Migration guide: For PSOC™ Control C3 to C1 MCU family

About this document

Scope and purpose

This document provides a structured approach for transitioning applications from the PSOC™ Control C3 (PSC3M3xD series) MCU to the PSOC™ Control C1 PSC1M3x MCU. It focuses on identifying architectural differences, peripheral mappings, and software adaptation requirements to enable efficient portability of existing designs. The document outlines key considerations, such as clock configuration, GPIO behavior, communication interfaces, and development toolchain differences, to minimize migration effort.

The purpose of this document is to help engineers reuse as much of their existing firmware and hardware design as possible while ensuring functional equivalence and performance consistency on the PSC1Mx platform. It also highlights potential incompatibilities and provides recommendations, best practices, and example strategies to accelerate the migration process and reduce development risk.

Intended audience

This document is intended for embedded system engineers, firmware developers, and hardware designers who are planning to migrate applications from the PSOC™ Control C3 (PSC3M3xD series) MCU to the PSC1Mx MCU. It is particularly useful for professionals with prior experience in developing applications on the PSOC™ Control C3 MCU platform and seeking guidance on adapting their designs to the PSOC™ Control C1 MCU family.

The document is also beneficial for system architects and technical leads who are responsible for evaluating feasibility, planning migration strategies, and ensuring continuity of functionality and performance during the transition. A basic understanding of microcontroller architectures, peripheral configuration, and embedded software development is assumed to effectively utilize this document.

Table of contents

About this document . . . 1

Table of contents . . . 2

1 Introduction . . . 4

2 Device overview comparison . . . 5

2.1 Core architecture . . . 5

2.2 Memory architecture . . . 5

2.3 Peripheral feature matrix . . . 6

3 Development ecosystem . . . 7

3.1 Toolchain overview . . . 7

3.2 PSOC™ Control C3 MCU development libraries (source) . . . 7

3.3 PSOC™ Control C1 MCU development setup in ModusToolbox™ . . . 7

3.4 Debug interface comparison . . . 8

3.5 Security feature migration . . . 8

4 Digital peripheral migration . . . 9

4.1 General-purpose I/O (GPIO) . . . 9

4.2 Universal asynchronous receiver transmitter (UART) . . . 10

4.3 Serial peripheral interface (SPI) . . . 11

4.4 Inter-integrated circuit (I2C) . . . 12

4.5 DMA comparison . . . 13

5 Analog peripheral migration . . . 15

5.1 Analog-to-Digital converter (ADC) . . . 15

5.2 Analog comparators (ACMP) . . . 17

6 Timer and PWM migration . . . 18

6.1 Timer/PWM module comparison . . . 18

7 Communication peripherals . . . 20

7.1 CAN . . . 20

8 Interrupt system . . . 22

9 Clock system . . . 23

10 Power management . . . 25

11 PCB hardware design considerations . . . 26

11.1 KIT_PSC1M3_EVAL hardware overview . . . 26

11.2 Pin count reduction guidance . . . 26

12 Boot and programming migration . . . 27

12.1 PSOC™ Control C3 MCU boot process . . . 27

12.2 PSOC™ Control C1 MCU Boot Mode Index (BMI) . . . 27

12.3 Programming interface mapping . . . 27

Table of contents

13 API reference mapping 29
13.1 Complete API migration table 29

14 Migration checklist 31
14.1 Pre-migration hardware review 31
14.2 ModusToolbox™ project setup 31
14.3 Firmware code migration 31
14.4 Verification and testing 32

15 Summary 33

Revision history 35

Trademarks 36

Disclaimer 37

1 Introduction

This document provides a comprehensive, step-by-step migration guide for developers transitioning from the Infineon PSOC™ Control C3 PSC3M3xD sub-family to the PSOC™ Control C1 PSC1M3x MCU sub-family. The PSC3M3xD (Cortex®-M33 at 100 MHz, 256 KB Flash, 64 KB SRAM, 12-bit/6-Msps SAR ADC, 2× CAN FD (one node up to 8 Mbps), TrustZone® + CryptoLite) and the PSC1M3x (Cortex®-M0 at 48 MHz, up to 200 KB Flash, 16 KB SRAM, 12-bit/1.1 MS/s VADC, MultiCAN+ Classical) share the ModusToolbox™ ecosystem but serve different application domains with distinct architectural trade-offs.

The PSOC™ Control C3 PSC3M3xD MCU is a high-performance Arm® Cortex®-M33-based MCU running at up to 100 MHz, featuring hardware TrustZone®, FPU, DSP extensions, MPU, 16 KB I-Cache, CORDIC, and a rich analog subsystem (HPPASS with a 12-bit/6-Msps SAR ADC, 5× CSG comparators with 10-bit DAC, and slope generators) targeting advanced motor control, power conversion, and industrial applications. PSA L2 certified; Class B/SIL 2 safety test libraries available; supply 1.71 V to 3.6 V; Ta -40°C to 115°C; packages VQFN-48, E-LQFP-48, VQFN-64, and E-LQFP-64.

The PSOC™ Control C1 PSC1M3x MCU is a 32-bit Arm® Cortex®-M0-based industrial microcontroller running at up to 48 MHz (0.84 DMIPS/MHz) with up to 200 KB Flash (ECC) and 16 KB SRAM (parity), a 24-bit CORDIC + 32-bit divide MATH coprocessor, four USIC channels (UART/SPI/IIC/IIS/LIN up to 12 Mb/s), MultiCAN+ Classical (2 nodes/32 message objects, up to 1 MBaud), a 12-bit ADC with up to 12 analog inputs, up to four fast analog comparators, CCU4/CCU8 96 MHz timers, BCCU for LED lighting, supply 1.8 V to 5.5 V, in TSSOP-38, VQFN-40, VQFN-48, or LQFP-48 packages.

