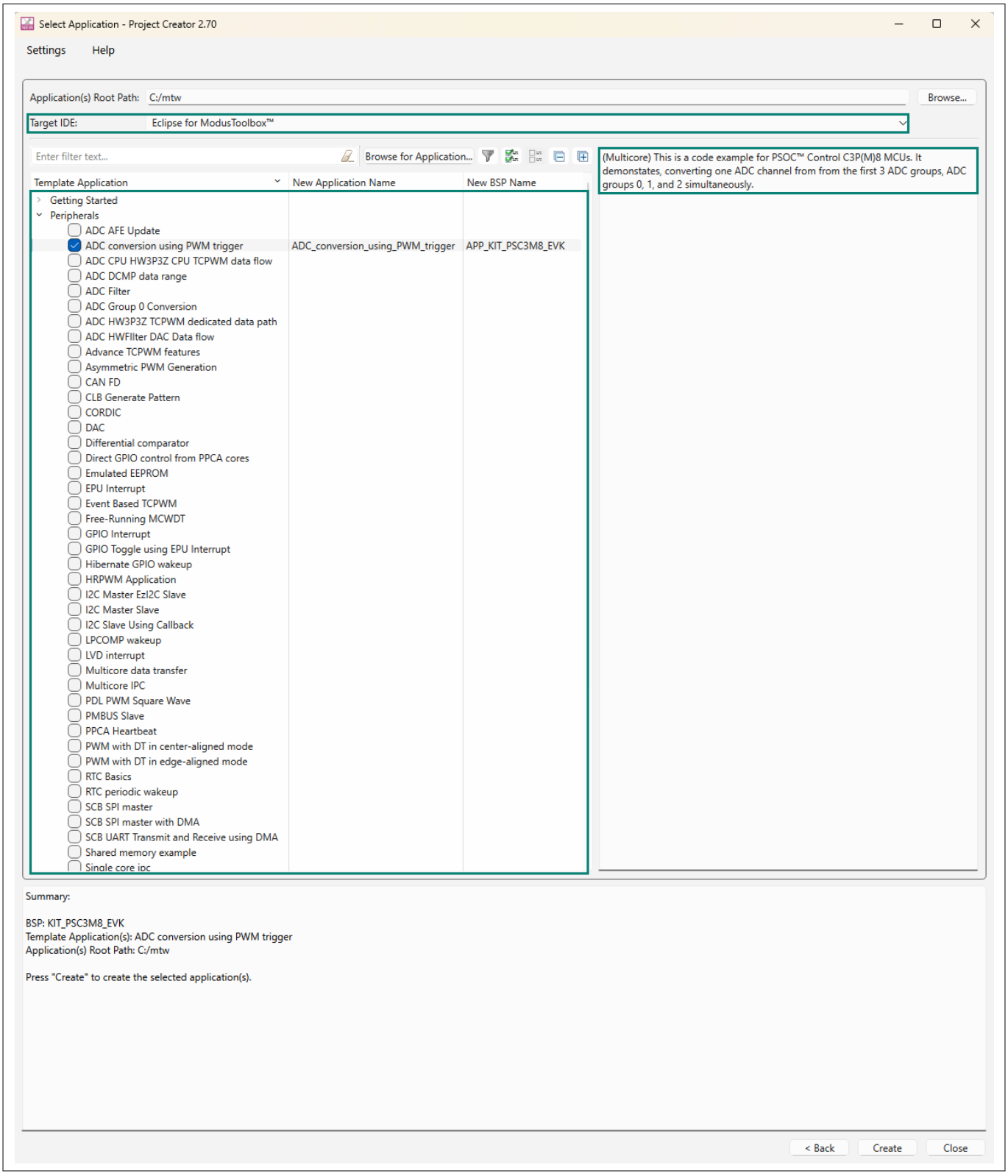


Figure 4 ModusToolbox™- Project Creator

After creating the required code example, open it in the IDE it is created for and go through the implementation. ModusToolbox™ Device Configurator is the tool available for configuring the peripherals.

In the project folder (`<Project name>\bsps\TARGET_APP_KIT_PSC3M8_EVK\config\`), a file called `design.modus` is available. Open this file with Device Configurator to see the configuration.

For details of the PPCA peripheral and PSC3M/P8, PSC3M/P7, see the PSOC™ Control C3M/P8, C3M/P7 architecture reference manual, and PSOC™ Control C3M/P8, C3M/P7 registers reference manual. Details of the driver library and other software are available with the software package inside the `mtb_shared` folder created when a new project is created.

2.2.4 Configurators and runtime APIs for PPCA resources

This section lists the features of the PPCA block, introduces the configurator support for it, and shows you where driver resources can be found to use the resource.

In Device Configurator, PPCA subblocks are grouped together in the "Peripherals" section as shown in Figure 5.

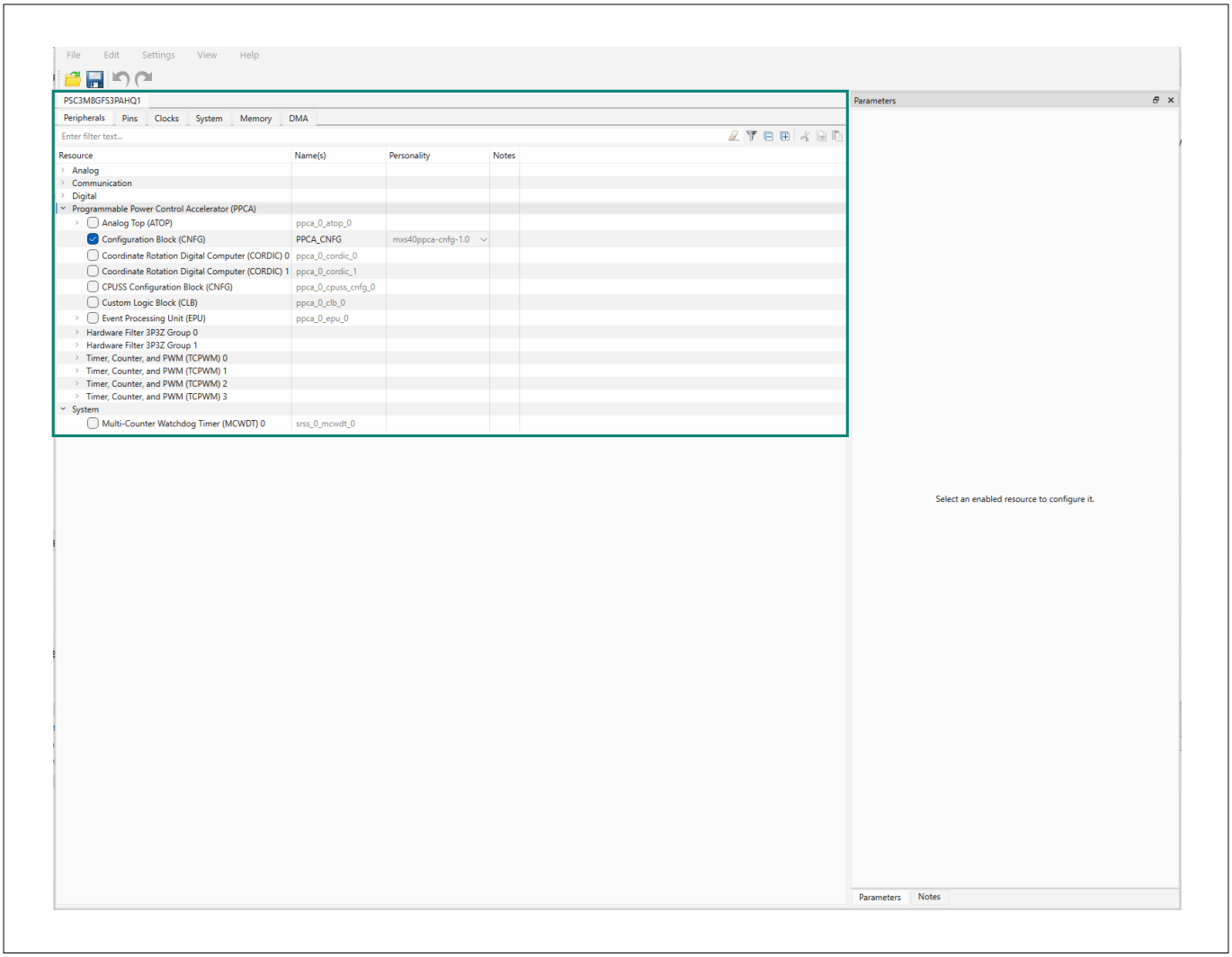


Figure 5 ModusToolbox™- Device Configurator

Use the Peripheral Driver Library (PDL) for initialization and runtime accessing of the PPCA resources. The PDL is located under the following path in the Project Explorer after a ModusToolbox™ project is created: `\mtb_shared\mtb-dsl-psc3m8\release-vXXXX\pdl`.

2.2.4.1 PPCA CPUs

PPCA contains up to two Arm® Cortex®-M33 CPUs running at up to 200 MHz, local and shared memories of the CPUs, an interprocessor communication module, a bus matrix and a REMAP function. These CPUs contain a fast integrated multiplier, a fast integrated hardware divider, a DSP unit, and an FPU.

Having two CPUs helps in implementing multiple control loops that are required for solutions like PFC and motor control solutions.

PDL support is available to configure and manage the execution of the PPCA CPU and are available in the files `\drivers\include\cy_system_ppca_init.h` and `\drivers\source\cy_system_ppca_init.c`.

The following code snippet shows the initialization code for the PPCA CPU. Include it in your program to use the PPCA CPU functionality.

```
/* These are the addresses where the core0 and core1 images are located. */
#define CORE0_IMAGE_ADDRESS    CYMEM_CM33_0_S_m33s_ppca0_nvm_C_S_START // 0x12070000u
#define CORE1_IMAGE_ADDRESS    CYMEM_CM33_0_S_m33s_ppca1_nvm_C_S_START // 0x12078000u

/* Load and boot PPCA Core 0 (ppca_cm33_0) and Core 1 (ppca_cm33_1) from flash.*/
/* Once started, both cores run concurrently and independently of this core.   */
    Cy_System_Init_CPU0((void*)CORE0_IMAGE_ADDRESS, IMAGE_SIZE);
    Cy_System_Init_CPU1((void*)CORE1_IMAGE_ADDRESS, IMAGE_SIZE);
```

In the definition of the above APIs, the following APIs mentioned are called.

```
/* Enable PPCA RAM access before copying application image */
Cy_System_PPCA_RAM_Enable();

/* Copy first 32KB to PPCA M0 memory region */
Cy_System_Copy_PPCA_Image_To_Memory((void*)image_addr_m0, IMAGE_SIZE_32K, CY_PPCA_MEMORY_0);

/* Boot PPCA Core0 to start execution */
Cy_SysEnable_PPCA_Core0();
```

2.2.4.2 PPCA Configuration (PPCA CNFG)

PPCA Configuration (PPCA CNFG) is the resource available in the PPCA for controlling the peripheral behavior, resource allocation, accessing of mirrored data from the ADC, selecting of EPU inputs, and selecting of outputs to the PPCA boundary.

Peripherals must be assigned to one of the two CPUs using the "Configuration" section. Assigning them to a specific CPU enables zero wait-state access to the peripherals. The option in the Device Configurator for using the PPCA configuration is shown in Figure 6.

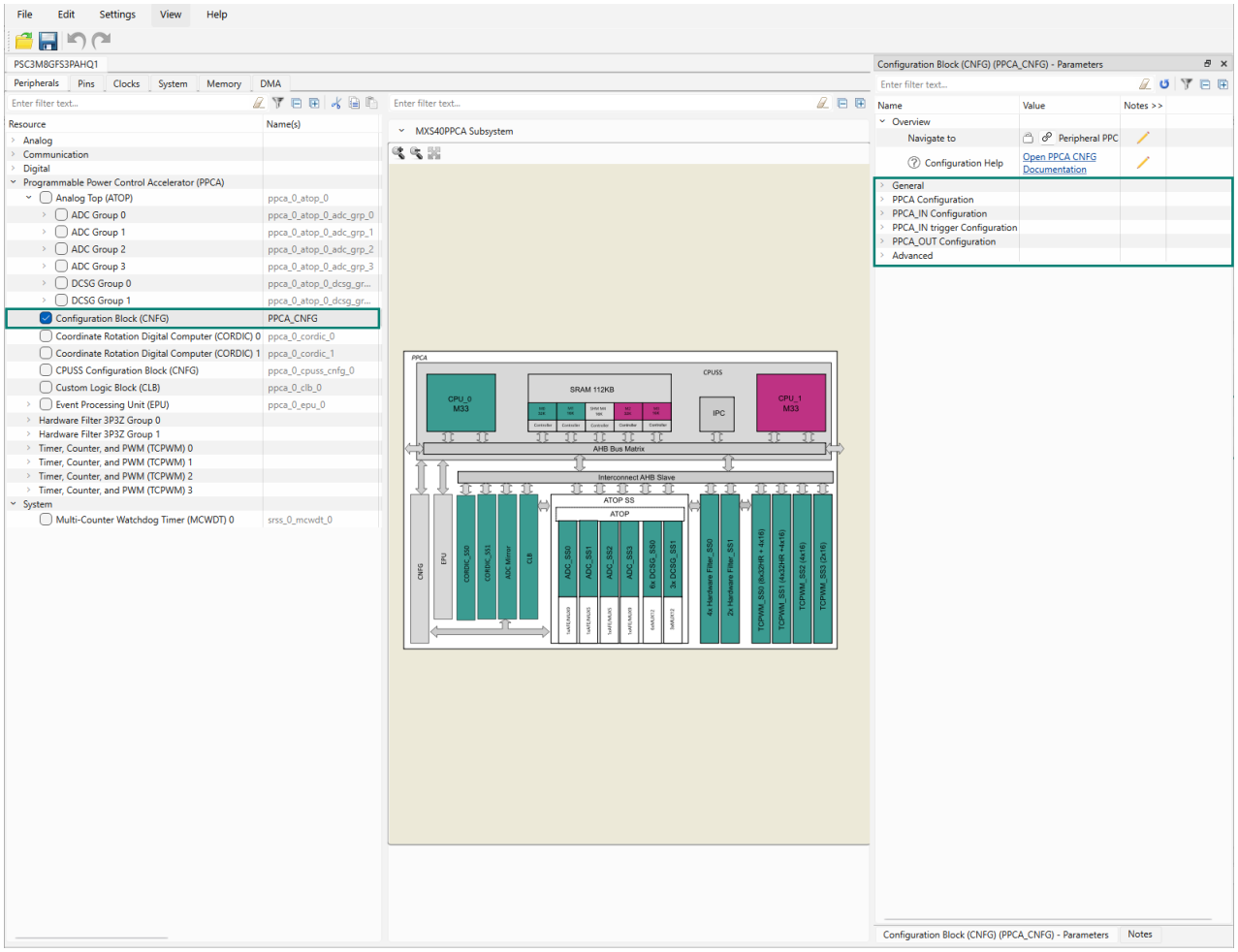


Figure 6 ModusToolbox™ - Device Configurator - PPCA Configuration block

2.2.4.2.1 PPCA general configuration

Enable or disable:

- PPCA Deep Sleep retention
- PPCA bus matrix remap (which, when enabled, sets reduced data memory, and requires CPU1 to be disabled)
- CPU0/CPU1 HRESP error propagation as a fault control
- CPU1 clocks

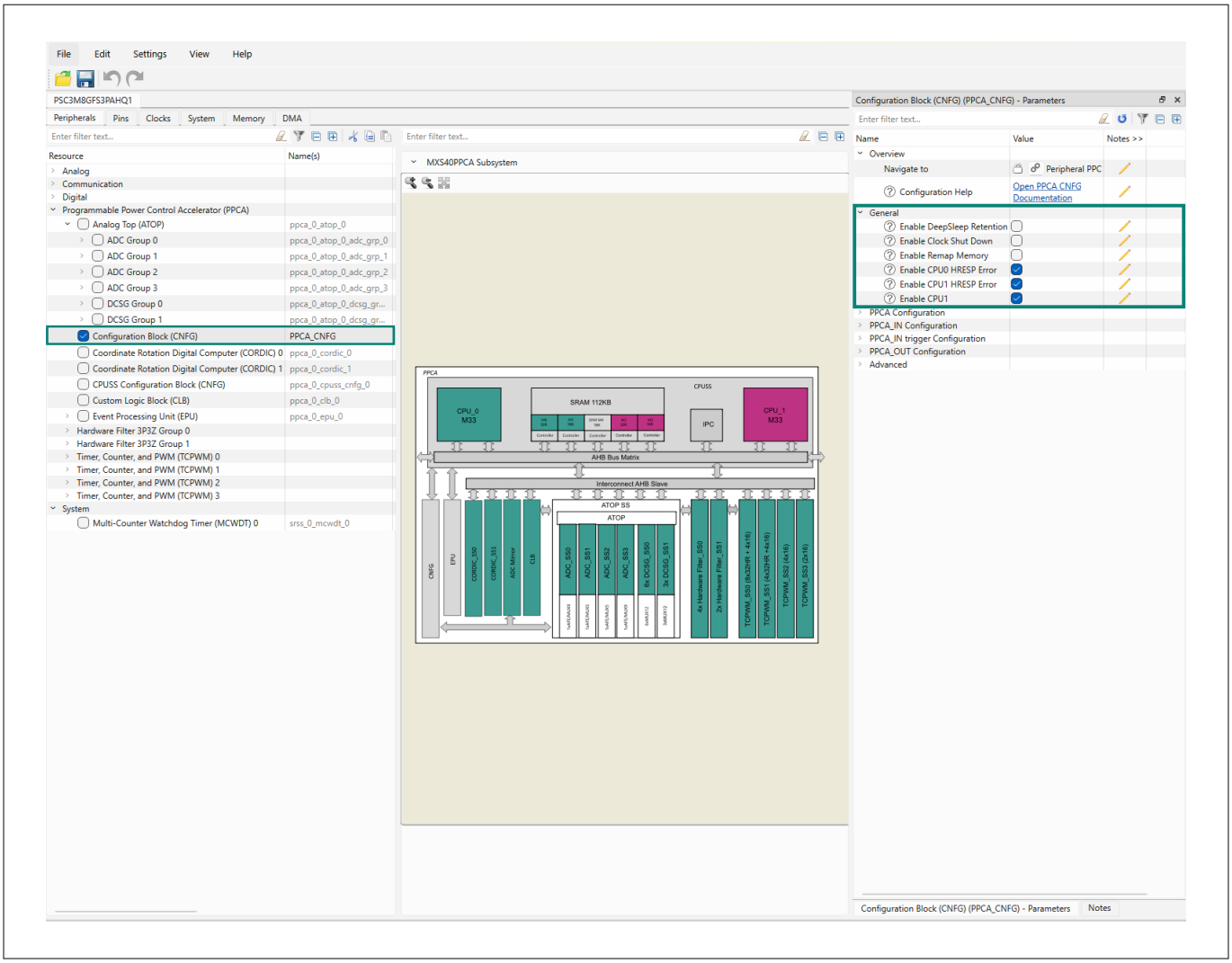


Figure 7 PPCA general configuration

2.2.4.2.2 PPCA configuration

Assign peripherals, such as TCPWM, HWFILTER, ADC, CORDIC, and DCSG to CPU0 or CPU1. Only the assigned CPU can control the peripheral.