2 Device overview comparison

2.1 Core architecture

Table 1 Core architecture

Feature	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU
CPU Core	Arm® Cortex®-M33	Arm® Cortex®-M0
Architecture	ARMv8-M	ARMv6-M
Core Speed	Up to 100 MHz	Up to 48 MHz
FPU	Yes (Single Precision)	No
DSP Instructions	Yes (SIMD, MAC)	No
TrustZone® (SAU)	Yes	No
MPU	Yes	No
Instruction Cache	Yes (16 KB I-Cache)	No
Debug Interface	SWD/JTAG	SWD/SPD
IPC Channels	4× IPC	Not available
DMA	2× DMA (16 ch each)	Not available (no DMA controller)
Math Coprocessor	CORDIC + DSP	MATH (24-bit CORDIC + 32-bit DIV)

Critical Consideration: *The PSOC™ Control C3 PSC3M3xD MCU uses a Cortex®-M33 at up to 100 MHz with FPU, DSP extensions, TrustZone®, MPU, 16 KB I-Cache, CORDIC and two 16-channel DMA controllers. None of these features are available on the PSOC™ Control C1 PSC1M3x MCU Cortex®-M0 (48 MHz, MATH coprocessor with 24-bit CORDIC and 32-bit divide). All floating-point, DSP, security, and memory protection logic must be completely refactored before migration can begin.*

2.2 Memory architecture

Table 2 Memory architecture

Memory Type	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU
Flash	Up to 256 KB	Up to 200 KB
SRAM	Up to 64 KB	16 KB
EEPROM	Emulated via Flash	Emulated via Flash
ROM (Bootloader)	Yes	Yes
Backup Domain RAM	Yes (Backup Reset, RTC)	Not available

⚠ ***Memory Warning:*** *PSC3M3xD offers up to 256 KB read-while-write Flash with ECC and 64 KB SRAM with ECC (plus 64 KB ROM and 32 KB SFlash), while PSC1M3x provides a maximum of 200 KB Flash with ECC, 16 KB SRAM with parity and 8 KB ROM. Significant code and data size optimization is mandatory before migration. Evaluate middleware stacks, lookup tables and DMA buffers carefully.*

2.3 Peripheral feature matrix

Table 3 Peripheral feature matrix

Peripheral	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU
GPIO	Up to 39 GPIOs + 22 Smart I/Os	Up to 36 (VQFN-40) / 44 (VQFN-48/ LQFP-48)
UART/LIN	4× SCB	4× (USIC)
SPI	5× SCB	4× (USIC)
I2C	4× SCB	4× (USIC)
CAN	2× CAN-FD	MultiCAN+ (Classical only)
ADC	6-Msps SAR (HPPASS)	VADC 1.1 MS/s (PSC1M3x)
Analog Comparators	5× CSG (HPPASS)	4× ACMP
PWM (Complementary)	TCPWM	CCU8 (8 slices)
Timers	TCPWM 32-bit/16-bit	CCU4 (16) + CCU8 (16)
POSIF (Position Sensing)	1× MOTIF	1× POSIF
WDT	Yes	Yes
RTC	Yes	Yes
DMA	2× 16-channel	Not available
Security	TrustZone® + CryptoLite	CRC engine

3 Development ecosystem

3.1 Toolchain overview

Table 4 Toolchain overview

Feature	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU
Primary IDE	ModusToolbox™	ModusToolbox™
Compiler	GCC/Arm® Compiler 6/IAR	GCC/Arm® Compiler 6/IAR
Debug Probe	KitProg3/MiniProg4	J-Link/DAP-miniWiggler
Debug Protocol	SWD/JTAG	SWD/SPD
Peripheral Config Tool	Device Configurator	Device Configurator
Driver Library	PDL (`Cy_`) + HAL (`cyhal_`)	PSC1Mx Library (mtb-dsl-psc1m3) (`Cy_`)
BSP	mtb-hal-psc3, mtb-pdl-cat1	mtb-dsl-psc1m3 (PSC1Mx Library)
Security Provisioning	Yes (CySecureTools, TrustZone®)	Not applicable
RTOS Support	FreeRTOS	FreeRTOS
Motor Control Suite	ModusToolbox™ Motor Suite	Available soon

Tool Continuity: *Both PSOC™ Control C3 MCU and the PSOC™ Control C1 MCU family share the ModusToolbox™ Eclipse IDE environment. This reduces migration friction in terms of project creation, Library Manager, and Device Configurator workflows. Only the BSP and driver library APIs differ.*

3.2 PSOC™ Control C3 MCU development libraries (source)

The PSOC™ Control C3 MCU ecosystem in ModusToolbox™ uses the following key repositories:

- CMSIS Repository
- Core Library
- Core GNU make Build System
- Device Database (device-db)
- Hardware Abstraction Layer
- MTB CAT1 Peripheral Driver Library

3.3 PSOC™ Control C1 MCU development setup in ModusToolbox™

Steps to create a new PSOC™ Control C1 MCU project in ModusToolbox™:

1. Open ModusToolbox™ Eclipse IDE
2. Go to File → New → ModusToolbox™ Application
3. Select BSP: `TARGET_KIT_PSC1M3_EVK` (VQFN-48, PSC1M3 device PSC1M3FBLGQ0AT)
4. Choose a starter template (Empty App or FreeRTOS App)
5. Configure MCLK = 48 MHz and PCLK = 96 MHz via `Cy_SCU_CLOCK_Init()`
6. Recreate peripheral configurations using Device Configurator

PSC1Mx Library (mtb-dsl-psc1m3) Repository (MTB)

3.4 Debug interface comparison

Both PSOC™ Control C3 MCU and PSOC™ Control C1 MCU expose an SWD-class debug port routed to standard 10-pin Cortex® headers on the corresponding evaluation kits. The most important migration item is the secondary protocol: PSOC™ Control C3 MCU supports 4-wire JTAG, whereas PSOC™ Control C1 MCU supports the single-pin SPD protocol. Tooling differences are summarized below.

Table 5 Debug Interface Comparison

Feature	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU family
Primary Protocol	SWD	SWD
Secondary Protocol	JTAG	SPD (Single Pin Debug)
Virtual COM Port	Via KitProg3	Via on-board XMC4200 debugger
On-board Debug MCU	KitProg3 (custom)	XMC4200 Cortex®-M4F

3.5 Security feature migration

PSOC™ Control C3 MCU includes comprehensive security features:

- TrustZone®-M (SAU) — Secure/Non-secure world partitioning
- CryptoLite — TRNG, VU, SHA-256, AES-128 hardware accelerator
- Secure boot with key provisioning via CySecureTools
- Device Firmware Update (DFU) middleware supporting UART, I2C, SPI

The PSOC™ Control C1 MCU provides only a CRC engine. All TrustZone®-based secure boot, AES-128/SHA-256, and TRNG functionality must be removed or replaced with software implementations when migrating.

⚠ ***Security Warning:*** *If your application relies on PSOC™ Control C3 MCU's TrustZone®, CryptoLite (AES-128, SHA-256, TRNG), these cannot be migrated to PSOC™ Control C1 MCU.*

4 Digital peripheral migration

4.1 General-purpose I/O (GPIO)

Table 6 General-Purpose I/O (GPIO)

Feature	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU
GPIO Count	Up to 39 GPIOs + 22 Smart I/Os	Up to 43 GPIOs
Analog-capable Pins	Up to 18	Up to 14 analog inputs
Smart I/O/Programmable Logic	Yes (28 Smart I/Os, 56 LUTs)	No — Fixed pin mux
HSIOM	High-Speed I/O Matrix	Fixed multiplexing
Drive Modes	Strong, Open-Drain, Hi-Z, Pull-Up/Down	Push-Pull, Open-Drain, Hi-Z
Hardware Override Paths	Via Smart I/O	HW0/HW1 override
Voltage Levels	1.71 V – 3.6 V	1.8 V – 5.5 V
GPIO Interrupt	Port/Pin level (direct)	ERU-based

⚠ ***Key Consideration:*** *PSOC™ Control C3 MCU uses the HSIOM for flexible pin routing and Smart I/O for programmable logic at the pad level. PSOC™ Control C1 MCU uses fixed pin multiplexing. All I/O assignments must be reviewed against the PSOC™ Control C1 MCU pin assignment tables in the datasheet before PCB layout is finalized.*

PSOC™ Control C3 MCU — PDL GPIO Code:

```
/* PSOC™ Control C3 - GPIO Output Configuration using PDL */
cy_stc_gpio_pin_config_t pinConfig = {
    .outVal    = 0UL,
    .driveMode = CY_GPIO_DM_STRONG,
    .hsiom     = P0_0_GPIO,
    .intEdge   = CY_GPIO_INTR_DISABLE,
    .vtrip     = CY_GPIO_VTRIP_CMOS,
    .slewRate  = CY_GPIO_SLEW_FAST,
};
Cy_GPIO_Pin_Init(GPIO_PRT0, 0U, &pinConfig);
Cy_GPIO_Inv(GPIO_PRT0, 0U);  /* Toggle */
```

PSOC™ Control C1 MCU — PSC1Mx Library (mtb-dsl-psc1m3) GPIO Code:

```
/* PSOC™ Control C1 - GPIO Output Configuration using PSC1Mx Library (mtb-dsl-psc1m3) */
const Cy_GPIO_CONFIG_t gpio_config = {
    .mode            = CY_GPIO_MODE_OUTPUT_PUSH_PULL,
    .output_level    = CY_GPIO_OUTPUT_LEVEL_LOW,
    .output_strength = /* no output_strength param on PSC1Mx */
};
Cy_GPIO_Init(PORT1, 0U, &gpio_config);
Cy_GPIO_ToggleOutput(PORT1, 0U);  /* Toggle */
```

PSOC™ Control C1 MCU — GPIO Toggle via SysTick ISR:

ERU-Based GPIO Interrupt on PSOC™ Control C1 MCU:

```
/* PSOC™ Control C1 - External Request Unit (ERU) for GPIO Interrupt */
/* ERU replaces  C3 direct pin-level interrupt mechanism */
const Cy_ERU_ETL_CONFIG_t eru_etl_config = {
    .input_a            = CY_ERU_ETL_INPUT_A0(ERU0_ETL0_INPUTA_P2_1),
    .source             = CY_ERU_ETL_SOURCE_A,
    .edge_detection     = CY_ERU_ETL_EDGE_DETECTION_RISING,
    .status_flag_mode   = CY_ERU_ETL_STATUS_FLAG_MODE_HWCTRL,
    .enable_output_trigger = true,
    .output_trigger_channel = CY_ERU_ETL_OUTPUT_TRIGGER_CHANNEL0
};
Cy_ERU_ETL_Init(ERU0_ETL0, &eru_etl_config);
```

4.2 Universal asynchronous receiver transmitter (UART)

Table 7 Universal asynchronous receiver transmitter

Feature	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU family
Module Name	SCB (Serial Communication Block)	USIC (Universal Serial Interface Channel)
Instances	5× SCB + 1× DeepSleep SCB	2 USIC Modules/4 Channels total
UART/LIN Count	Up to 4× UART/LIN	Up to 4× UART channels
DMA Support	Yes	Yes
IrDA	Yes	Yes
LIN	Yes	Yes
ISO 7816	Yes	No
DeepSleep Slave	Yes (1× SCB)	No
FIFO	Yes	Yes (up to 16 entries)
Hardware Flow Control	Yes (RTS/CTS)	Yes

⚠ ***Errata*** *— USIC ASC Initialization (USIC_AI.D001): There is a known incorrect initialization example for the USIC module in ASC (UART) mode documented in USIC_AI.D001, affecting all PSOC™ Control C1 MCU devices. Apply the corrected port pin configuration:*

```
/*  Control C1 - Corrected ASC Mode Port Pin Configuration
 * Per Errata USIC_AI.D001
 * Configure P0.7 as ALT7 output function for DOUT0 */
PORT0->IOCR4 |= (0x27U << 26U);
```

PSOC™ Control C3 MCU — PDL UART Code:

```
/* PSOC™ Control C3 - UART Init and Transmit */
Cy_SCB_UART_Init(UART_HW, &UART_config, &UART_context);
Cy_SCB_UART_Enable(UART_HW);
Cy_SCB_UART_PutString(UART_HW, "Hello PSOC™ Control C3\r\n");
```