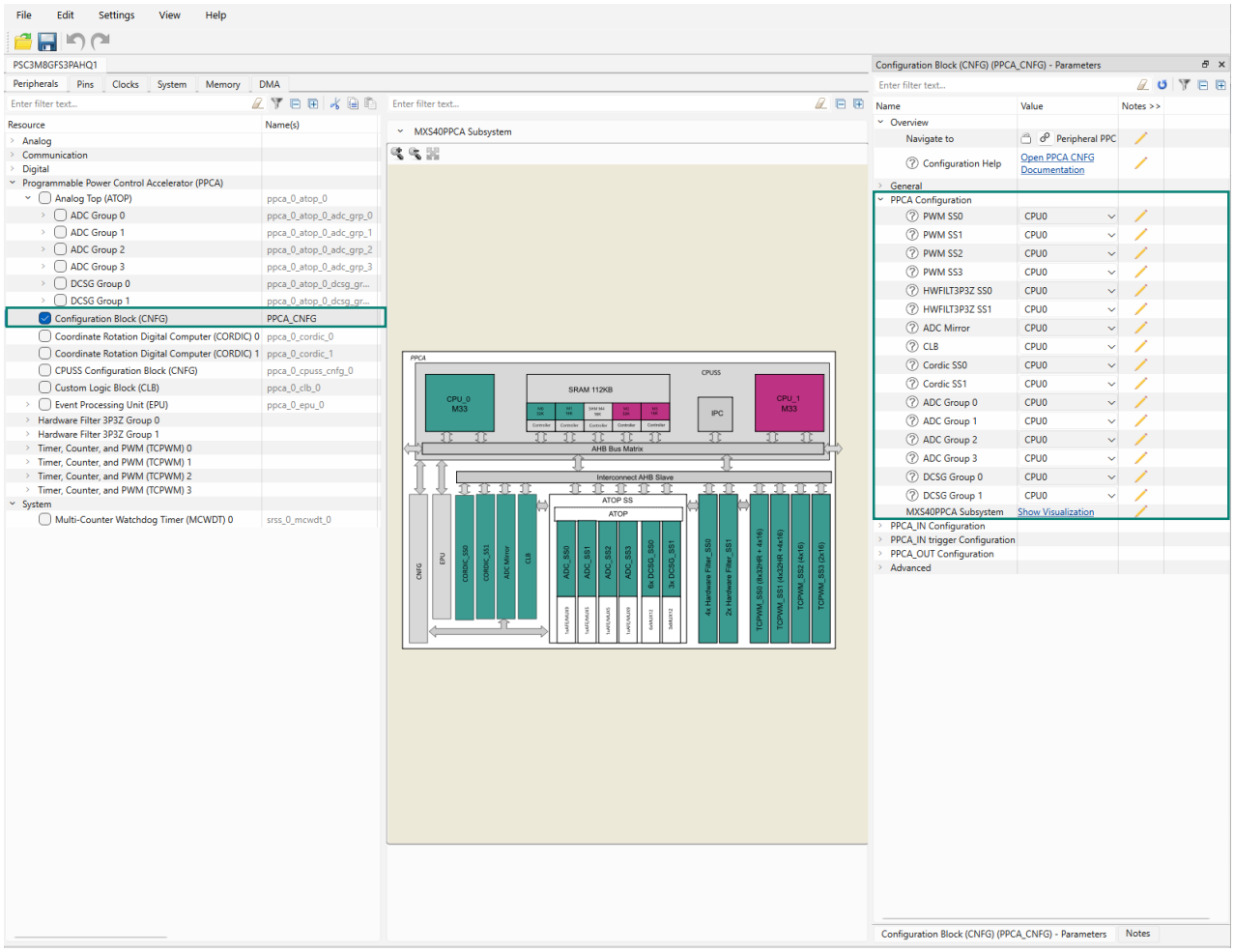


Figure 8 PPCA configuration

2.2.4.2.3 PPCA input configuration

Configure the source of the input routed to the PPCA boundary.

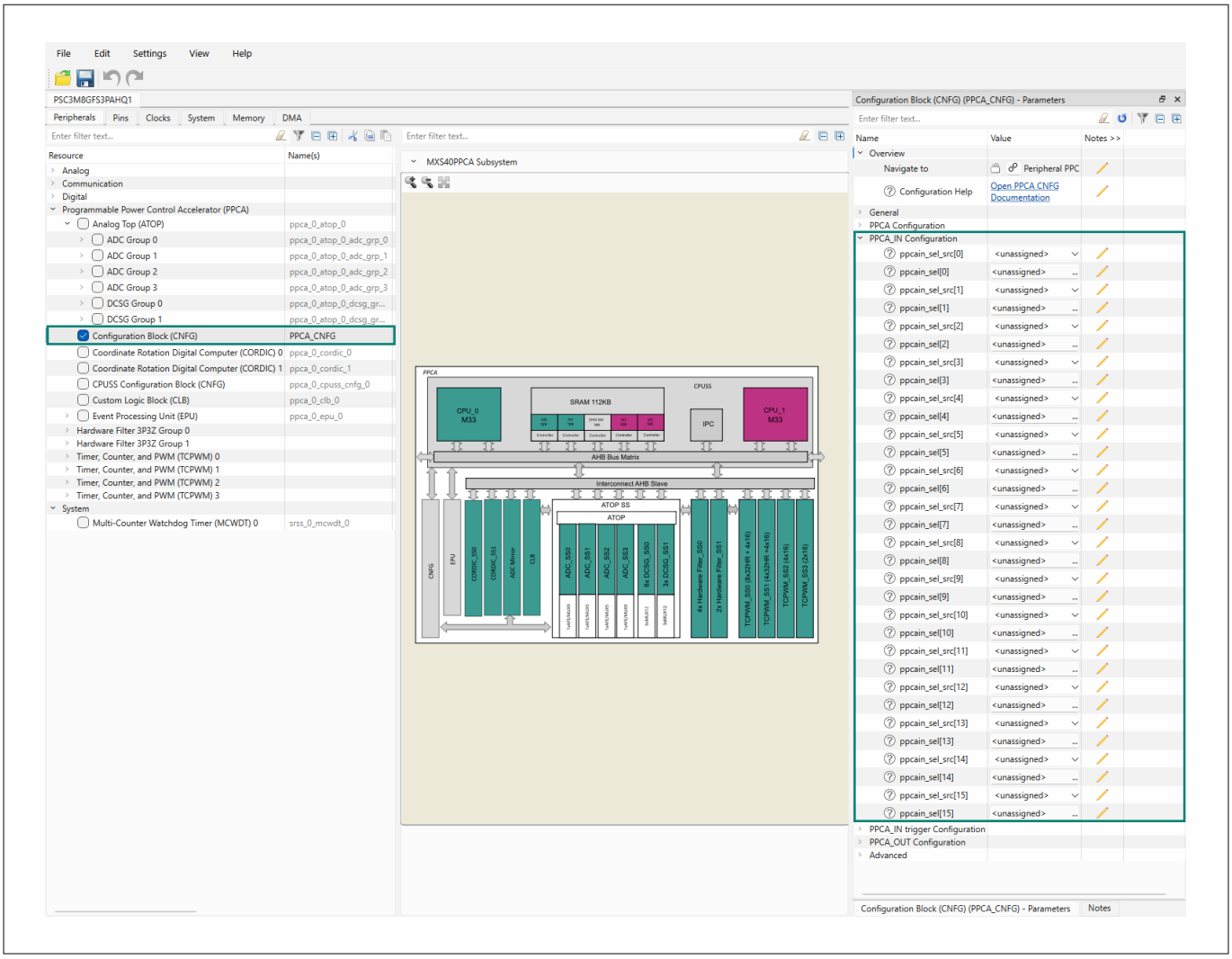


Figure 9 PPCA input configuration

2.2.4.2.4 PPCA input trigger configuration

Configure the trigger source for inputs entering the PPCA boundary.

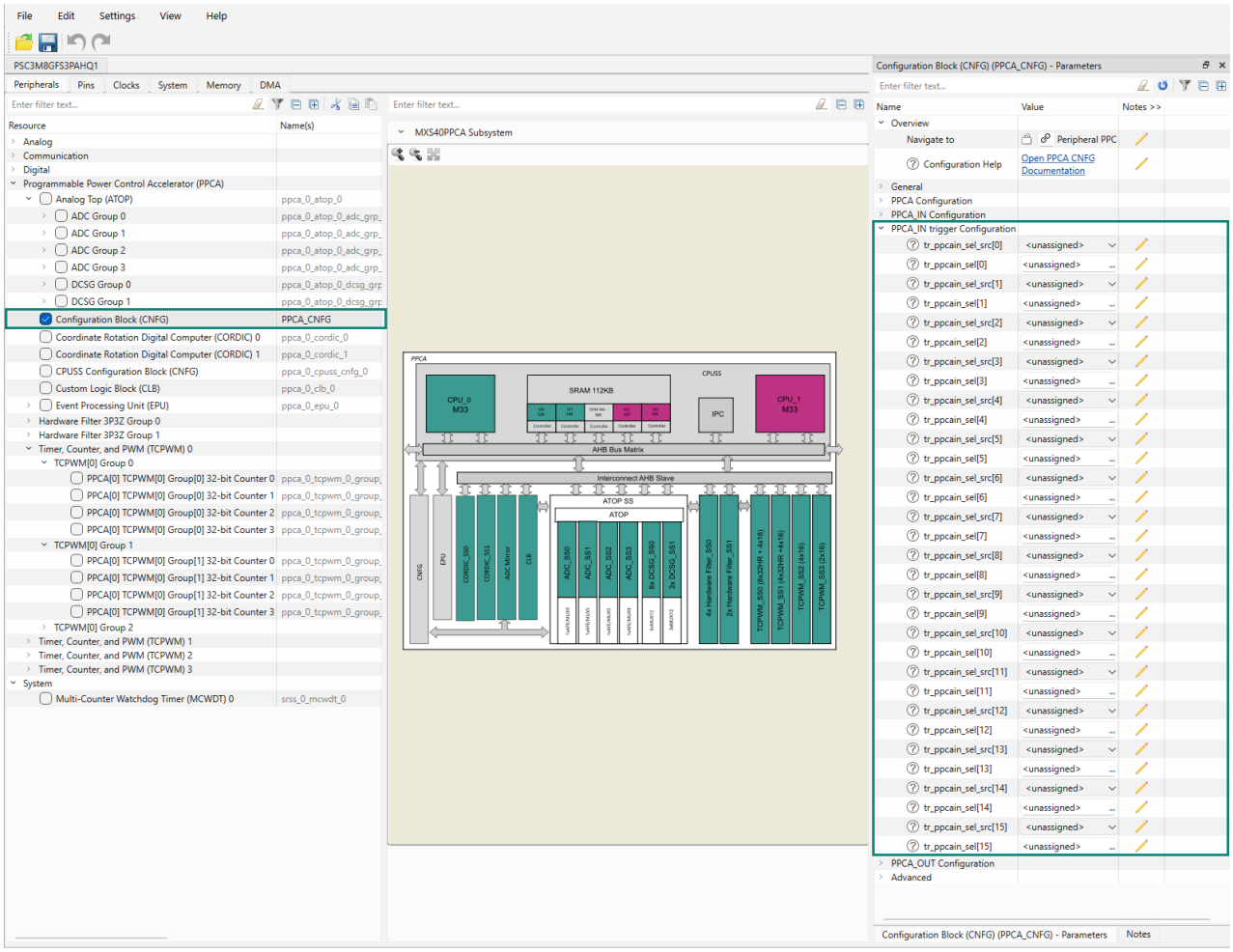


Figure 10 PPCA input trigger configuration

2.2.4.2.5 PPCA output configuration

Configure PPCA boundary outputs, selecting between TCPWM or CLB.

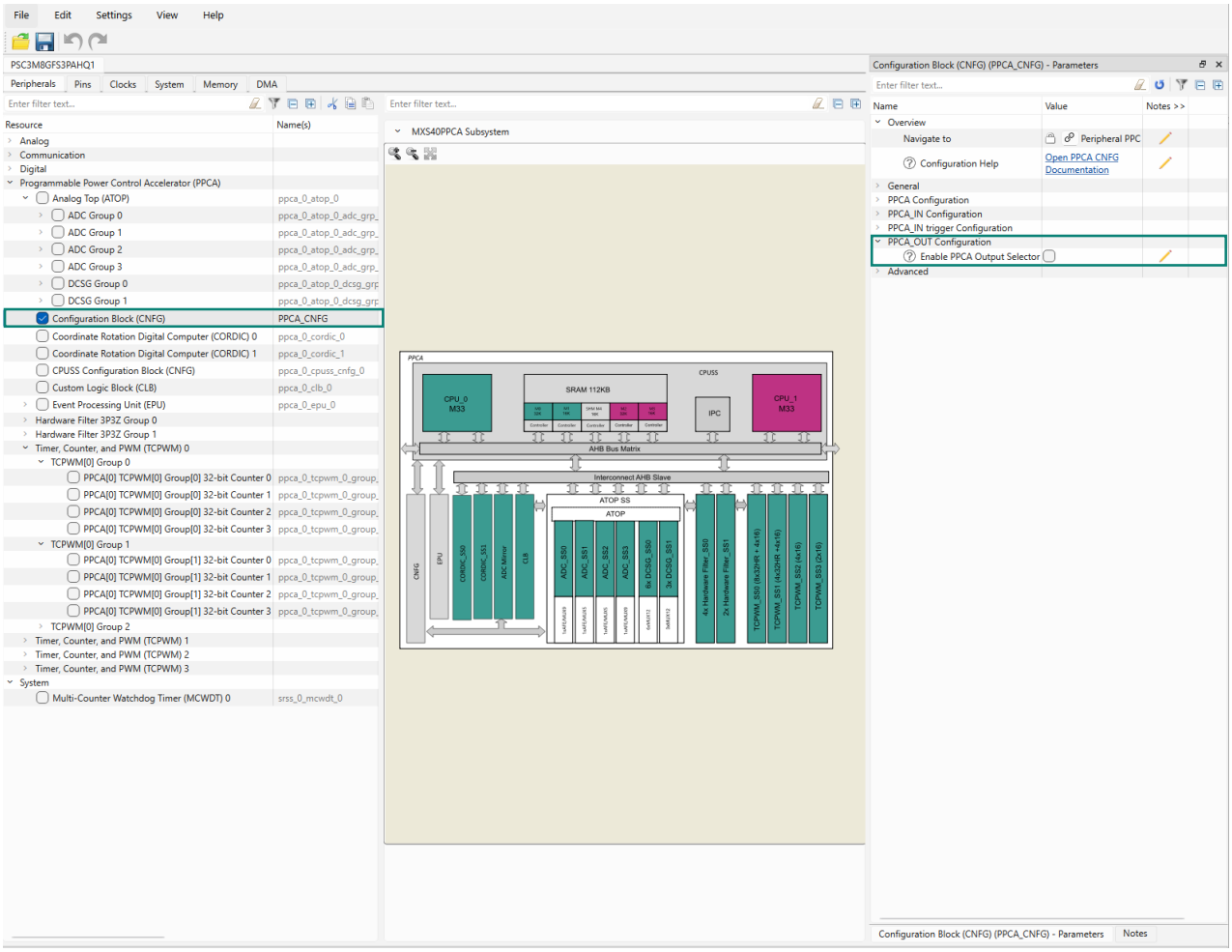


Figure 11 PPCA output configuration

As shown in the figure, you can use the Device Configurator for performing resource allocation to the PPCA CPUs, input selection, input trigger selection, PPCA output selection, etc.

PDL APIs for PPCA configuration are located in `\drivers\include\cy_ppca_cnfg.h` and `\drivers\source\cy_ppca_cnfg.c` files.

The following code snippet shows the initialization code for the PPCA Configuration block. Include it in your program to use the PPCA CNFG functionality.

```
/* Initializing the PPCA Configuration. */
/* The PPCA manages the two PPCA CM33 cores (CPU0 and CPU1). */
Cy_PPCA_CNFG_Init(PPCA_CNFG_HW, &PPCA_CNFG_config);

/* Enabling the PPCA Configuration. */
Cy_PPCA_Enable(PPCA_CNFG_HW);
```

2.2.4.3 PPCA ATOP

The Analog TOP (ATOP) subsystem within the PPCA block contains multiple analog resources required in most industrial control solutions. ATOP contains the following resources:

- ADCs
- Analog reference
- DCSGs
- R2R DACs
- Filters and digital comparators for ADC postprocessing
- Temperature sensor

Within the ATOP subsystem, the resources are organized as six groups: four ADC groups and two DCSG groups. In Device Configurator, a folder called "Analog Top (ATOP)" is available. The resources in the Device Configurator are also grouped in the same way as shown in Figure 12.