PSOC™ Control C1 MCU — PSC1Mx Library (mtb-dsl-psc1m3) UART Code:

```
/* PSOC™ Control C1 - USIC UART Channel Init and Transmit */
const Cy_UART_CH_CONFIG_t uart_config = {
    .baudrate     = 115200U,
    .stop_bits    = 1U,
    .data_bits    = 8U,
    .frame_length = 8U,
    .oversampling = 16U,
    .parity_mode  = CY_USIC_CH_PARITY_MODE_NONE
};

Cy_UART_CH_Init(CY_UART0_CH0, &uart_config);
Cy_UART_CH_SetInputSource(CY_UART0_CH0,
                          CY_UART_CH_INPUT_RXD,
                          USIC0_C0_DX0_P1_4);
Cy_UART_CH_Start(CY_UART0_CH0);

/* Transmit byte */
Cy_UART_CH_Transmit(CY_UART0_CH0, (uint16_t)'H');
```

***Note**: The PSOC™ Control C3 MCU SCB supports ISO 7816 smart card interface and a dedicated DeepSleep SCB slave. Neither feature is available in the PSOC™ Control C1 MCU USIC. Applications using these modes must be completely redesigned.*

4.3 Serial peripheral interface (SPI)

Table 8 Serial peripheral interface (SPI)

Feature	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU
Module Name	SCB-SPI	USIC-SPI (SSC)
Dedicated Fast SPI	No (no fast SPI SCB on PSC3M3xD)	No dedicated fast SPI
Modes	Master/Slave	Master/Slave
FIFO Depth	Yes (configurable)	16/32 entries
DMA	Yes	Yes
SSC Delay Compensation	N/A	Not usable — errata USIC_AI.008
SSC Slave Clock Phase	Configurable	Limited — errata USIC_AI.017
MSLS Bit Behavior	N/A	Clearing PSR.MSLS de-asserts SELOx — errata USIC_AI.018

⚠ ***Errata*** *— SPI (SSC) Known Issues on PSOC™ Control C1 MCU:*
- USIC_AI.008: SSC delay compensation feature cannot be used
- USIC_AI.014: No serial transfer possible while running capture mode timer simultaneously
- USIC_AI.017: Clock phase of data shift in SSC slave cannot be changed
- USIC_AI.018: Clearing PSR.MSLS bit immediately desserts the SELOx output signal
Review all USIC errata items before finalizing SPI implementation

PSOC™ Control C3 MCU — PDL SPI Code:

```
/* PSOC™ Control C3 - SPI Master Init and Write */
Cy_SCB_SPI_Init(SPI_HW, &SPI_config, &SPI_context);
Cy_SCB_SPI_Enable(SPI_HW);
Cy_SCB_SPI_Write(SPI_HW, txData);
while (Cy_SCB_SPI_IsBusBusy(SPI_HW));
```

PSOC™ Control C1 MCU — PSC1Mx Library (mtb-dsl-psc1m3) SPI Code:

```
/* PSOC™ Control C1 - USIC SPI Master Init and Transmit */
const Cy_SPI_CH_CONFIG_t spi_config = {
    .baudrate       = 1000000U,
    .bus_mode       = CY_SPI_CH_BUS_MODE_MASTER,
    .selo_inversion = CY_SPI_CH_SLAVE_SEL_INV_TO_MSLS,
    .parity_mode    = CY_USIC_CH_PARITY_MODE_NONE
};

Cy_SPI_CH_Init(CY_UART0_CH0, &spi_config);
Cy_SPI_CH_Start(CY_UART0_CH0);

Cy_SPI_CH_Transmit(CY_UART0_CH0, txData,
                   CY_SPI_CH_MODE_STANDARD);
while (Cy_SPI_CH_GetStatusFlag(CY_UART0_CH0,
       CY_SPI_CH_STATUS_FLAG_TRANSMIT_SHIFT_INDICATION) == 0U);
```

4.4 Inter-integrated circuit (I2C)

Table 9 Inter-integrated circuit (I2C)

Feature	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU
Module Name	SCB-I2C	USIC-I2C
Modes	Master/Slave/Master-Slave	Master/Slave
Speed	Standard (100K) / Fast (400K) / Fast+ (1 Mbps)	Standard (100K) / Fast (400K)
DeepSleep I2C Slave	Yes (1× SCB)	No
SMBus	Yes	Limited
FIFO	Yes	Yes
DMA	Yes	Yes

⚠ ***Errata*** *— I2C Slave Lock-up Recovery: A known issue exists where the I2C slave can lock up on PSOC™ Control C1 MCU. Apply the following recovery sequence:*

```
/*  Control C1 - I2C Slave Recovery Sequence
 * Per PSC1M3 (XMC1400-AA-Step compatible) Errata
 * Apply when IIC slave becomes locked: */

/* Step 1: Detect lock condition at application level */

/* Step 2: Clear all pending I2C interrupts */
NVIC_ClearPendingIRQ(USIC0_0_IRQn);

/* Step 3: Invalidate internal TBUF – Write 10B to FMR.MTDV */
CY_I2C0_CH0->FMR |= (0x2U << USIC_CH_FMR_MTDV_Pos);

/* Step 4: Clear all I2C status flags */
Cy_I2C_CH_ClearStatusFlag(CY_I2C0_CH0, 0xFFFFFFFFU);

/* Step 5: Reprogram SCL and SDA port pin functions */
PORT2->IOCR0 |= (0x18U << 8U);    /* SDA pin */
PORT2->IOCR0 |= (0x18U << 24U);   /* SCL pin */
/* IIC slave is now ready to communicate with master */
```

PSOC™ Control C3 MCU — PDL I2C Code:

```
/* PSOC™ Control C3 - I2C Master Write */
cy_stc_scb_i2c_master_xfer_config_t masterTransfer = {
    .slaveAddress = 0x48U,
    .buffer       = txBuffer,
    .bufferSize   = sizeof(txBuffer),
    .xferPending  = false
};
Cy_SCB_I2C_MasterWrite(I2C_HW, &masterTransfer, &I2C_context);
```

PSOC™ Control C1 MCU — PSC1Mx Library (mtb-dsl-psc1m3) I2C Code:

```
/* PSOC™ Control C1 - I2C Master Write */
Cy_I2C_CH_MasterStart(CY_I2C0_CH0,
                      0x48U,
                      CY_I2C_CH_CMD_WRITE);
Cy_I2C_CH_MasterTransmit(CY_I2C0_CH0, dataByte);
Cy_I2C_CH_MasterStop(CY_I2C0_CH0);
```

4.5 DMA comparison

Table 10 DMA Comparison

Feature	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU

(table continues...)