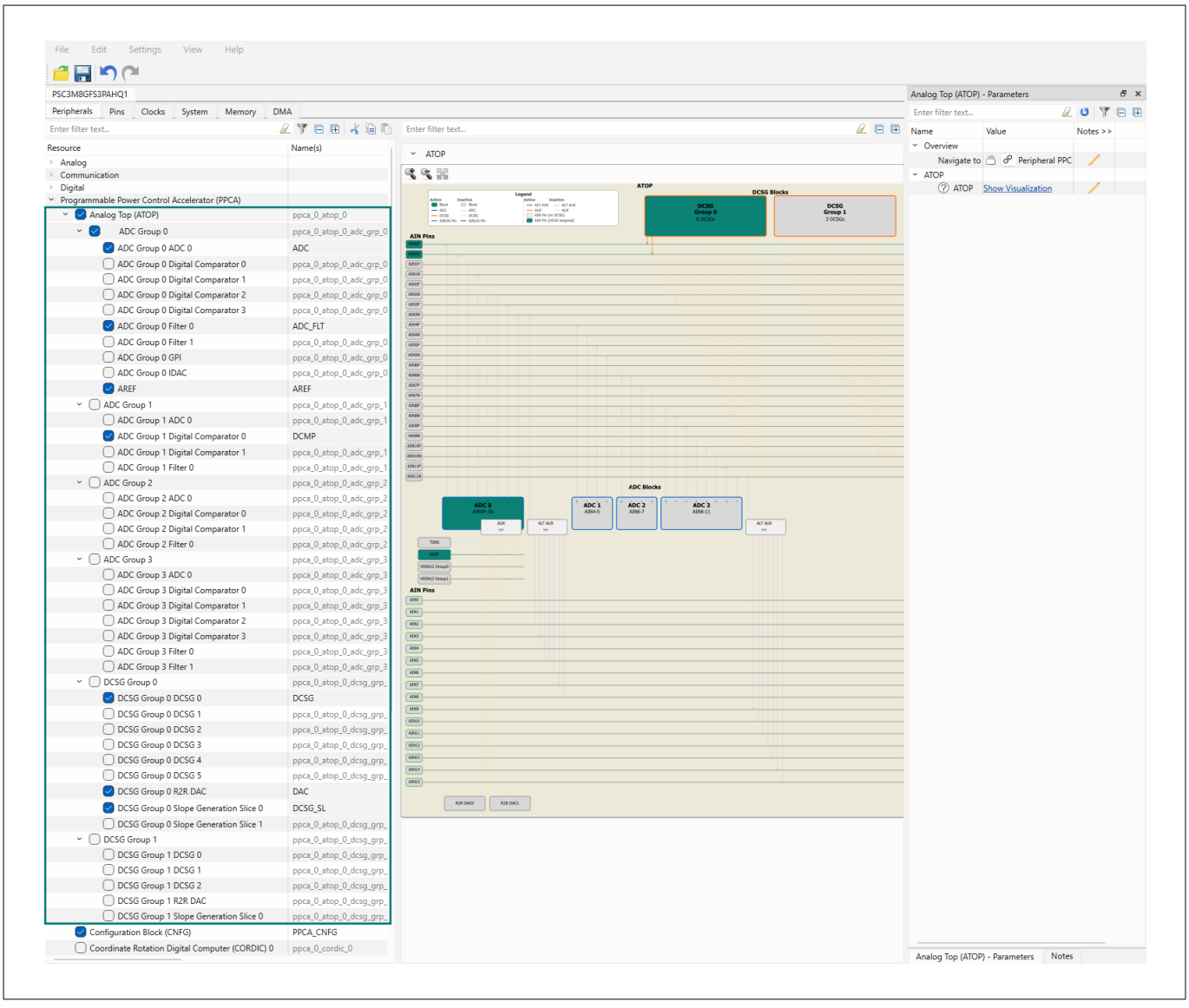


Figure 12 ModusToolbox™ - Device Configurator - ATOP configuration

2.2.4.3.1 ATOP analog reference (AREF)

Configure the reference voltage and current of the analog modules in the ATOP subsystem. Enable AREF to use PPCA analog resources.

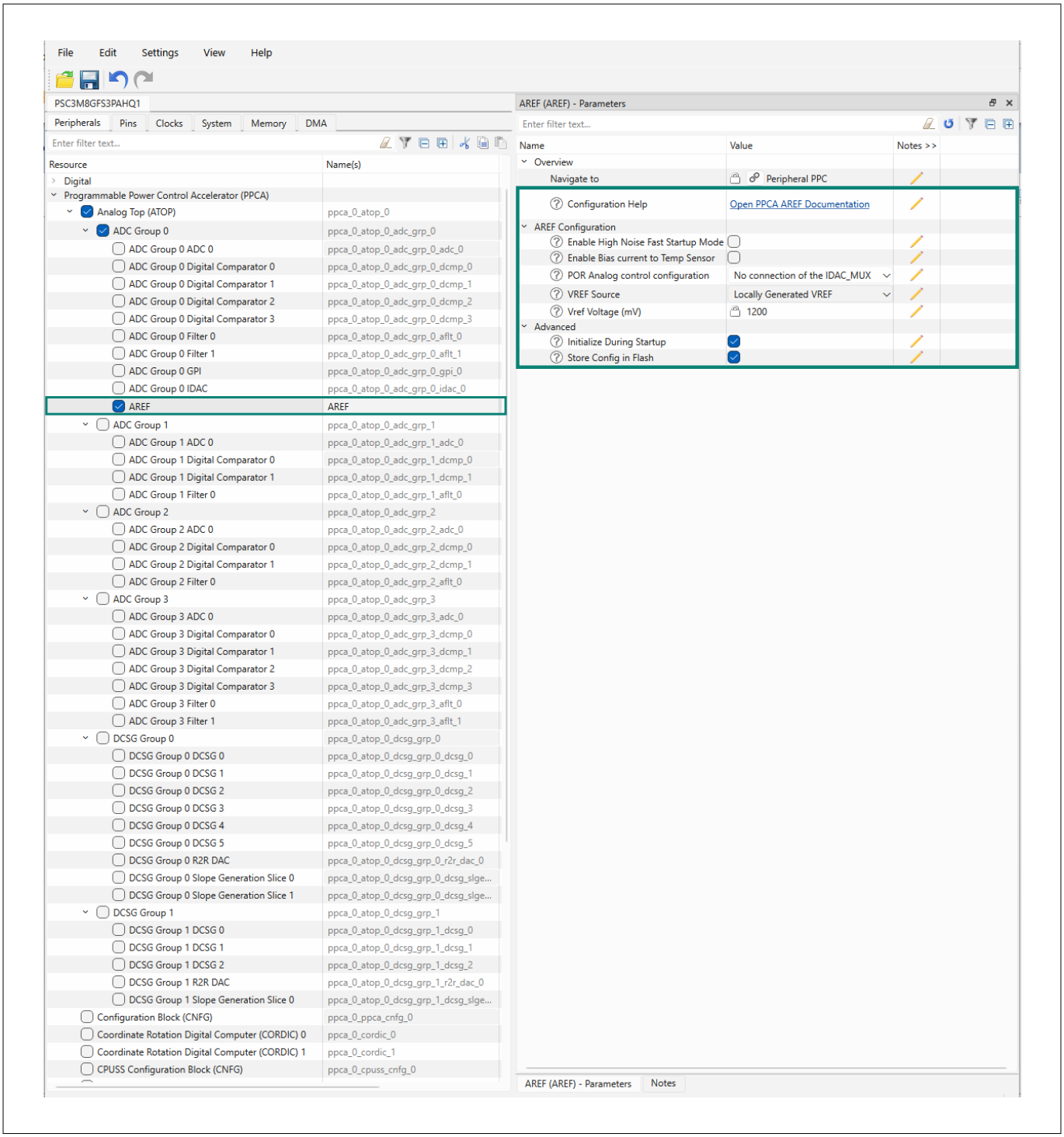


Figure 13 ATOP AREF configuration

The following code snippet shows the initialization code for the ATOP peripherals. Include it in your program to use the ATOP functionality. The PDL functions for the ATOP are grouped based on the functionality. The PDL functions, structures, and macros for ATOP resources are in the following files:

`\drivers\include\cy_ppca_aref.h`

\drivers\source\cy_ppca_aref.c

```
/* Initializing ATOP Analog reference */
Cy_PPCA_AREF_Init(AREF_HW, &AREF_config);

/* Enabling AREF */
Cy_PPCA_AREF_Enable(AREF_HW);
```

2.2.4.3.2 ATOP ADC

Configure ADC, auxiliary, and alternate auxiliary channels, as well as groups ADC channels, select ADC trigger modes, set the sampling cycles, and manage the ADC end-of-conversion (EOC) and start-of-conversion (SOC) signals.

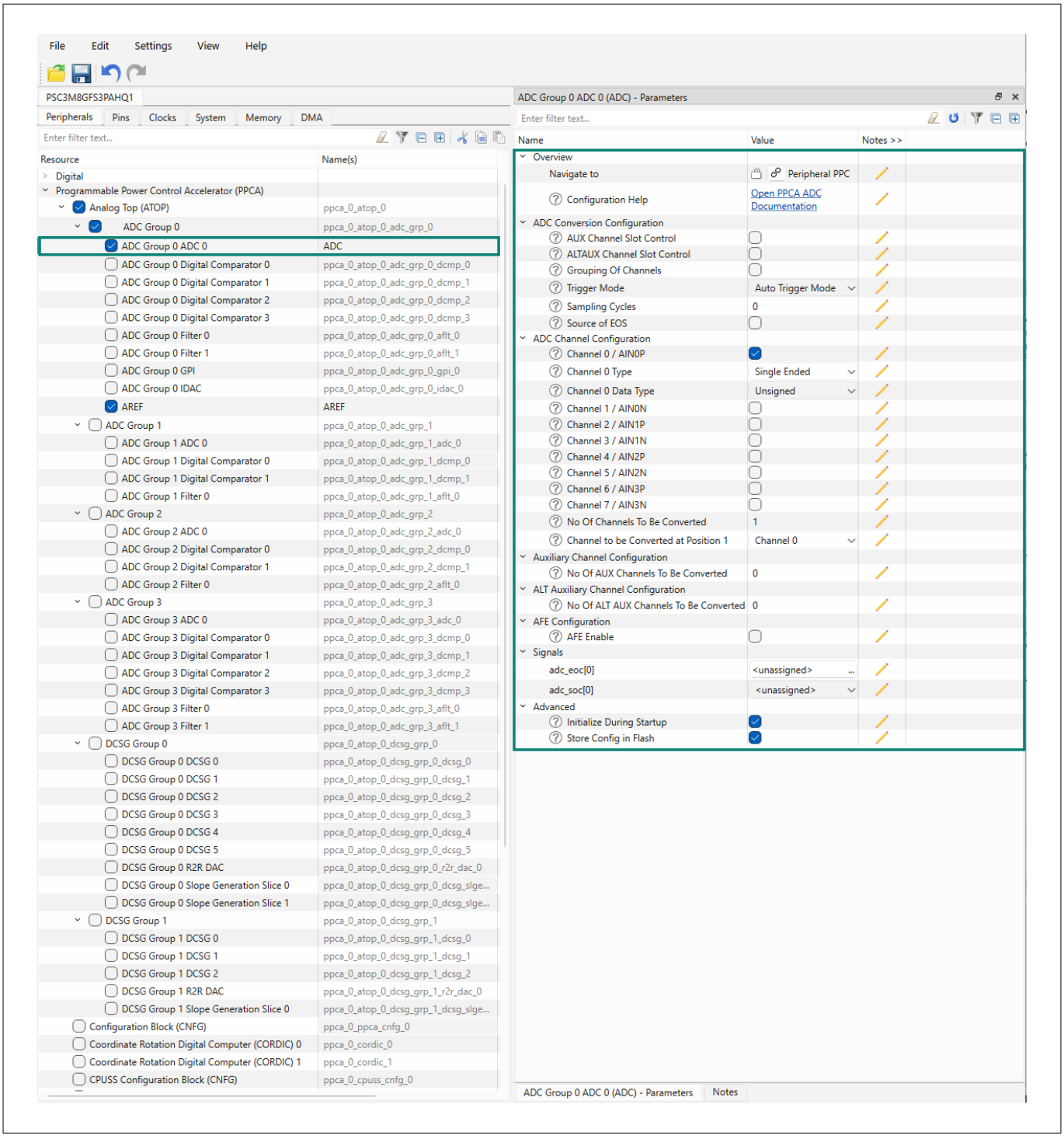


Figure 14 ATOP ADC configuration

`\drivers\include\cy_ppca_adc.h`

\drivers\source\cy_ppca_adc.c

```
    /* Initializing ATOP ADC */
    Cy_PPCA_ADC_Init(ADC_HW, &ADC_config);

    /* Enabling ADC */
    Cy_PPCA_ADC_Enable(ADC_HW);
```

2.2.4.3.3 ATOP ADC filter

Enable and bypass the filters.

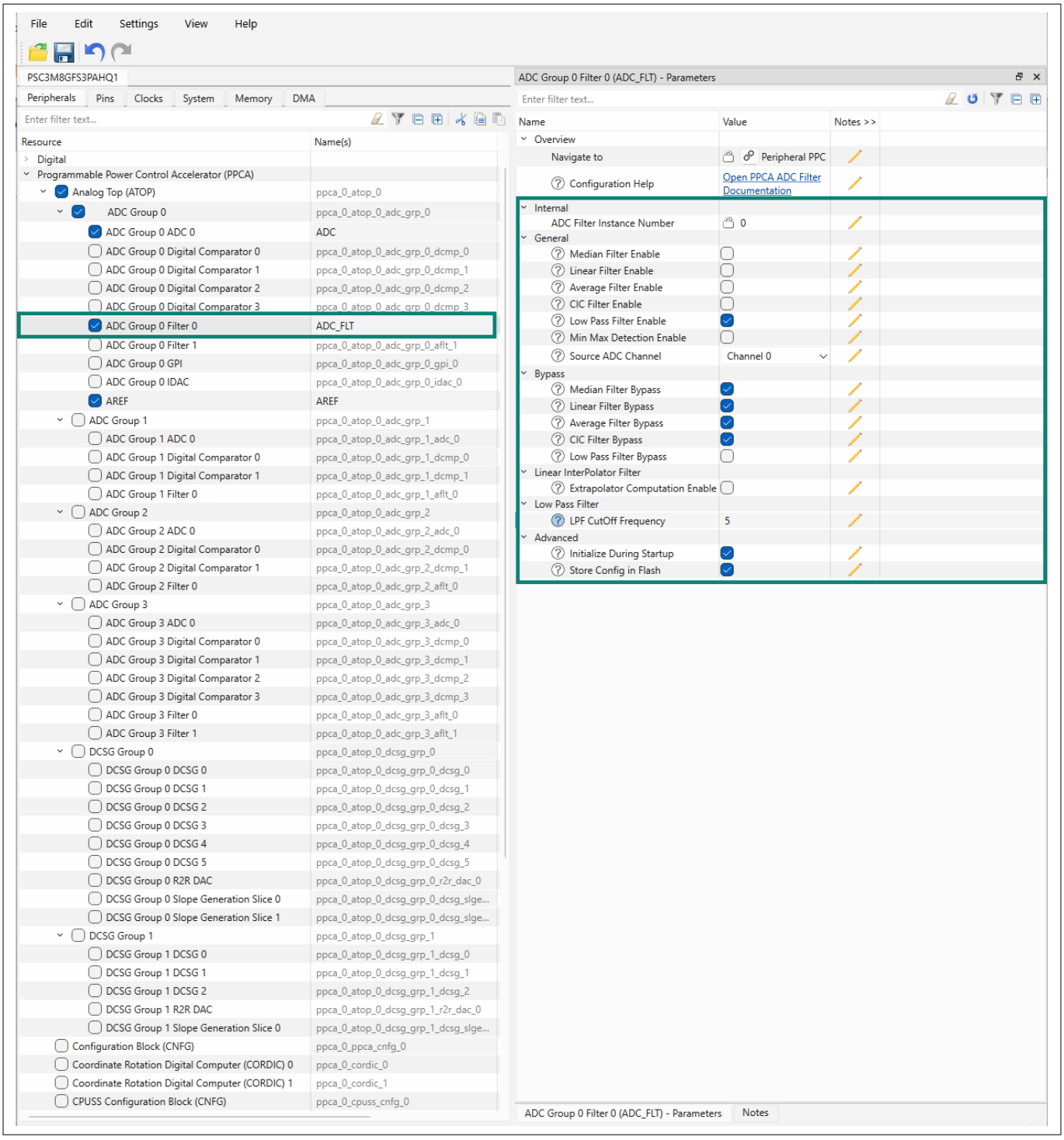


Figure 15 ATOP ADC filter configuration

`\drivers\include\cy_ppca_adc_filter.h`

\drivers\source\cy_ppca_adc_filter.c

```
/* Initializing ADC filter */
Cy_PPCA_ADC_Filter_Init(ADC_FLT_HW,&ADC_FLT_config);

/* starting ADC average filter */
Cy_PPCA_ADC_AVG_Filter_Trigger_Start(ADC_FLT_HW);
```

2.2.4.3.4 ATOP R2R DAC

Enable and configure the digital-to-analog converter (DAC). To use the DAC, enable the digital comparator and slope group (DCSG) of the particular group.

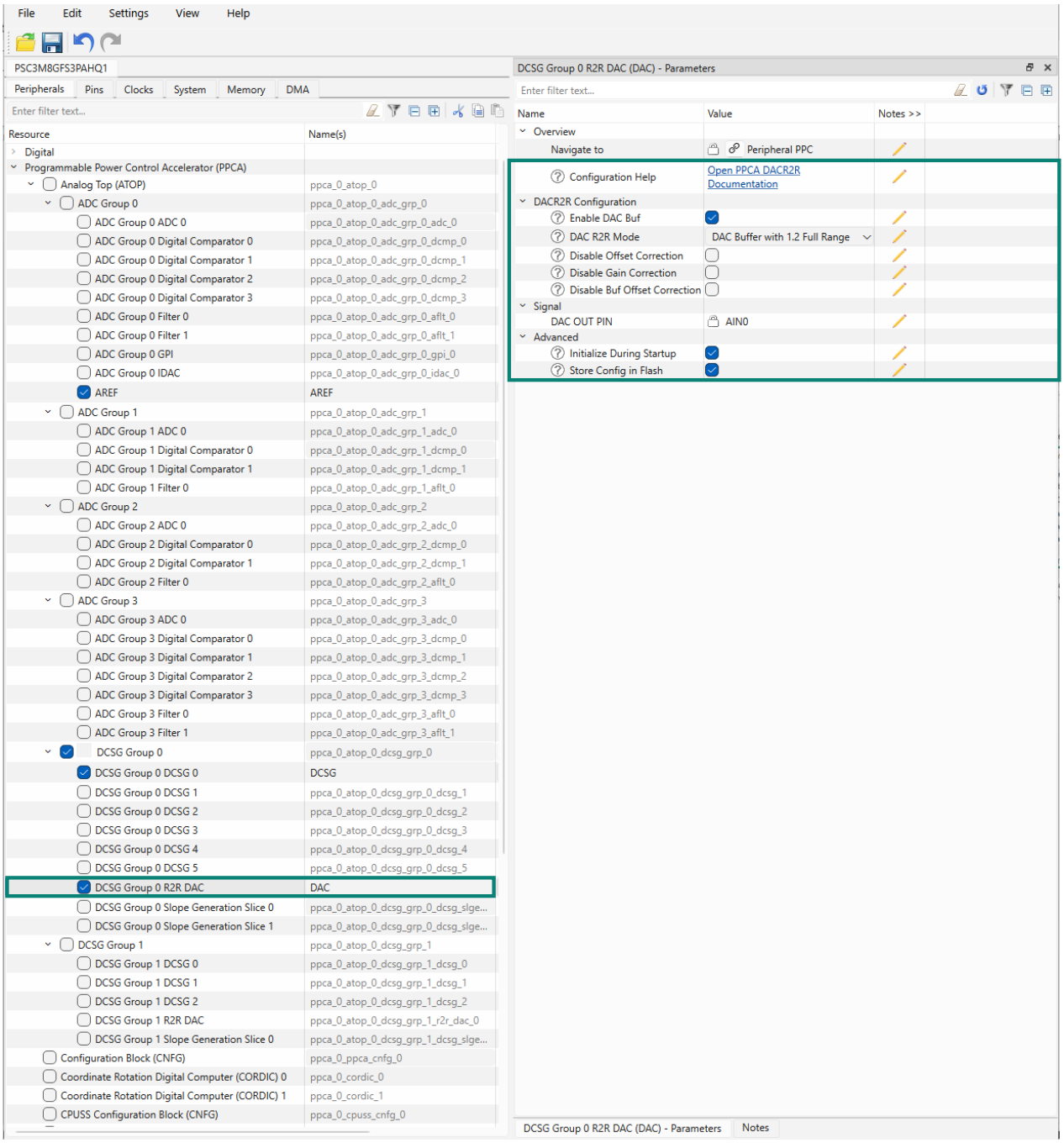


Figure 16 ATOP DAC configuration

`\drivers\include\cy_ppca_dacr2r.h`

\drivers\source\cy_ppca_dacr2r.c

```
    /* Initializing the DAC */
    Cy_PPCA_DAC_Init(DAC_HW, &DAC_config);

    /* Enabling the DAC buffer */
    Cy_PPCA_DACBUF_Enable(DAC_HW);

    /* Enabling the DAC */
    Cy_PPCA_DAC_Enable(DAC_HW);
```

2.2.4.3.5 ATOP DCMP

Enable and configure the digital comparator.

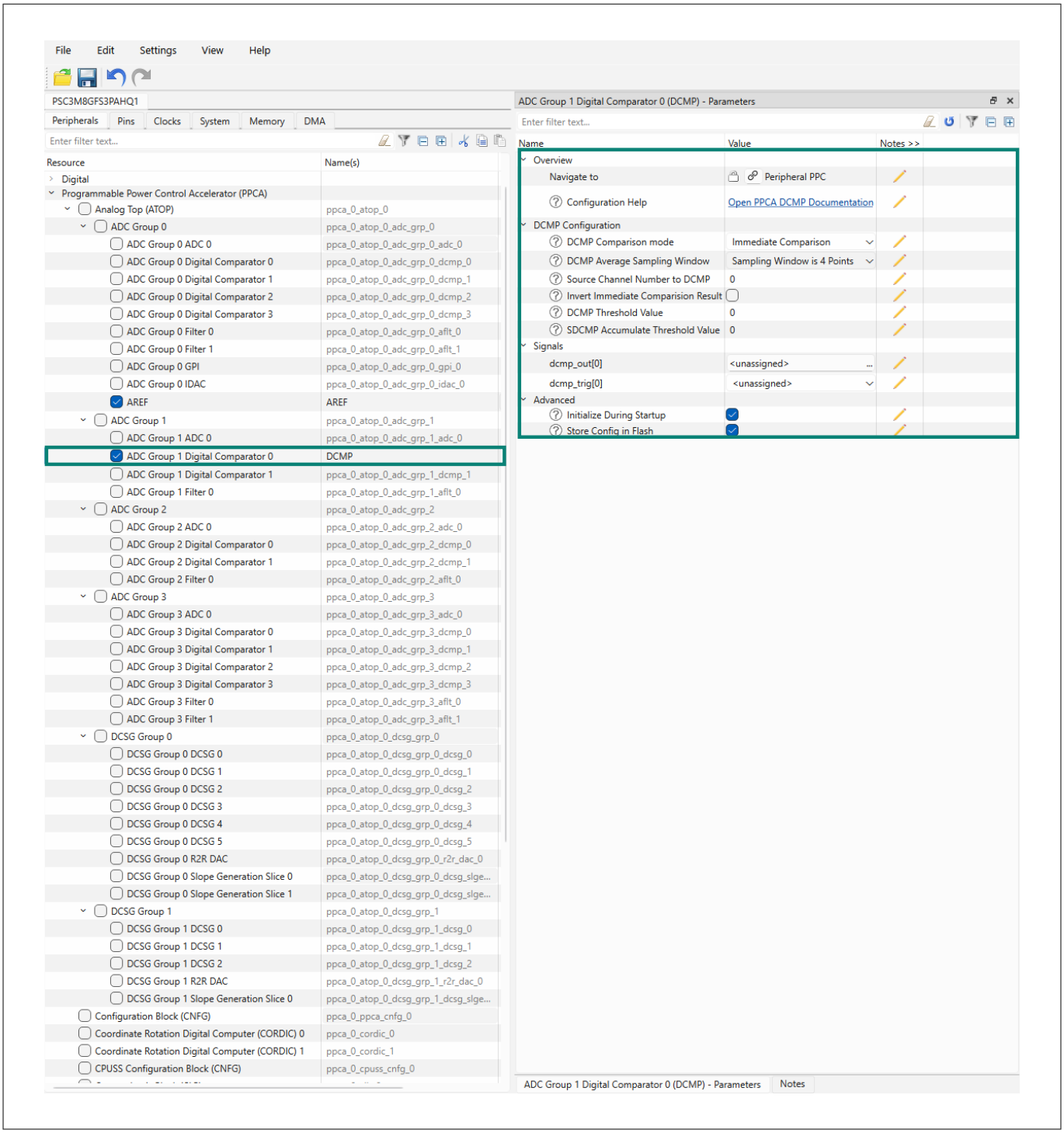


Figure 17 ATOP digital comparator configuration

`\drivers\include\cy_ppca_dcmp.h`

\drivers\source\cy_ppca_dcmp.c

```
    /* Initializing ATOP Digital Comparator */
    Cy_PPCA_DCMP_Init(DCMP_HW, &DCMP_config);

    /* Enabling the digital comparator block */
    Cy_PPCA_DCMP_Enable(DCMP_HW);
```

2.2.4.3.6 ATOP DCSG slope generator

Enable and configure the slope generator.

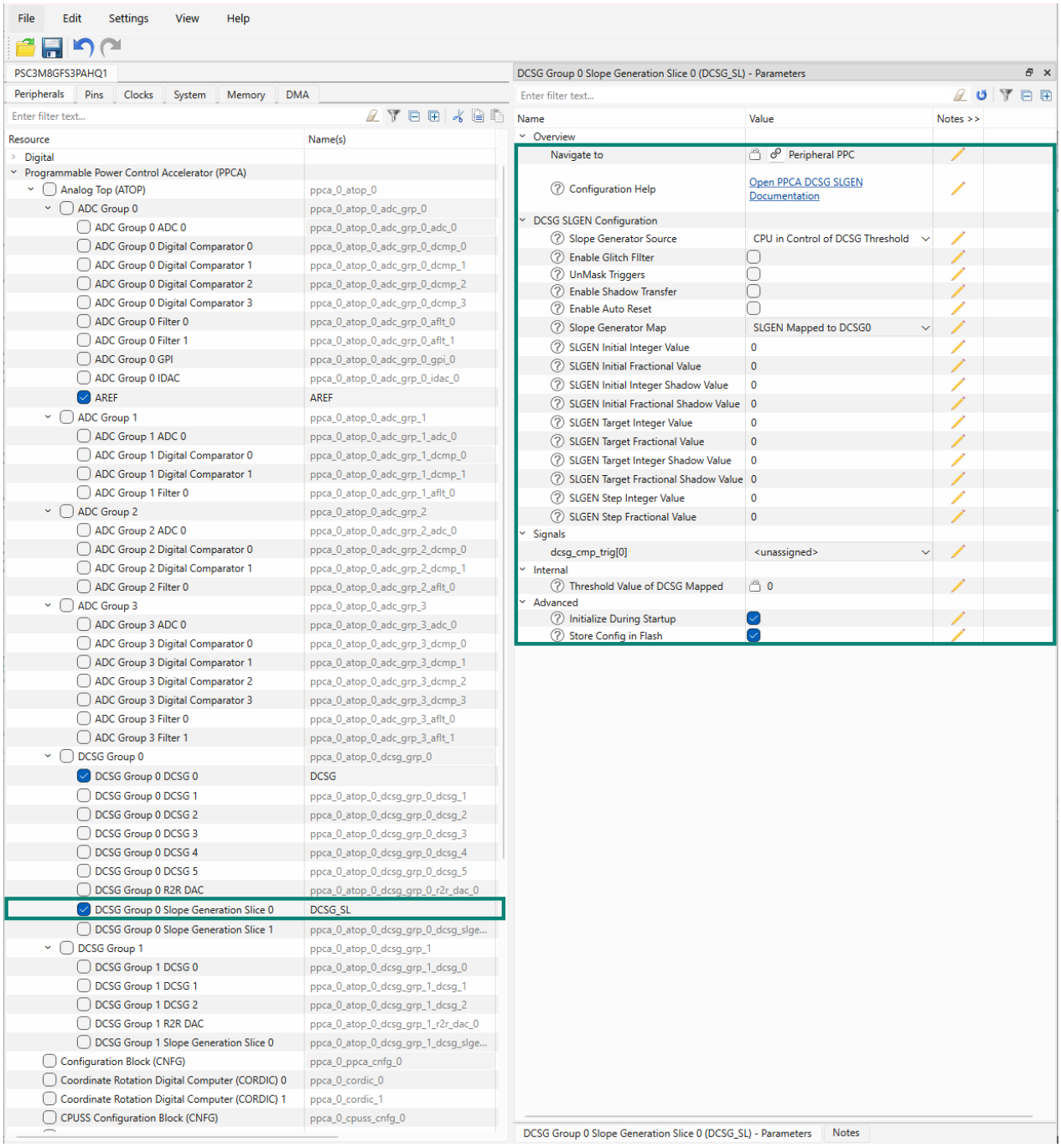


Figure 18 ATOP DCSG slope generator configuration

`\drivers\include\cy_ppca_dcsg_slgen.h`

`\drivers\include\cy_ppca_dcsg_slgen.h`

`\drivers\source\cy_ppca_dcsg.h`

\drivers\source\cy_ppca_dcsg.h

```
/* Initializing DCSG */
Cy_PPCA_DCSG_Init(DCSG_HW, &DCSG_config);

/* Enabling interrupts. */
Cy_PPCA_DCSG_Enable(DCSG_HW);

/* Initializing DCSG slope generation. */
Cy_PPCA_DCSG_SLGEN_Init(DCSG_SL_HW, &DCSG_SL_config);

/* Enabling DCSG slope generation. */
Cy_PPCA_DCSG_SLGEN_Enable(DCSG_SL_HW);

/* Triggering the slope ramp generation. */
Cy_PPCA_DCSG_SLGEN_Trigger_Ramp_Generation(DCSG_SL_HW);
```

2.2.4.4 PPCA TCPWM

Timers are an integral part of the control applications. PPCA contains four instances of TCPWM that can be configured as timer, counter, and PWM. Resources available in PPCA TCPWM are:

- Twelve 32-bit high-resolution counters with <100 ps resolution
- Fourteen 16-bit counters
- Two MOTIF blocks for motor control applications

These resources help to implement power conversion and motor control applications.

In the Device Configurator, four timer instances are kept at different folders. Groups of timer channels are kept as folders under it. Figure 19 shows the PPCA TCPWM configuration in the Device Configurator.

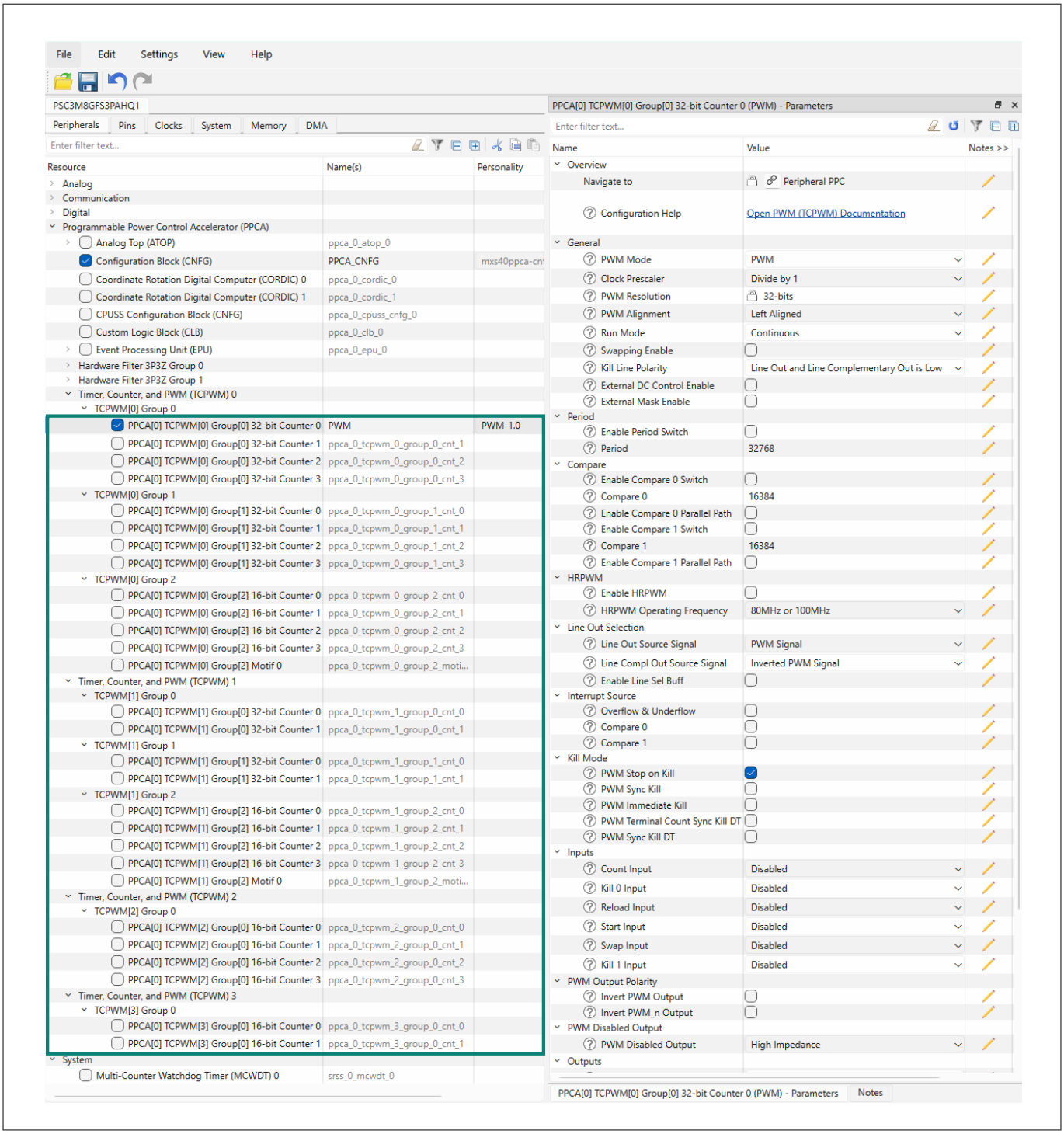


Figure 19 ModusToolbox™ - Device Configurator - PPCA TCPWM configuration

The high resolution option can be enabled in 32-bit timers the **Parameters** pane of the Device Configurator.

The two MOTIF blocks available are part of the TCPWM instance 0 Group 2 and TCPWM instance 1 Group 2, which are shown in Figure 19.

PDL functions for TCPWM are in the following files:

`\drivers\include\cy_tcpwm.h`

`\drivers\include\cy_tcpwm_counter.h`

`\drivers\include\cy_tcpwm_pwm.h`

`\drivers\include\cy_tcpwm_motif.h`

\drivers\source\cy_tcpwm_counter.c

\drivers\source\cy_tcpwm_pwm.c

\drivers\source\cy_tcpwm_motif.c

The following code snippet shows the PPCA TCPWM block. Include it in your program to use the TCPWM functionality.

```
/* Initializing the TCPWM */
Cy_TCPWM_PWM_Init(PWM_HW, PWM_NUM, &PWM_config)

/* Enable the initialized PWM */
Cy_TCPWM_PWM_Enable(PWM_HW, PWM_NUM);

/* Then start the PWM */
Cy_TCPWM_TriggerStart_Single(PWM_HW, PWM_NUM);
```

2.2.4.5 PPCA custom logic block (CLB)

PPCA CLB is a module dedicated for power conversion applications. PPCA contains a single instance of the CLB block. CLB has two subsystems called as "digital current observer (DICO)" and "matrix converter (MACO)". DICO is for interleaved multi-phase, multi-mode, totem-pole power factor correction (TP PFC), while MACO is designated for use with the matrix converter control algorithm supporting soft-start for resonant topologies via trajectory control and a commutation matrix. DICO and MACO cannot be enabled together; only one can be used at a time.

In the Device Configurator, the CLB is located under PPCA in the Peripherals pane. Selection of DICO or MACO can be done in the Parameters pane. Based on the selected mode, the options in the Parameters pane and the visual image in the middle will change. Figure 20 shows the configuration when DICO is selected.