Table 10 (continued) DMA Comparison

DMA Controllers	2× DMA	Not available
Channels per Controller	16	—
Total Channels	32	0 (no DMA)
Descriptor Types	1D, 2D scatter-gather	—
Triggered By	TCPWM, SCB, HPPASS	Use interrupt-driven I/O

⚠ ***DMA Channel Reduction:*** *PSC3M3xD provides 32 DMA channels (2× DMA controllers, 16 channels each). The PSC1M3x has no DMA controller at all, so every PSC3M3xD DMA transfer must be re-architected as an interrupt-driven (NVIC) or polled transfer. Plan CPU bandwidth carefully — high-rate ADC, USIC, and CCU traffic that ran 'for free' on the PSC3M3xD DMA will land on the Cortex®-M0 core after migration.*

5 Analog peripheral migration

5.1 Analog-to-Digital converter (ADC)

The ADC subsystem represents the most significant architectural difference between the two devices.

Table 11 Analog-to-Digital converter (ADC)

Feature	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU
Module Name	HPPASS — SAR ADC	VADC (Versatile ADC)
Resolution	12-bit	12-bit
Sampling Speed	6 Msps	Up to 1.1 MS/s (PSC1M3x)
ADC Kernels	1 SAR with 12+4 samplers	2 independent kernels
Analog Inputs	Up to 18 (AN_A and AN_B)	Up to 12 (VQFN-40) / 14 (VQFN-48)
Pseudo-Differential	Yes	Limited
Slope Generator (CSG)	Yes (10-bit DAC, 30 Msps)	No
Result FIFO	Yes (32 entries)	No dedicated FIFO
Result Registers	28 result registers	Per-kernel result registers
Startup Calibration	Yes	Configurable
DMA Support	Yes	Yes
ADC Triggering	TCPWM trigger/HPPASS FSM	CCU4/CCU8 trigger, software
Scan Request Source	Yes (queue, background)	Yes (queue, scan, injection)

PSOC™ Control C3 MCU HPPASS ADC Architecture:

The PSOC™ Control C3 PSC3M3xD MCU HPPASS includes:

- 12-bit SAR ADC operating at 6 Msps (PSC3M3xD)
- Up to 16 sample-and-hold (S/H) circuits with parallel idle sampling, connected to up to 16 dedicated analog pads (AN_A0–A7, AN_B0–B7)
- Two additional GPIOs usable as analog inputs; configurable input gain of 1, 3, 6, 12 on every S/H stage
- CSG_DAC_OUT accessible via MUX0 select 1
- Temperature sensor accessible via MUX select 3
- Digital comparator on the ADC output for boundary checking and low-latency triggering of TCPWM
- 5× Comparator-Slope Generator (CSG), each with a built-in 10-bit DAC and slope generator (<10 ns response)

PSOC™ Control C3 MCU — PDL ADC Code:

```
/* PSOC™ Control C3 - SAR ADC Result Read via HPPASS */
uint16_t result;
Cy_SAR2_Channel_GetResult(SAR2_HW, channelNum, &result);
```

PSOC™ Control C1 MCU — PSC1Mx Library (mtb-dsl-psc1m3) VADC Code:

```
/* PSOC™ Control C1 - VADC Global and Group Initialization */
const Cy_VADC_GLOBAL_CONFIG_t vadc_global_config = {
    .boundary0               = 0U,
    .boundary1               = 0U,
    .module_disable          = 0U,
    .msb_conversion_disable  = 0U,
    .arbiter_mode            = CY_VADC_ARBMODE_ALWAYS,
    .clock_config = {
        .analog_clock_divider  = 3U,
        .msb_conversion_clock  = 0U,
        .arbiter_clock_divider = 1U
    }
};

const Cy_VADC_GROUP_CONFIG_t vadc_group_config = {
    .emux_config = {
        .stce_usage  = 0U,
        .emux_mode   = CY_VADC_GROUP_EMUXMODE_SWCTRL,
        .emux_coding = CY_VADC_GROUP_EMUXCODE_BINARY,
    },
    .class0 = {
        .sample_time_std_conv     = 3U,
        .conversion_mode_standard = CY_VADC_CONVMODE_12BIT,
    }
};

Cy_VADC_GLOBAL_Init(VADC, &vadc_global_config);
Cy_VADC_GROUP_Init(VADC_G0, &vadc_group_config);
Cy_VADC_GROUP_QueueInsertChannel(VADC_G0, &channel_config);

/* Read ADC result */
uint16_t result = (uint16_t)Cy_VADC_GROUP_GetDetailedResult(
                            VADC_G0, resultRegNum);
```

⚠ ***Errata*** *— ADC Trigger Bus (ADC_CM.D001): The definition of trigger bus bits in register OCS is incorrect in earlier documentation. Verify the correct trigger bus bit definitions for VADC in the latest errata sheet before configuring ADC triggers.*

⚠ ***Critical ADC Migration Note:*** *The PSC3M3xD HPPASS provides a 12-bit/6-Msps SAR ADC with up to 16 parallel sample-and-hold stages, integrated slope generator and digital comparator. The PSC1M3x VADC provides a single 12-bit ADC up to 1.1 MS/s with two sample-and-hold stages, an adjustable input gain and a 0–5.5 V input range. Applications requiring high-speed multi-channel synchronous sampling (e.g. three-shunt motor control at high PWM frequencies) must be carefully redesigned. Refer to Section 5.1 above for the PSC1M3x-specific VADC configuration when implementing PMSM FOC three-shunt sensing.*

5.2 Analog comparators (ACMP)

Table 12 Analog comparators (ACMP)

Feature	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU
Module Name	HPPASS CSG (Comparator-Slope Generator)	ACMP (Analog Comparator)
Comparator Count	5× CSG comparators	4× ACMP
Slope Generator (DAC)	Yes (10-bit DAC, 30 Msps)	No
Hysteresis	Yes	Yes
Interrupt	Yes via HPPASS FSM	Yes via ERU
Output to PWM	Direct HPPASS trigger	Via service request/ERU

PSOC™ Control C3 MCU HPPASS CSG Configuration (ModusToolbox™):