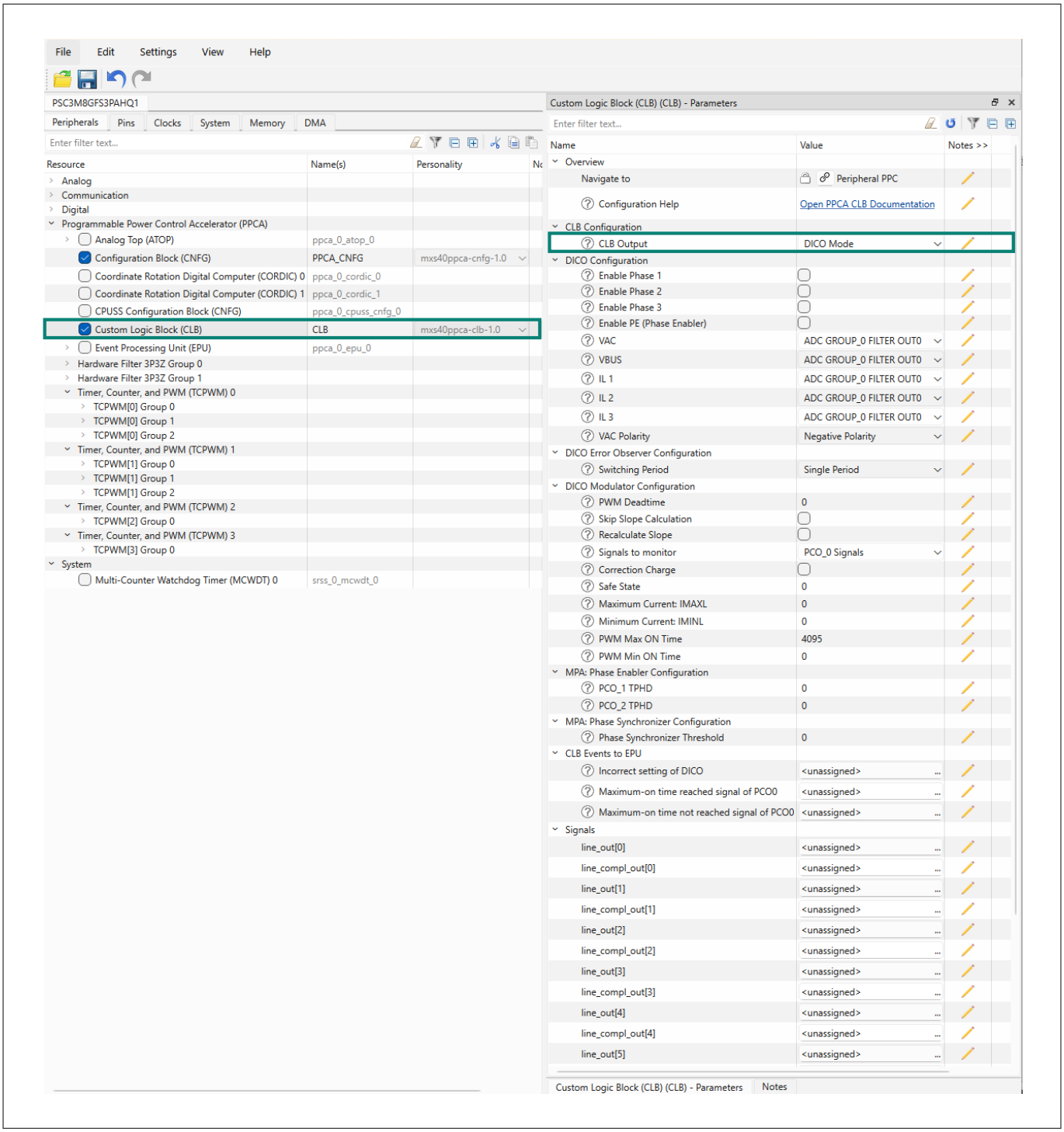


Figure 20 ModusToolbox™ - Device Configurator - CLB - DICO configuration

Similarly, Figure 21 shows when MACO is selected.

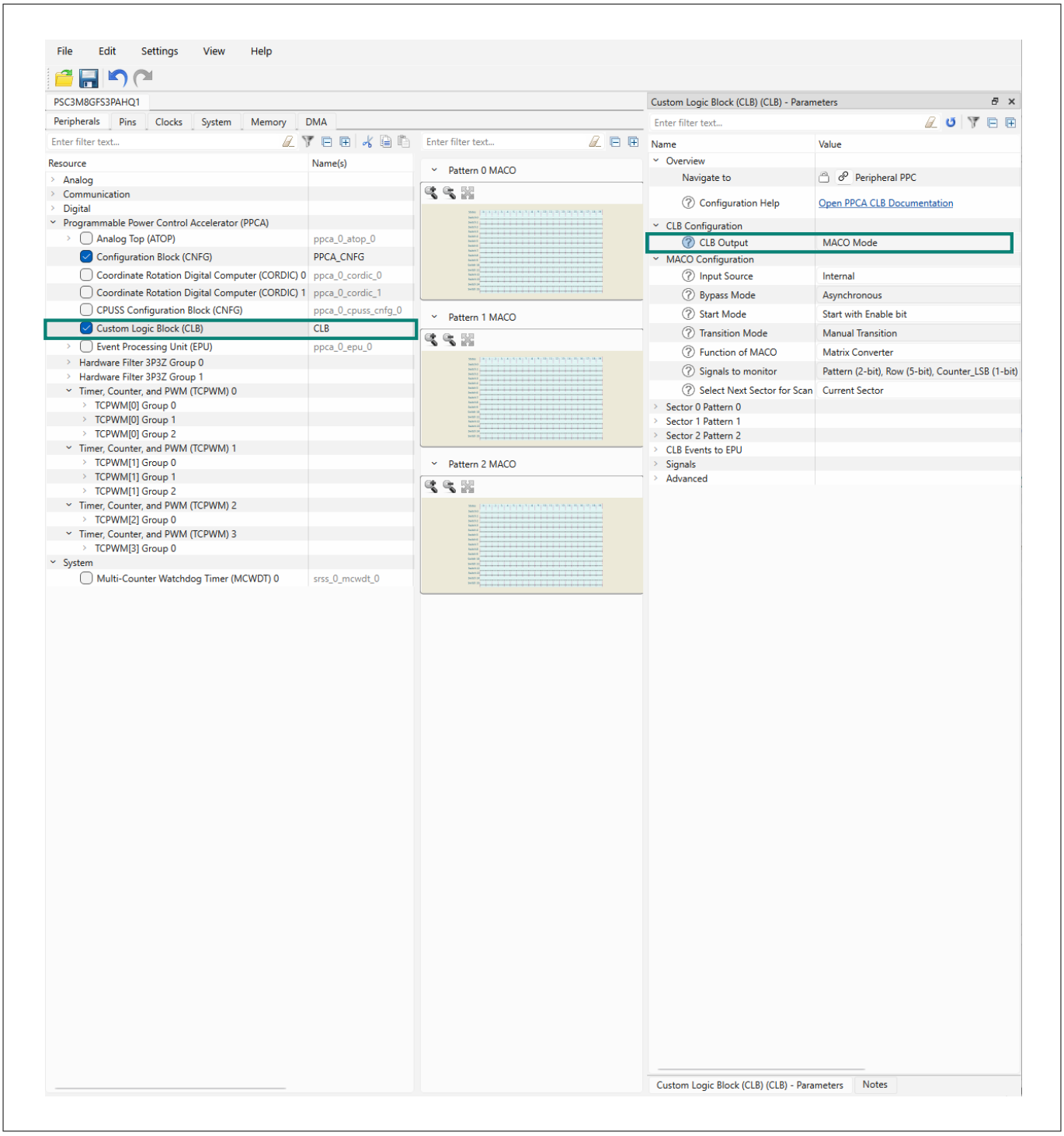


Figure 21 ModusToolbox™ - Device Configurator - CLB - MACO configuration

PDL functions for CLB can be found in the following files:

`\drivers\include\cy_ppca_clb.h`

`\drivers\include\cy_ppca_clb_dico.h`

`\drivers\include\cy_ppca_clb_maco.h`

`\drivers\source\cy_ppca_clb.c`

The following code snippet shows the initialization code for the PPCA CLB MACO block. Include it in your program to use the CLB MACO functionality.

```
/* Select CLB output as MACO */
Cy_PPCA_CLB_Select_Output(CLB_HW, CY_CLB_MACO_SEL);

/* Enable CLB */
Cy_PPCA_CLB_Enable(CLB_HW);

/* Initialize and configure CLB MACO */
Cy_PPCA_CLB_MACO_InitConfig(CLB_MACO_HW, &CLB_macoConfig);

/* Enable CLB MACO */
Cy_PPCA_CLB_MACO_Enable(CLB_MACO_HW);
```

2.2.4.6 PPCA event processing unit (EPU)

All input events in the PPCA are routed through the EPU. Each output event is generated by a dedicated combiner. The EPU module is used to create a hardware triggers based on processed events to enable the operation of PPCA subsystem modules, thereby offloading instantiated processors and speeding up the operations. In addition, the EPU is used to send interrupts to CPU0 or CPU1 as well as at the boundary of PPCA to share triggers between the SoC and PPCA.

The EPU contains processing units, a combiner, and the combiner feedback. These resources are organized in different folders under the EPU folder as shown in Figure 22.

The EPU processing units can be assigned to the required PPCA CPU in the Peripherals pane (when the EPU is enabled). You can also select the combiners for the interrupt in the Peripherals pane.

PDL for the initialization and runtime management of the EPU is available in the following files:

`\drivers\include\cy_ppca_epu.h`

`\drivers\source\cy_ppca_epu.c`

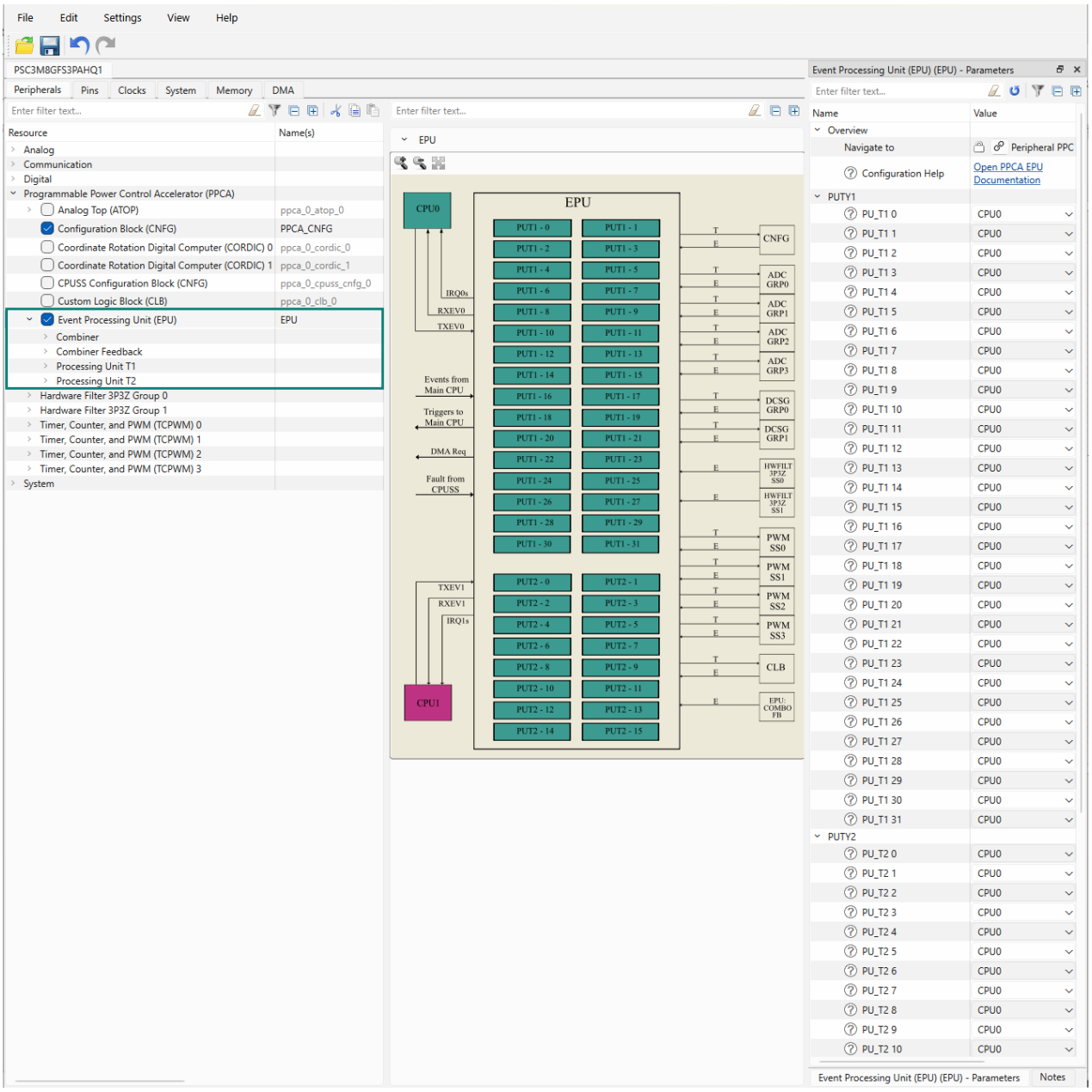


Figure 22 ModusToolbox™ - Device Configurator - EPU configuration

2.2.4.6.1 EPU combiner

Enable the combiner to generate a linear combination of two outputs from processing units T1 and T2.

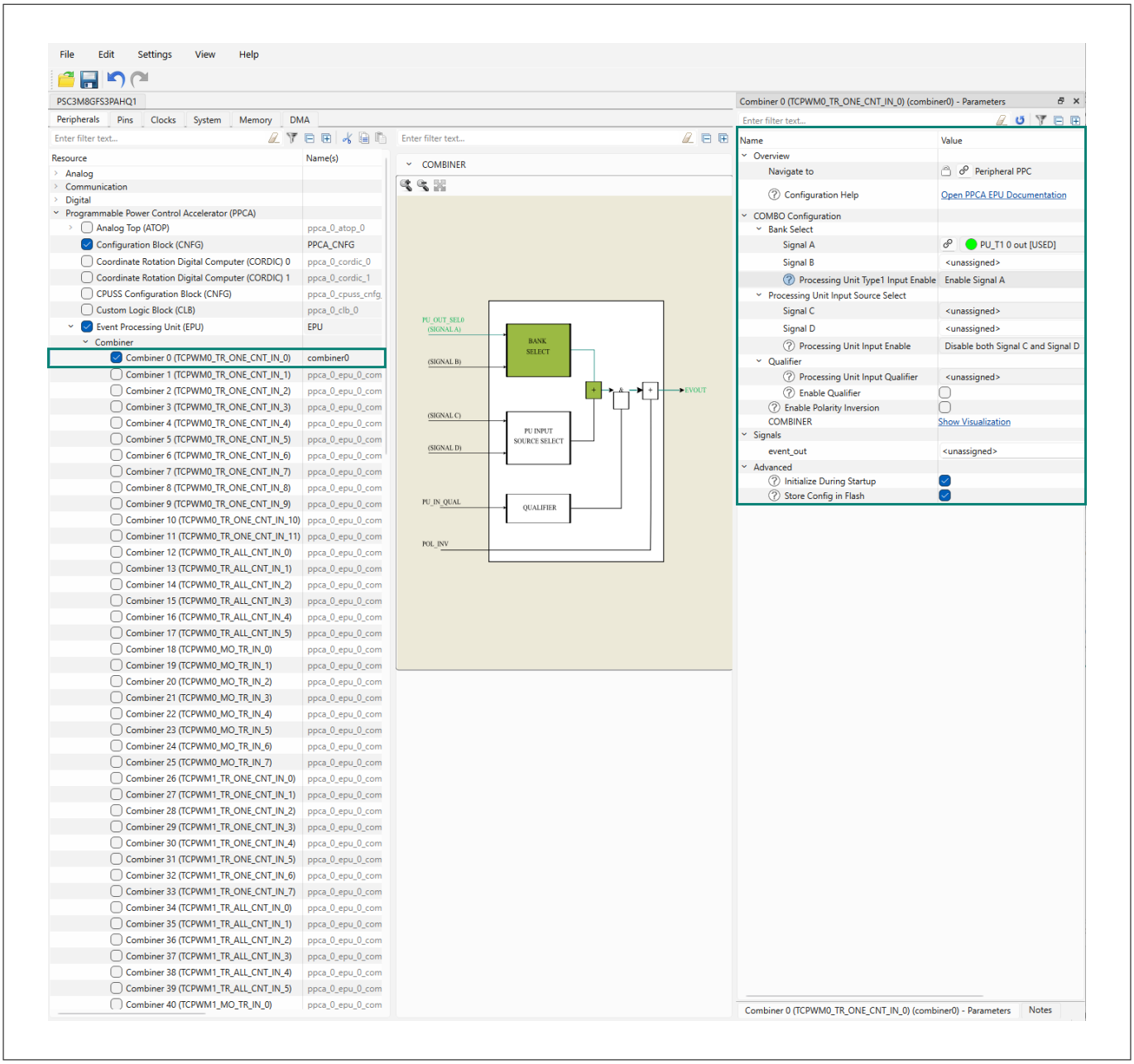


Figure 23 EPU combiner configuration

2.2.4.6.2 EPU combiner feedback

Enable combiner feedback.

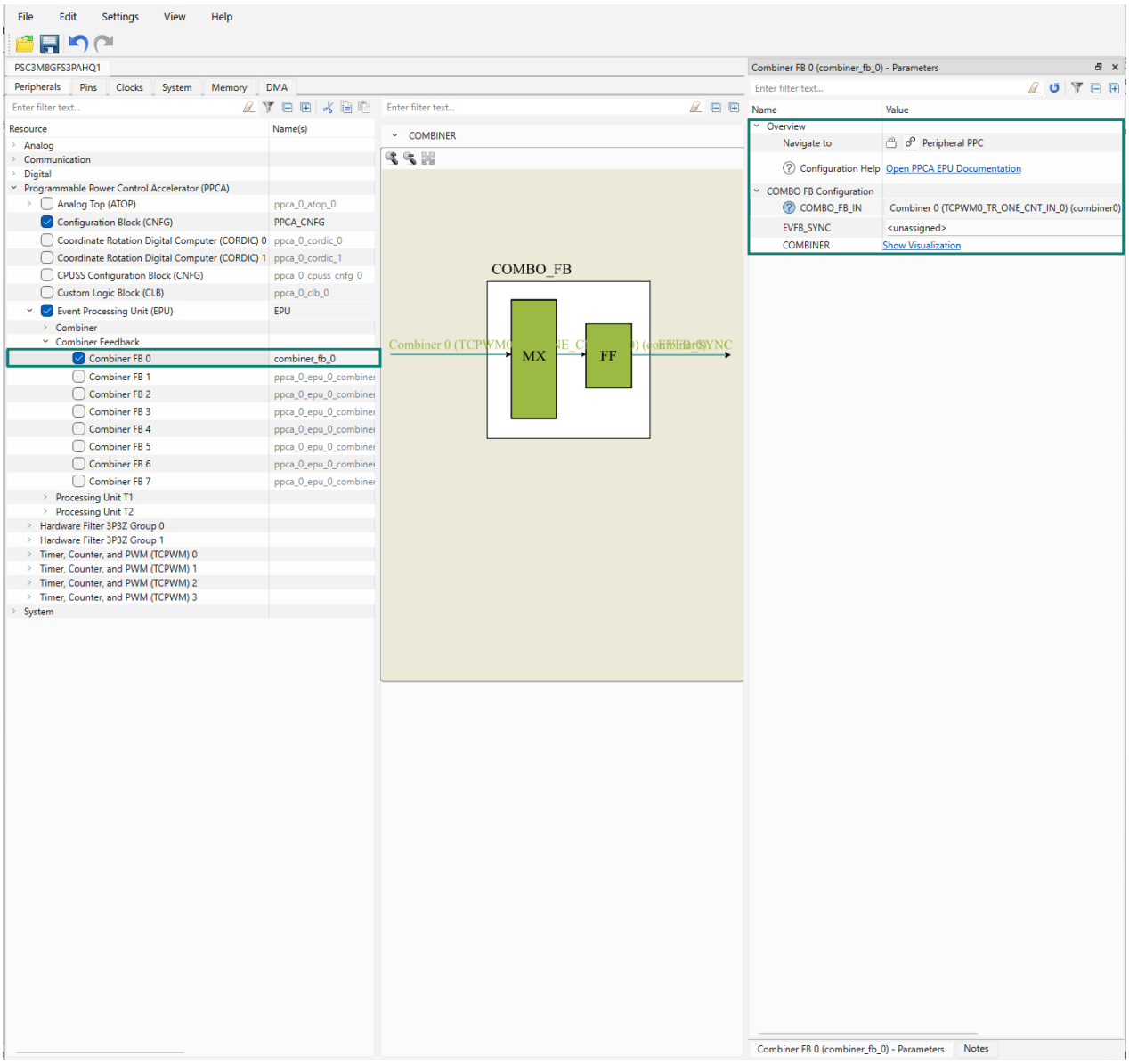


Figure 24 EPU combiner feedback configuration

2.2.4.6.3 EPU Processing Unit 1 (PU1)

Enable EPU PU1 to perform various operations on the input signal, including synchronous or asynchronous bypass, filtering (such as masking, blanking, and debouncing), pulse delay, input polarity inversion, and edge detection.

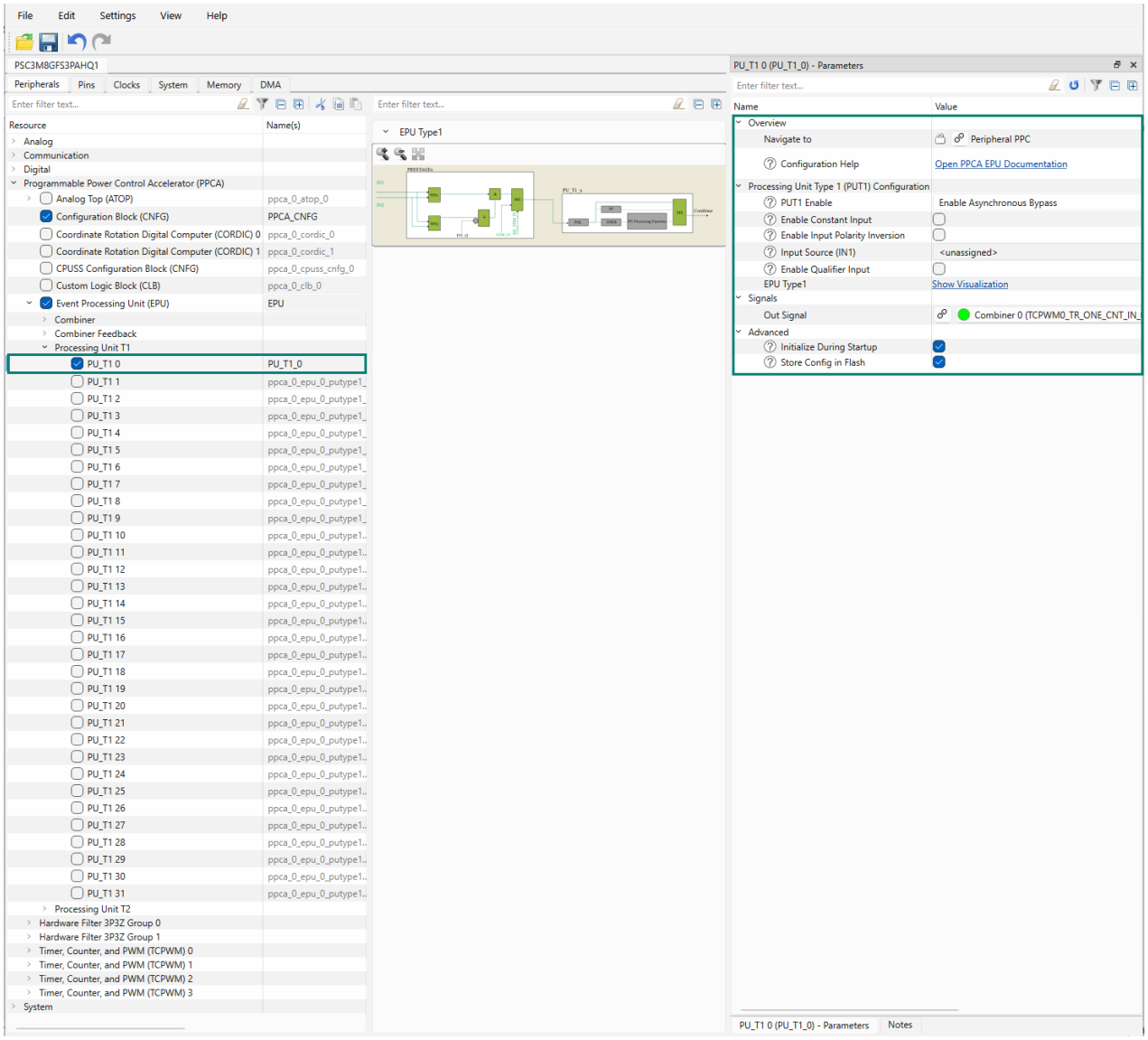


Figure 25 EPU PU1 configuration

2.2.4.6.4 EPU Processing Unit 2 (PU2)

Enable EPU PU2. It is similar to PU1 but has fewer capabilities and reduced flexibility. PU2 does not support the qualification and filtering option available in PU1.

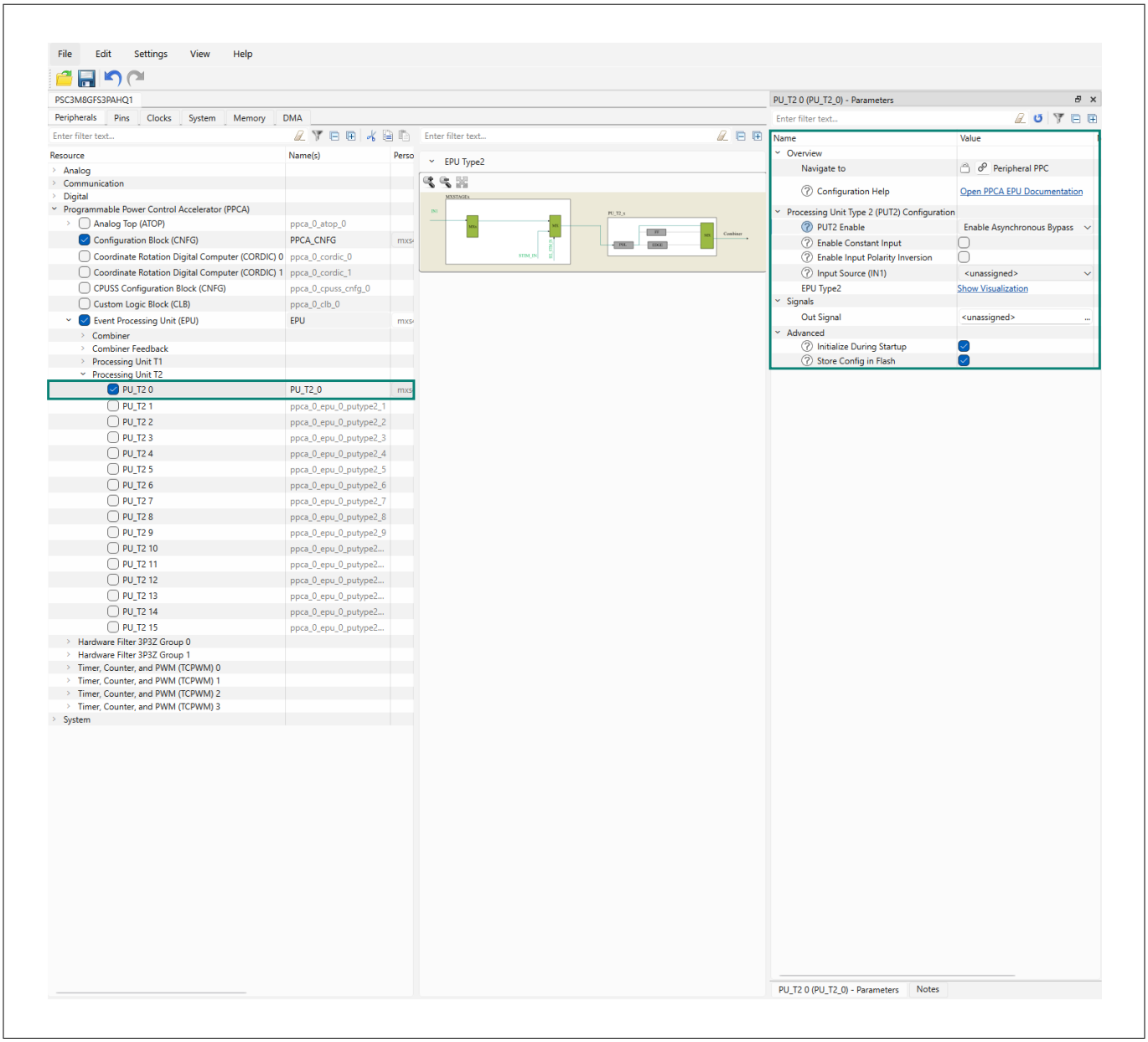


Figure 26 EPU PU2 configuration

The following code snippet shows the initialization code for the PPCA EPU block. Include it in your program to use the EPU functionality.

```
    /* Enable exclusive access to the EPU resources based
      * on the provided resources allocation configuration. */
     Cy_PPCA_EPU_EnableExclusiveAccess(EPU_HW, true);

    /* Enable the EPU */
     Cy_PPCA_EPU_Enable(EPU_HW);

    /* configure the combiner */
     Cy_PPCA_EPU_Combo_Configure(combiner_HW, combiner_INDEX, &combiner_combo_config);

    /* configure the combiner feedback */
     Cy_PPCA_EPU_Combo_Feedback_Configure(combiner_fdb_HW, combiner_fdb_INDEX,
combiner_fdb_SRCIN);

    /* configure the processing unit 1 */
    Cy_PPCA_EPU_PU_T1_Configure(PU_T1_HW, PU_T1_INDEX, &PU_T1_put1_config);

    /* configure the processing unit 2 */
    Cy_PPCA_EPU_PU_T2_Configure(PU_T2_HW, PU_T2_INDEX, &PU_T2_put2_config);
```

2.2.4.7 Hardware 3-pole, 3-zero (3P3Z) filter

PPCA has a three poles/three zeros (3P3Z) configurable filter. Two hardware filter groups with total six channels are available in PPCA. These can be used either by accessing it from the CPU or through the dedicated data path. See the architecture reference manual for more details.

In the Device Configurator, the 3P3Z filter is available under the PPCA folder as shown in Figure 27. Options are available in the Parameters pane to set the filter coefficients, limits, access mode, etc.

For initializing the peripheral and accessing it during the runtime, PDL resources are available in the files listed below.

`\drivers\include\cy_ppca_hwfilt3p3z.h`

`\drivers\source\cy_ppca_hwfilt3p3z.c`

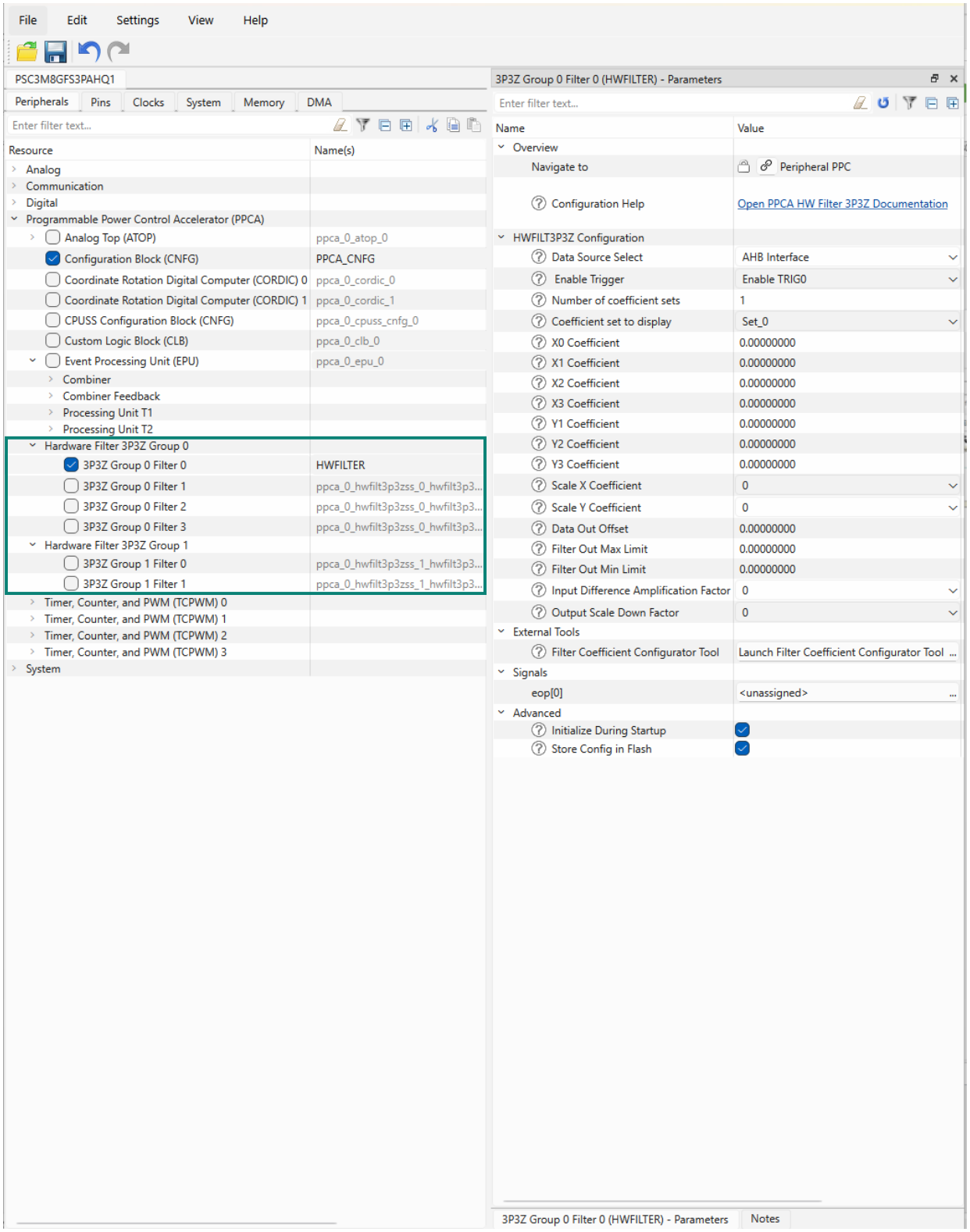


Figure 27 ModusToolbox™ - Device Configurator - HW 3P3Z configuration

The following code snippet shows the initialization code for the 3 pole 3 zero hardware filter block. Include it in your program to use the 3P3Z functionality.

```
/* Enable Hardware filter peripheral */
Cy_PPCA_HWFILT3P3Z_SS_PeripheralEnable(HWFILTER_SS_HW);

/* Enable Hardware filter */
Cy_PPCA_HWFILT3P3Z_FilterEnable(HWFILTER_HW, true);

/* Initialize and configure Hardware filter */
Cy_PPCA_HWFILT3P3Z_InitFilterConfig(HWFILTER_HW, &HWFILTER_Set_0_config);
```

2.2.4.8 PPCA CORDIC

The CORDIC module implements hardware support to perform trigonometric functions and the square-root function. Two CORDIC instances are available in the PPCA, which do not require any initialization to use. Therefore, the Device Configurator does not support CORDIC modules.