- Configure TCPWM0 Group 0 Counter 2 in counter mode with 1-second period
- Enable overflow event for TCPWM trigger output 1
- Connect TCPWM trigger output 1 to HPPASS input trigger 0
- Set DAC operation mode to buffered mode
- Configure HPPASS input trigger 0 to trigger DAC mode start and buffer update

PSOC™ Control C1 MCU ACMP Configuration:

```
/* PSOC™ Control C1 - Analog Comparator (ACMP) Init */
const Cy_ACMP_CONFIG_t acmp_config = {
    .filter_disable         = false,
    .output_invert          = false,
    .hysteresis             = CY_ACMP_HYSTERESIS_20
};

Cy_ACMP_Init(ACMP0, CY_ACMP_INP_SOURCE_STANDARD_PORT, &acmp_config);
Cy_ACMP_EnableComparator(ACMP0);

/* Read comparator output */
bool over_current = Cy_ACMP_GetOutput(ACMP0);
```

⚠ ***Design Note:*** *The PSOC™ Control C1 MCU ACMP does not include an integrated DAC slope generator. Applications using PSOC™ Control C3 MCU's CSG slope compensation must be redesigned using the PSOC™ Control C1 MCU VADC with software-based slope calculation or an external DAC reference.*

6 Timer and PWM migration

6.1 Timer/PWM module comparison

Table 13 Timer/PWM module comparison

Feature	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU
Module Name	TCPWM	CCU4/CCU8
High-Res PWM	Not available (HRPWM is on PSC3M5xD only)	No HRPWM
16-bit Standard PWM	2× TCPWM Groups (8 ch each) = 16 ch	CCU4: 16 slices/CCU8: 8 slices
32-bit PWM	1× TCPWM Group (4 ch)	Not available (16-bit only)
Complementary PWM	Yes (TCPWM)	Yes (CCU8)
Dead Time	Yes	Yes (CCU8)
MOTIF/POSIF	1× MOTIF	1× POSIF
Quadrature Decoder	Yes (MOTIF)	Yes (POSIF/CCU4 position)
Hall Sensor Decoder	Yes (MOTIF)	Yes (POSIF)
Capture Mode	Yes	Yes (CCU4)
PWM Dither	Yes	Yes (CCU4)
Shadow Transfer	Yes	Yes (CCU4/CCU8)
Multi-Channel PWM Mode	Yes	Yes (CCU8 MCM)
PWM Clock Frequency	Up to 100 MHz (PERI clock)	Up to 96 MHz (PCLK)

PSOC™ Control C3 MCU TCPWM — Left-Aligned PWM Register Configuration:

```
MODE              = PWM
UP_DOWN_MODE      = COUNT_UP
AUTO_RELOAD_CC0   = 1
AUTO_RELOAD_LINE_SEL = 1
Period Register   → defines PWM period
CC0 Register      → defines duty cycle
```

PSOC™ Control C3 MCU — PDL TCPWM/PWM Code:

```
/* PSOC™ Control C3 - TCPWM PWM Init and Start */
Cy_TCPWM_PWM_Init(PWM_HW, PWM_NUM, &PWM_config);
Cy_TCPWM_PWM_Enable(PWM_HW, PWM_NUM, &PWM_config);
Cy_TCPWM_TriggerReloadOrIndex_Single(PWM_HW, PWM_NUM);
```

6 Timer and PWM migration

PSOC™ Control C1 MCU CCU4 — Edge-Aligned PWM Register Configuration:

```
CC4yTC.CMOD    = 0B
CC4yTC.TCM     = 0B
CC4yPRS.PRS    → defines PWM period
CC4yCRS.CRS    → defines duty cycle
CC4yPSL.PSL    = 1B (polarity)
```

PSOC™ Control C1 MCU — PDL CCU4/PWM Code:

```
/* PSOC™ Control C1 – CCU4 PWM Init and Start */
Cy_CCU4_Init(ccu4_0_HW, CY_CCU4_SLICE_MCMS_ACTION_TRANSFER_PR_CR);
Cy_CCU4_StartPrescaler(ccu4_0_HW);

Cy_CCU4_SLICE_CompareInit(PWM_HW, &PWM_compare_config);
Cy_CCU4_SLICE_SetTimerCompareMatch(PWM_0_HW, 11718U);
Cy_CCU4_SLICE_SetTimerPeriodMatch(PWM_0_HW, 46874U);
Cy_CCU4_EnableShadowTransfer(PWM_HW, CY_CCU4_SHADOW_TRANSFER_SLICE_0 |
                                     CY_CCU4_SHADOW_TRANSFER_DITHER_SLICE_0 |
                                           CY_CCU4_SHADOW_TRANSFER_PRESCALER_SLICE_0);
Cy_CCU4_SLICE_StartTimer(PWM_HW);
```

7 Communication peripherals

7.1 CAN

Table 14 CAN

Feature	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU
Module	CAN FD	MultiCAN+
CAN Instances	2× CAN-FD nodes	2 nodes
Protocol	CAN 2.0A/B + CAN-FD	CAN 2.0A/B only
Message Objects (MOs)	—	Up to 32 MOs
Max Bitrate	Up to 8 Mbps (FD data phase)	Up to 1 Mbps
CAN Transceiver	External	External

⚠ ***Critical*** *— CAN-FD Not Supported: PSC3M3xD supports 2× CAN FD nodes (one node up to 8 Mbps in the FD data phase). PSOC™ Control C1 MCU only supports Classical CAN 2.0A/B up to 1 Mbps. If your application requires CAN-FD, the PSOC™ Control C1 MCU family is not a suitable target — evaluate a higher-end derivative such as the PSOC™ Control C3 MCU with CAN-FD, or an XMC4000-series device.*

⚠ ***Errata*** *— MultiCAN+ Known Issues on PSOC™ Control C1 MCU:*

- MultiCAN_AI.H005: TxD pulse upon short disable request

- MultiCAN_AI.H006: Time stamp influenced by resynchronization

- MultiCAN_AI.H007: Additional application hint regarding message handling

- MultiCAN_AI.H009: Additional application note

- MultiCAN_TC.H003/H004: Test condition hints

Review all MultiCAN+ errata items before finalizing CAN implementation.