PDL support for CORDIC can be found in the following files:

`\drivers\include\cy_cordic.h`

`\drivers\source\cy_cordic.c`

3 Solutions using PPCA

The PPCA is a specialized peripheral embedded in PSC3M/P8, PSC3M/P7 MCUs designed to ease the implementation of advanced digital control techniques in power applications and push their performance boundaries. It integrates multiple hardware modules to realize high-bandwidth, low-latency control loops required for modern power supplies, such as those using GaN technology. The PPCA includes:

- **ADC:** High-speed parallel sampling ADCs with up to 25 Msps for precise and fast signal acquisition
- **Measurement filtering (part of ADC):** Integrated filter module to post process ADC samples, signals and reduce noise, ensuring stability and accuracy:
  - Median filter (three sample window)
  - Linear interpolation/extrapolation filter (LINTP)
  - Low-pass filter (LP)
  - Cascade integrator 3rd-order comb filter (CIC3)
  - Configurable average filter (AVG)
  - Detection of minimum of the input data stream
  - Detection of maximum of the input data stream
  - Configurable minimum and maximum saturation limits
- **3P3Z compensator:** A three-pole, three-zero compensator used as a regulator to implement control algorithms such as proportional-integral-derivative (PID) or custom transfer functions
- **PWM:** High-resolution PWM generation with duty cycle control for driving power stages
- **Configurable logic block (CLB):** Allows customization of logic functions to enhance control and signal processing
- **Event processing unit (EPU):** Handles real-time events, synchronization, and triggers
- **CORDIC**

3.1 Average Current Mode Control (ACMC) strategy using PPCA

The PPCA enhances the implementation of **Average Current Mode Control (ACMC)** by combining high-speed, hardware-accelerated control loops with flexible firmware co-design on dual Cortex® -M33 MCUs. Its integrated architecture allows precise and efficient control of multi-phase GaN power supplies, making it particularly suited for high-performance server and telecom applications. This section provides a comprehensive overview of ACMC implementation on PPCA, highlighting its control strategy, hardware resources, architecture, firmware partitioning, and benefits over conventional methods.

3.1.1 Introduction to ACMC strategy

ACMC is a versatile control strategy commonly used in power converters, particularly in multi-phase and interleaved configurations. It consists of two control loops:

- **Outer voltage loop:** Regulates the output voltage by generating a current reference for the inner loops
- **Inner current loops:** Precisely controls the phase currents to reduce ripple, improve efficiency, and ensure equal current sharing

This dual-loop structure is scalable and works seamlessly across various power topologies, including interleaved buck or boost converters. ACMC is ideal for GaN technology due to its ability to deliver high bandwidth and low latency. By leveraging PPCA's dedicated hardware and advanced control capabilities, ACMC implementations achieve high precision and efficiency while meeting the stringent demands of modern power systems.

3.1.2 PPCA resources for ACMC strategy

The PPCA integrates specialized hardware modules that streamline the implementation of ACMC. The key resources include:

- **Dedicated hardware datapaths**
  - Up to six datapaths, including two for the voltage loop (outer loop) and four for interleaved current loops (inner loops)
  - Each datapath independently runs the control flow:
    - ADC acquisition (voltage or current sampling)
    - Measurement filtering (built-in ADC filter for signal noise reduction)
    - Regulation by the 3P3Z compensator to generate the duty cycle
    - PWM generation in one-shot mode to drive high/low-side switches
- **Event processing unit (EPU)**
  - Coordinates module interactions such as triggering ADC sampling via PWM and ensuring precise timing for inner loop current measurements and phases shifting
- **Synchronization for interleaved phases**
  - PWM synchronization ensures dynamic and precise phase shifting between the four interleaved inner-loop datapaths. This minimizes ripple and balances the load across phases
- **Shared memory and dual Cortex® -M33 CPUs**
  - The PPCA hosts two Cortex® -M33 CPUs with shared memory to efficiently partition firmware responsibilities. This architecture allows rapid data exchange between CPUs while maintaining coherence and meeting the real-time constraints of the control loop. As example, the MCU could be dedicated to a specific task ensuring the real time constraints

These resources allow the PPCA to handle the high-speed demands of ACMC with main CPU processing power available for application system-level tasks.

3.1.3 Hardware architecture of the ACMC using PPCA

The PPCA's hardware architecture is designed to offload the most computationally intensive tasks of ACMC, with each datapath optimized for real-time operation. The following describes the configuration and functionality of the PPCA for ACMC:

- **Outer voltage loop (Datapath 1)**
  - Voltage reference is generated by the PPCA CPU
  - ADC samples the input and output voltage, and the built-in gain, offset and filter (many filtering variants) conditions the sample signal
  - 3P3Z filters process the input voltage and the voltage error to generate the current reference
  - Current reference is sent to the PPCA CPU, which adapts it for dynamic phase shedding and sets it as the input for the inner loops' current reference
- **Inner current loops (Datapaths 2-5)**:
  - Each inner loop controls the current for one interleaved phase
  - The PWM module triggers the ADC average filter at precisely the right timing to measure the average phase currents. Then, the built-in gain, offset, and filter (many filtering variants) condition the signal
  - The ADC average value signals are processed by the 3P3Z compensator to regulate the duty cycle, which is sent back to the PWM module for controlling high/low-side switches
  - Phase synchronization is achieved by configuring PWM Sync modules as master to trigger the other PWMs and enforce precise phase shifts

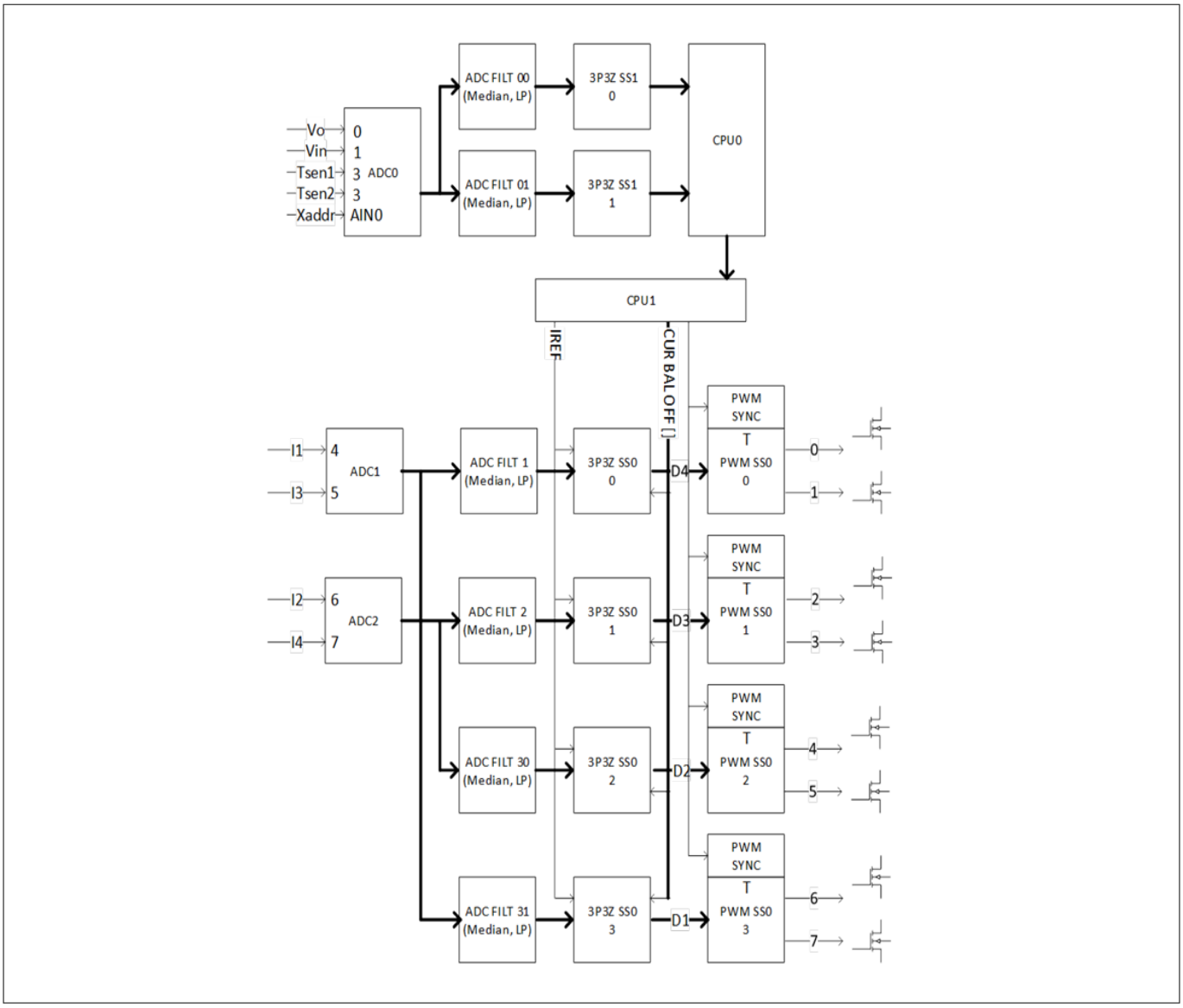


Figure 28 Inner current loops (Datapaths 2-5)

- **Configuration using Device Configurator**:
  - The ModusToolbox™ Device Configurator allows you to set up and configure ADC sampling rates, ADC filters compensator coefficients, PWM timing, and synchronization parameters

This architecture ensures tight integration between hardware modules and achieves a high degree of control precision and system synchronization.

3.1.4 Firmware co-design

While the PPCA hardware handles the real-time control tasks, the dual Cortex®-M33 CPUs complement the system with additional firmware-level functionalities. The firmware is partitioned effectively to balance the CPU load while meeting the real-time constraints of ACMC. Key elements of the firmware co-design include:

- **CPU-based voltage reference generation**
  - The CPU calculates the voltage reference for the outer loop and dynamically adjusts it based on operating conditions, such as load transients, soft start/stop (functional state machine) or input voltage variations (derating, …)

- **Dynamic phase shedding**
  - The CPU monitors system load and disables specific phases during light-load conditions to improve efficiency. When demand increases, the CPU re-enables phases seamlessly, ensuring smooth transitions without disrupting the control loop
- **Feed-forward control**
  - The CPU implements feed-forward algorithms to preemptively adjust duty cycles during large input voltage changes, enhancing transient response
- **Compensator coefficient adaptation**
  - Real-time computation and updating of 3P3Z compensator coefficients allow the system to maintain stability and performance during changing operating conditions
- **Reset and fault handling**
  - The CPU manages reset operations by clearing specific memory states (e.g., integrators) without losing parameters, enabling quick recovery after faults

By offloading timing-critical tasks to hardware and handling advanced functions in firmware, the PPCA achieves optimal performance while maintaining flexibility for system-level adaptations.

3.1.5 Benefits compared to legacy implementation

The PPCA-based ACMC implementation offers substantial advantages over traditional software-based or partially hardware-assisted solutions:

- **Superior control response and bandwidth**
  - Hardware-accelerated datapath enables response times as low as **120 ns**, significantly faster than traditional systems, ensuring rapid adaptation to transients
  - High-speed ADC sampling (25 Msps) and built-in gain, offset and filtering (many variants) improve signal accuracy. Compared to any software-based equivalent, these built-in capabilities (gain, offset and filtering) are a differentiation factor
- **Improved phase synchronization**
  - Real-time synchronization of PWM modules ensures precise phase shifts in interleaved designs, reducing output ripple and improving efficiency
- **Scalability and flexibility**
  - The modular architecture supports up to **four interleaved phases**, making it suitable for high-power, multiphase converters
  - Dynamic phase shedding enhances light-load efficiency without requiring additional hardware modifications
- **Enhanced stability and efficiency**:
  - Precise tuning of 3P3Z compensators improves gain and phase margins, ensuring robust operation across a broad range of conditions
- **Ease of development**:
  - The ModusToolbox™ Device Configurator simplifies setup, while the partitioned firmware approach reduces development complexity
  - Pre-integrated hardware modules eliminate the need for low-level coding, accelerating time-to-market

The combination of advanced hardware, efficient firmware co-design, and intuitive development tools makes PPCA a game-changing solution for ACMC in GaN power supplies.

3.2 Interleaved totem-pole PFC implementation using PPCA

PPCA enhances the implementation of power converters for high-performance server and telecom applications. The totem-pole PFC topology is particularly beneficial there, where high efficiency, high power

density, and low total harmonic distortion (THD) are crucial. This topology achieves all this. Additionally, it can operate at high switching frequencies, which can result in smaller passive components and a more compact design. This section provides an overview of its implementation, highlighting its control strategy, hardware resources, architecture, and firmware partitioning.

3.2.1 Introduction to interleaved totem-pole PFC

The totem-pole power factor correction (TPPFC) circuit is a bridgeless topology. Hence, there is no traditional diode bridge rectifier at the input. It mainly consists of a fast-switching leg (half-bridge) and a slow-switching leg (half-bridge). The slow leg operates with the mains frequency and acts as the return path. High-frequency modulation takes place in the fast leg and the PFC inductor is connected to the midpoint of the fast leg. Here, two-phase interleaved totem-pole PFC is introduced. Therefore, there are two fast legs with two PFC inductors. Figure 29 shows a simplified schematic of the totem-pole interleaved PFC using two phases.

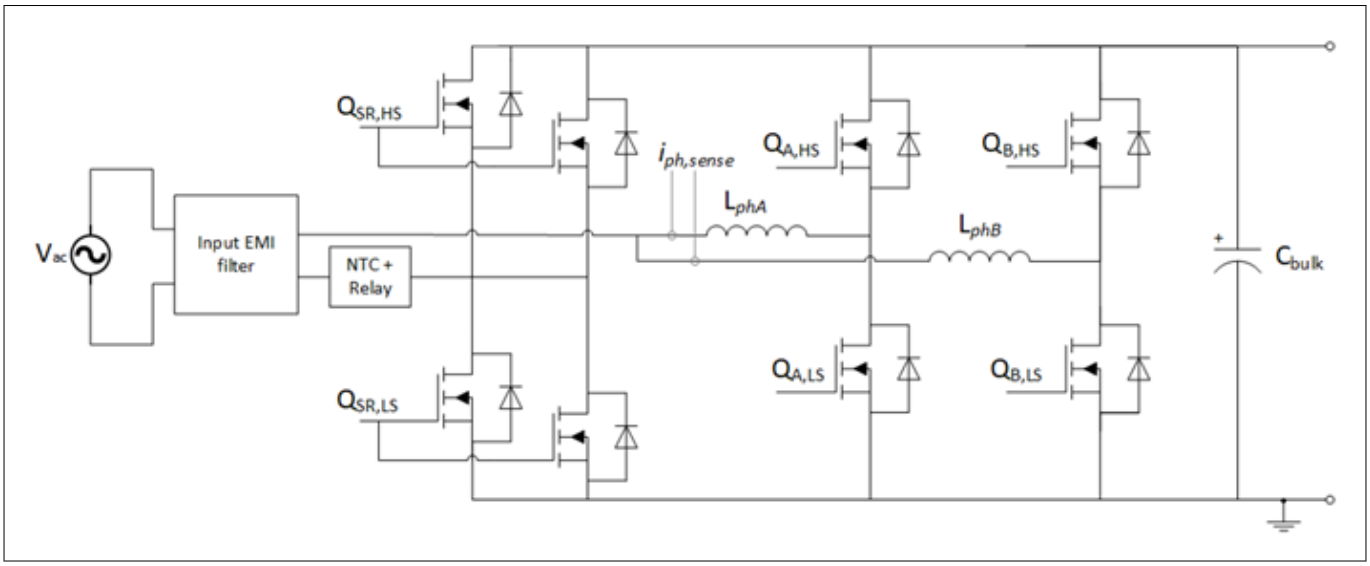


Figure 29 Simplified schematic of interleaved totem-pole PFC

3.2.2 PPCA resources for interleaved totem-pole PFC

PPCA offers enough resources to do the time critical tasks. The key resources include:

- **Shared memory and dual Cortex®-M33 CPUs**

  PPCA hosts two Cortex®-M33 CPUs with shared memory to efficiently partition firmware responsibilities. This architecture allows rapid data exchange between CPUs while maintaining coherence and meeting the real-time constraints of the control loop. Each MCU is dedicated to fast application specific tasks ensuring the real time constraints
- **TCPWM**

  High-resolution PWM generation to drive high-/low-side switches. Timer is used for scheduling tasks
- **ADC**

  Accurate and fast sampling of voltages and currents, and provides AC polarity detection for control and protection
- **Digital comparator**

  Fast AC voltage threshold detection and protection
- **Digital comparator with slope generation (DCSG):**

  Fast detection of overvoltage and overcurrent scenarios
- **Event processing unit (EPU):**

Direct module interactions, such as triggering ADC sampling via PWM or shutting down the PWM via the DCSG in fault scenarios

- **Hardware filter 3P3Z**
  Implementing the PID controller in hardware (optional) to free up CPU resources

3.2.3 Hardware architecture of the interleaved totem-pole PFC using PPCA

Time-critical and power application-specific tasks of the interleaved totem-pole PFC are handled within PPCA. The controller's core tasks ensure precise output voltage regulation using an average current mode control (ACMC) algorithm in continuous conduction mode (CCM) as shown in Figure 30.

3.2.3.1 Voltage loop dedicated to CPU from PPCA 0

- ADCs sample the input voltage (VIN), output voltage (VBUS), and inductor currents at fixed intervals, synchronized with MCU interrupts
- Measurements are filtered and corrected using the computed offset calibration data
- Outer voltage control loop regulates VBUS and generates a reference current
- The controller computes the VIN RMS value and modulates the reference current to optimize efficiency and power factor
- Based on load and line conditions, the phase shedding logic determines how many phases are active and adjusts the reference current
- Phase shift interleaving for optimal current ripple

3.2.3.2 Current loop dedicated to CPU from PPCA 1

- PI controller
- Feed-forward compensation
- Discontinuous conduction mode (DCM) compensation
- Update duty cycle of the HF PWMs (fast legs) to close the loop
- Disabling PWM outputs and re-initialization of regulators during AC zero-crossing windows
- After AC zero-crossing starting with soft start on fast legs
- Control of slow legs based on the AC polarity and choke current

3.2.3.3 Fast protections

Directly handle shut down (kill) of PWMs via the EPU to get an immediate reaction and enable cycle-by-cycle protections. The firmware follows afterwards to ensure safe operation.