PSOC™ Control C1 MCU CAN Pin Configuration:

```
/*  Control C1 - CAN Tx/Rx Pin Assignment */
/* P4.9 = CAN Tx (CAN_TXD), P4.8 = CAN Rx (CAN_RXD) */
/* On KIT_PSC1M3_EVAL: TLE9351BVSJ CAN transceiver */

const Cy_GPIO_CONFIG_t can_tx_config = {
    .mode = CY_GPIO_MODE_OUTPUT_PUSH_PULL_ALT9
};
const Cy_GPIO_CONFIG_t can_rx_config = {
    .mode = CY_GPIO_MODE_INPUT_TRISTATE
};

Cy_GPIO_Init(P4_9, &can_tx_config); /* CAN TXD */
Cy_GPIO_Init(P4_8, &can_rx_config); /* CAN RXD */
```

7 Communication peripherals

PSOC™ Control C1 MCU MultiCAN+ Message Object Init:

```
/*  Control C1 - MultiCAN+ Node and MO Initialization */
const Cy_CAN_NODE_NOMINAL_BIT_TIME_CONFIG_t can_baud_config = {
    .can_frequency = 48000000U,   /* CAN clock = 48 MHz */
    .baudrate      = 500000U,     /* 500 kbps */
    .sample_point  = 8000U,       /* 80% sample point */
    .sjw           = 1U
};

Cy_CAN_Init(CAN, CY_CAN_CANCLKSRC_MCLK, 48000000U);
Cy_CAN_NODE_NominalBitTimeConfigure(CAN_NODE0, &can_baud_config);
Cy_CAN_NODE_Enable(CAN_NODE0);
```

8 Interrupt system

Table 15 Interrupt system

Feature	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU
Interrupt Controller	NVIC (Cortex®-M33)	NVIC (Cortex®-M0)
Priority Levels	8 levels (3-bit)	4 levels (2-bit)
Vector Table	Relocatable (VTOR)	Fixed/Relocatable
GPIO Interrupt	Port/Pin level (direct)	ERU-based
TrustZone® Interrupt Routing	Yes (SAU/NVIC integration)	Not applicable
Interrupt Latency	~12 cycles (M33)	~16 cycles (M0)

PSOC™ Control C3 MCU SCB exposes the following interrupt sources (each routed to a dedicated NVIC line):

Table 16 SCB Interrupt

Interrupt Signal	Function
`interrupt_master`	I2C master and SPI master functionality
`interrupt_slave`	I2C slave and SPI slave functionality
`interrupt_tx`	UART transmitter and TX FIFO
`interrupt_rx`	UART receiver and RX FIFO
`interrupt_i2c_ec`	Externally-clocked I2C

PSOC™ Control C1 MCU — NVIC Configuration:

```
/* PSOC™ Control C1 - NVIC Enable with Priority */
/* Note: Maximum 4 priority levels on Cortex-M0 */
NVIC_SetPriority(USIC0_0_IRQn, 3U);
NVIC_EnableIRQ(USIC0_0_IRQn);

/* PSOC™ Control C1 - External Trigger ISR Example */
void IRQ_Hdlr_4(void)
{
 /* Change boot mode on external trigger */
    XMC1000_BmiInstallationReq(0xFF40U);
}
```

⚠ ***Key Action*** *— Priority Levels: PSOC™ Control C3 MCU supports up to 8 NVIC priority levels (3-bit). PSOC™ Control C1 MCU supports only 4 levels (2-bit). All ISR priority assignments must be reviewed and consolidated during migration. Re-evaluate all preemption relationships carefully.*

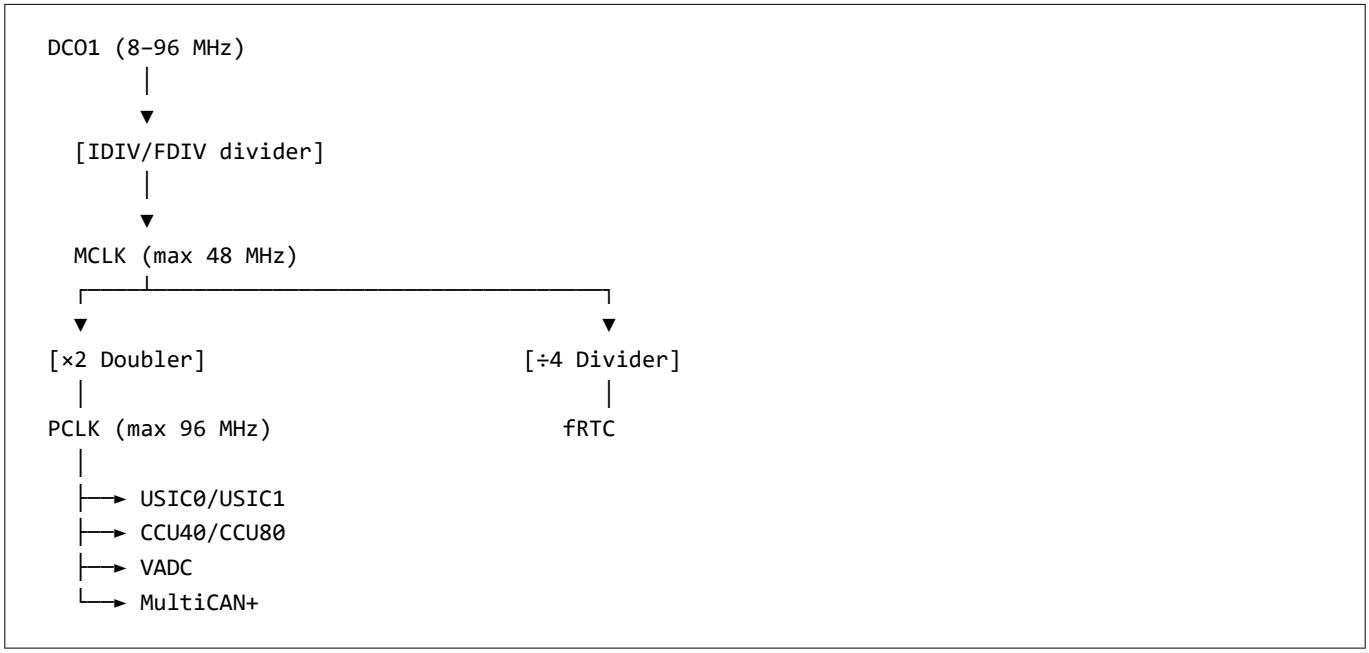
⚠ ***Key Action*** *— TrustZone® Interrupt Routing: All secure/non-secure interrupt routing through SAU must be completely removed — PSOC™ Control C1 MCU has no TrustZone®.*