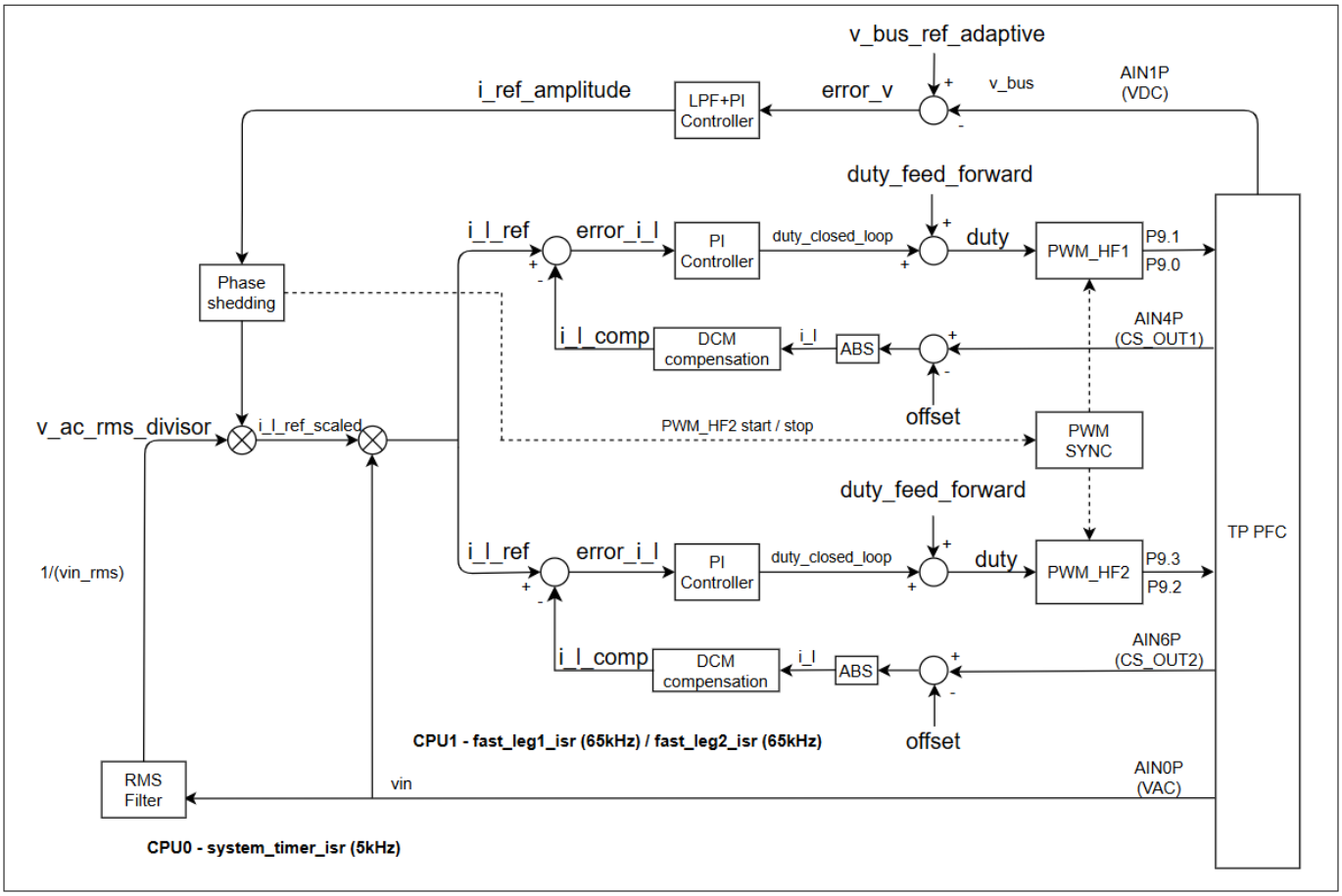


Figure 30 Interleaved totem-pole PFC control diagram

3.2.4 Firmware design

This controller design leverages the PSC3M/P8, PSC3M/P7 MCU's multicore architecture to efficiently manage power control tasks, where time-critical control loops run on dedicated cores, ensuring optimal performance, reliability, and response time for the power conversion.

Main Core (Cortex®-M33 at 180 MHz):

- User-interface
- External communication interface

PPCA Core 0 (Cortex®-M33 at 200 MHz):

- State machine management, protection, and fault recovery
- Voltage control loop processing
- PI control algorithm
- VIN RMS calculation and filtering
- Phase shedding

PPCA Core 1 (Cortex®-M33 at 200 MHz):

- Polarity detection
- Current control loop processing
- PI control algorithm
- Feed-forward compensation

- Discontinuous conduction mode (DCM) compensation
- PWM control

The three cores rely on the shared memory for inter-core communication and synchronization.

PPCA Core 0 runs a scheduler based on a dedicated hardware timer, whereas PPCA Core 1 runs in a continuous loop and a cycle-by-cycle interrupt for each phase. The table below shows a high-level overview of each core's activities.

Core	Activities
Main	• Read from shared memory • External communication • LED handling • FAN handling
PPCA 0	• Read and write from shared memory • VIN RMS calculation and filtering • Temperature handling • State machine handling • Voltage control loop • Phase shedding

PPCA 1	• Read and write from shared memory
	• Relay output setting
	• ADC EOC interrupt:
	  - Polarity detection
	  - Current control loop
	  - DCM compensation
	  - Feed-forward compensation
	  - PWM control

3.2.5 Benefits compared to legacy implementation

The PPCA-based implementation offers substantial advantages for high-performance power converters:

- **Superior control response and high bandwidth**
  - Enough CPU and hardware resources to increase the control bandwidth significantly
  - High-speed ADC sampling (25 Msps) and integrated filtering improve signal accuracy
- **Enhanced stability**
  - Precise tuning of 3P3Z compensators improves gain and phase margins, ensuring robust operation across a broad range of conditions
- **Phase synchronization**:
  - Synchronization of PWM modules ensure precise phase shifts in interleaved designs
  - Reduction of current ripple and reduction of input filter
- **Scalability and flexibility**
  - Modular architecture supports multiple phases
  - Dynamic phase shedding enhances light-load efficiency
  - Filter can be realized by software or hardware peripherals

- **Fast response**
  - DCSG enables response on hardware events
  - EPU to get a very fast reaction time
- **Ease of development**
  - The ModusToolbox™ Device Configurator simplifies the design, while the partitioned firmware approach reduces development complexity
  - Pre-integrated hardware modules eliminate the need for low-level coding, thus accelerating time-to-market

The combination of advanced hardware, efficient firmware co-design, and intuitive development tools makes PPCA a game-changing solution.

3.3 Half-bridge LLC implementation using PPCA

As already stated, PPCA enhances the implementation of power converters for high-performance server and telecom applications, and back-end DC-DC converters and specifically a half-bridge LLC topology with full-bridge rectification. This section provides an overview of its implementation, highlighting its control strategy, hardware resources, architecture, and firmware partitioning.

3.3.1 Introduction to half-bridge LLC

A half-bridge LLC (HB LLC) converter is a type of resonant converter widely used in power electronics for applications requiring high efficiency and compact design, such as in power supplies for telecommunications, computers, and renewable energy systems. This converter operates under the principle of resonant switching, which minimizes switching losses and electromagnetic interference (EMI), leading to improved performance. A notable key advantage is the safety isolation due to the usage of a transformer.

Specifically, in the half-bridge LLC architecture, the two primary-side switches are connected in series across the supply voltage, with the resonant tank circuit connected to the midpoint of the switches. This configuration allows for the generation of a high-frequency AC signal across the transformer primary, which is a key aspect of its operation.

The LLC resonant tank comprises a resonant and magnetizing inductance as well as a capacitor, or a half-bridge split capacitor in order to create a resonant circuit. This circuit enables zero-voltage switching (ZVS) and so further enhances the efficiency. As the switches toggle, the resonant behavior of the tank circuit helps to maintain smooth voltage and current waveforms, reducing stress on the components and improving overall reliability.

For the secondary-side, a full-bridge rectification (H-bridge) has been selected. Figure 31 shows a simplified schematic of the LLC HB DC-DC converter with split capacitor and an H-bridge at the output on the secondary side.

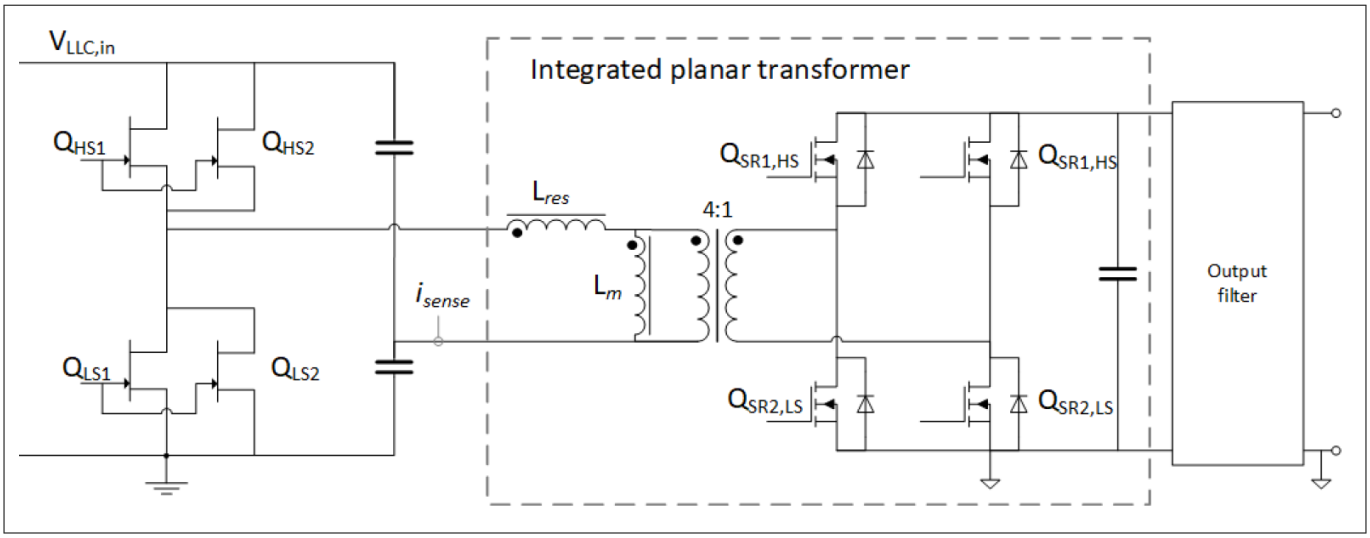


Figure 31 Simplified schematic of the LLC HB DC-DC converter

3.3.2 PPCA resources for half-bridge LLC

PPCA offers enough resources to do the time-critical tasks. The key resources include:

- **Shared memory and dual Cortex®-M33 CPUs**

  PPCA hosts two Cortex®-M33 CPUs with shared memory to efficiently partition firmware responsibilities. This architecture allows rapid data exchange between CPUs while maintaining coherence and meeting the real-time constraints of the control loop. Each core is dedicated to fast application specific tasks ensuring the real time constraints
- **TCPWM**

  High-resolution PWM generation to drive high-/low-side switches on the primary and secondary side. Timer used for scheduling tasks
- **ADC**

  Accurate and fast sampling of voltages and currents
- **Digital comparator with slope generation (DCSG)**:

  Fast detection of overvoltage and overcurrent scenarios
- **Event processing unit (EPU)**

  Direct module interactions, such as triggering ADC sampling via PWM, or shutting down the PWM via the DCSG in fault scenarios
- **Hardware filter 3P3Z**:

  Implementing the PID controller in hardware optional to free up CPU resources

3.3.3 Hardware architecture of the half-bridge LLC using PPCA

Time critical tasks are handled within the PPCA.

3.3.3.1 Voltage loop dedicated to CPU from PPCA

- Read data from shared memory
- Sense output voltage
- Calculate voltage error
- Calculate regulator output, either done by CPU or 3P3Z hardware filter

- Update PWMs
- Write data to shared memory

3.3.3.2 Dead-time and synchronous rectification control dedicated to CPU from PPCA 1

- Read data from shared memory
- Sense output current
- Calculate primary-side dead-time
- Calculate synchronous rectification timing and SR-on and SR-off control
- Write data to shared memory

3.3.3.3 Fast protections

Directly handle shutdown of PWMs via the EPU to get a very fast reaction time. Firmware follows afterward to maintain safe operation

3.3.4 Firmware design

This controller design leverages the PSC3M/P8, PSC3M/P7 MCU's multi-core architecture to efficiently manage power control tasks, where time-critical control loops run on dedicated cores, ensuring optimal performance, reliability, and response time for the power conversion.

Main Core (Cortex®-M33 at 180 MHz)

- Device initialization and configuration
- State machine management, protection, and fault recovery
- User-interface
- External communication interface

PPCA Core 0 (Cortex®-M33 at 200 MHz)

- Voltage control loop processing
- PID control algorithm with frequency clamping
- PWM signal update (both primary and SR PWMs)

PPCA Core 1 (Cortex®-M33 at 200 MHz)

- Current sensing
- Primary PWM dead-time calculation
- Synchronous rectification on/off timing calculation

The three cores rely on the shared memory for inter-core communication and synchronization.

Each core runs its own scheduler based on a dedicated hardware timer. Each scheduler manages and executes all its firmware tasks assigned to the processor.

The table below shows a high-level overview of each core's scheduler activities.

Core	Activities
Main	• Drive state machine • Write to shared memory

Core	Activities
PPCA 0	• Read from shared memory • Output voltage error calculation • Voltage loop control • Update primary-side and secondary-side PWMs • Write to shared memory
PPCA 1	• Read from shared memory • Output current sense with low-pass filter • Primary side PWM dead-time calculation • Synchronous rectification timing calculation • Write to shared memory

3.3.5 Benefits compared to legacy implementation

The PPCA-based implementation offers substantial advantages for high-performance power converters:

- **Superior control response and high bandwidth**
  - Enough CPU and hardware resources to increase the control bandwidth significantly
  - High-speed ADC sampling (25 Msps) and integrated filtering improve signal accuracy
- **Enhanced stability**
  - Precise tuning of 3P3Z compensators improves gain and phase margins, ensuring robust operation across a broad range of conditions
- **PWM synchronization**:
  - Synchronization of PWM modules ensures precise phase shifts for synchronous rectification
  - Interleaving of multiple DC-DC stages possible
- **Scalability and flexibility**
  - Modular architecture supports multiple phases
  - Filter can be realized by software or hardware peripherals
  - Easy switch between different kind of control methods
- **Fast response**
  - DCSG enables response on hardware events
  - EPU to get a very fast reaction time
- **Ease of development**
  - The ModusToolbox™ Device Configurator simplifies the design, while the partitioned firmware approach reduces development complexity
  - Pre-integrated hardware modules eliminate the need for low-level coding, accelerating time-to-market

The combination of advanced hardware, efficient firmware co-design, and intuitive development tools makes PPCA a game-changing peripheral for complete system solutions.

4 Summary

This application note explored the programmable power control accelerator (PPCA) available in PSC3M/P8, PSC3M/P7 MCU device, and ways to implement few power conversion solutions with that in easy and efficient ways. PPCA is a truly powerful peripheral with dedicated built-in CPU cores, configurable analog and digital functions, and memory and signaling mechanisms for implementing efficient power conversion solutions. In addition to that, the availability of resources like EPU, MOTIF, and CORDIC makes it a suitable candidate for the implementation of the motor control solutions.

References

[1] Infineon Technologies AG: *PSOC™ Control C3M/P8, C3M/P7 code examples*; Available online
[2] Infineon Technologies AG: *PSOC™ Control C3M/P8, C3M/P7-related documents*; Available online

Revision history

Document revision	Date	Description of changes
**	2025-10-13	Initial release
*A	2026-03-26	Added the initialization code snippets Added the description of the PPCA block Updated the images of Device Configurator in Section 2 Updated format in Table 1
*B	2026-09-11	Web release

Trademarks

The Bluetooth® word mark and logos are registered trademarks owned by Bluetooth SIG, Inc., and any use of such marks by Infineon is under license.

PSOC™, formerly known as PSoC™, is a trademark of Infineon Technologies. Any references to PSoC™ in this document or others shall be deemed to refer to PSOC™.

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2026-09-11

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2026 Infineon Technologies AG

All Rights Reserved.

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-lbh1755677730337

Important notice

Products which may also include samples and may be comprised of hardware or software or both ("Product(s)") are sold or provided and delivered by Infineon Technologies AG and its affiliates ("Infineon") subject to the terms and conditions of the frame supply contract or other written agreement(s) executed by a customer and Infineon or, in the absence of the foregoing, the applicable Sales Conditions of Infineon. General terms and conditions of a customer or deviations from applicable Sales Conditions of Infineon shall only be binding for Infineon if and to the extent Infineon has given its express written consent.

For the avoidance of doubt, Infineon disclaims all warranties of non-infringement of third-party rights and implied warranties such as warranties of fitness for a specific use/purpose or merchantability. Infineon shall not be responsible for any information with respect to samples, the application or customer's specific use of any Product or for any examples or typical values given in this document.

If this document is marked as 'Preliminary', 'Target', 'Draft' or 'Proposal', Infineon reserves the right to change all information given in this document at any time without notice. Subject to the development and release of a Product for series supply by Infineon, the technical specifications of the Product are set forth in the relevant datasheet provided by Infineon without such marking.

The data contained in this document is exclusively intended for technically qualified and skilled customer representatives. It is the responsibility of the customer to evaluate the suitability of the Product for the intended application and the customer's specific use and to verify all relevant technical data contained in this document in the intended application and the customer's specific use. The customer is responsible for properly designing, programming, and testing the functionality and safety of the intended application, as well as complying with any legal requirements related to its use.

Unless otherwise explicitly approved by Infineon, Products may not be used in any application where a failure of the Products or any consequences of the use thereof can reasonably be expected to result in personal injury.

However, the foregoing shall not prevent the customer from using any Product in such fields of use that Infineon has explicitly designed and sold it for, provided that the overall responsibility for the application lies with the customer. Infineon expressly reserves the right to use its content for commercial text and data mining (TDM) according to applicable laws, e.g. Section 44b of the German Copyright Act (UrhG). If the Product includes security features:

Because no computing device can be absolutely secure, and despite security measures implemented in the Product, Infineon does not guarantee that the Product will be free from intrusion, data theft or loss, or other breaches ("Security Breaches"), and Infineon shall have no liability arising out of any Security Breaches.

If this document includes or references software:

The software is owned by Infineon under the intellectual property laws and treaties of the United States, Germany, and other countries worldwide. All rights reserved. Therefore, you may use the software only as provided in the software license agreement accompanying the software.

If no software license agreement applies, Infineon hereby grants you a personal, non-exclusive, non-transferable license (without the right to sublicense) under its intellectual property rights in the software (a) for software provided in source code form, to modify and reproduce the software solely for use with Infineon hardware products, only internally within your organization, and (b) to distribute the software in binary code form externally to end users, solely for use on Infineon hardware products. Any other use, reproduction, modification, translation, or compilation of the software is prohibited.

For further information on the Product, technology, delivery terms and conditions, and prices, please contact your nearest Infineon office or visit www.infineon.com.