9 Clock system

Table 17 Clock system

Feature	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU
Internal Oscillator	IMO + IHO + ILO + FLL	DCO1 (8–96 MHz), DCO2 (32 kHz)
PLL	2× DPLL	No PLL
External Crystal	Yes	Yes (4–25 MHz)
PCLK Source	Flexible divider network	MCLK doubler (PCLK = 2×MCLK)
Max System Clock	Up to 100 MHz	48 MHz (MCLK) / 96 MHz (PCLK)
RTC Clock	Backup domain RTC	RTC from DCO2 or OSC_LP
Oscillator Watchdog	Yes	Yes (with errata)
Clock Config Tool	ModusToolbox™ Device Configurator	ModusToolbox™ Device Configurator

PSOC™ Control C1 MCU Clock Tree:



PSOC™ Control C3 MCU — Clock Init (via BSP):

```
/* PSOC™ Control C3 - System Clock auto-configured by BSP init */
/* Up to 100 MHz via DPLL, handled by cybsp_init() */
cybsp_init();
```

9 Clock system

PSOC™ Control C1 MCU — SCU Clock Init in ModusToolbox™:

```
/* PSOC™ Control C1 - Configure MCLK = 48 MHz, PCLK = 96 MHz */
const Cy_SCU_CLOCK_CONFIG_t clock_config = {
    .fdiv     = 0U,
    .idiv     = 1U,
    .pclk_src = CY_SCU_CLOCK_PCLKSRC_DOUBLE_MCLK,  /* PCLK = 2×MCLK */
    .rtc_src  = CY_SCU_CLOCK_RTCCLKSRC_DCO2
};
Cy_SCU_CLOCK_Init(&clock_config);
/* SystemCoreClock is updated automatically */
```

⚠ ***Errata*** *— Oscillator Issues on PSOC™ Control C1 MCU:*

- OSC_CM.D001: Additions and corrections to ANAOSCHPCTRL register — affects all PSOC™Control C1 MCU AA-Step

- SCU_CM.D002: XTAL oscillator watchdog status output validity issue

- SCU_CM.022: External oscillator startup problem

- SCU_CM.023: Reset of XTAL oscillator watchdog via XOWDRES does not work as specified

Review all oscillator errata items before using an external crystal oscillator.

10 Power management

Table 18 Power management

Feature	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU
Power Modes	Active, Sleep, DeepSleep, DeepSleep-RAM, DeepSleep-OFF, Hibernate	Run, Sleep, DeepSleep
Backup Domain	Yes (Backup Reset, RTC, backup RAM)	Not available
Wake Sources	GPIO, RTC, WDT, LPComp, SCB	GPIO, RTC, WDT, ERU
Low-Power Oscillator	ILO (32 kHz)	DCO2 (32 kHz), OSC_LP
Supply Voltage	1.71 V – 3.6 V	1.8 V – 5.5 V
POR/BOD	Yes	Yes
LDO	Yes	Yes (internal)

PSOC™ Control C1 MCU — Sleep Mode Entry:

```
/* PSOC™ Control C1 - Enter Sleep (CPU clock gated, peripherals active) */
/* WFI instruction halts the CPU until next interrupt */
__WFI();  /* Wait for Interrupt — standard CMSIS */

/* PSOC™ Control C1 - Enter DeepSleep (system oscillator halted) */
Cy_SCU_POWER_WaitForInterrupt(CY_SCU_POWER_MODE_DEEPSLEEP, false);
```

⚠ ***Note*** *— Hibernate Mode: PSOC™ Control C3 MCU's Hibernate mode with complete RAM retention and backup domain is not available on PSOC™ Control C1 MCU. Applications using backup RAM, RTC retention in Hibernate, or LPComp as wake source must be significantly redesigned. Use PSOC™ Control C1 MCU DeepSleep with RTC/ERU wake instead.*

⚠ ***Voltage Warning:*** *The supply ranges differ on both ends. PSC3M3xD operates from 1.71 V to 3.6 V, while PSC1M3x operates from 1.8 V to 5.5 V. The minimum rises slightly (1.71 V → 1.8 V), but more importantly the PSC1M3x tolerates supplies up to 5.5 V whereas the PSC3M3xD ceiling is 3.6 V. Re-validate the regulator design, level-shifters on shared 5 V signals and any pull-up resistor sizing before migration.*

Feature	PSOC™ Control C1 MCU
PMSM FOC Support	Yes
Control Law	Fixed-point Q-format (no FPU)
Overcurrent Protection	Via 4× ACMP hardware comparators
CAN Control Interface	Yes — MultiCAN+ (2 nodes, 32 MOs)
Evaluation Kit	TARGET_KIT_PSC1M3_EVK (VQFN-48)
PWM Module	CCU80 with complementary outputs and dead time
ADC Trigger	CCU80 Slice 3 trigger
Current Sensing	Three-shunt/single-shunt
Position Sensing	PLL Estimator/POSIF (Hall + QEI)

11 PCB hardware design considerations

Refer the Hardware design guide document.

11.1 KIT_PSC1M3_EVAL hardware overview

The KIT_PSC1M3_EVAL is the primary evaluation platform for the PSOC™ Control C1 MCU family. It hosts the PSC1M3FBLGQ0AT MCU (Cortex®-M0, 200 KB Flash, 16 KB SRAM, VQFN-48 package) together with an on-board XMC™ Link debug probe (XMC4200 Cortex®-M4F MCU exposing SWD primary and SPD alternate, with an integrated USB-UART bridge), user LED on P5.0, user button on P0.0, three-phase motor drive header, CAN transceiver footprint and an Arduino-compatible header. The BSP `TARGET_KIT_PSC1M3_EVK` provides the device-configurator design.modus, linker scripts (GCC_ARM and ARM toolchains) and the `Cy_`/PSC1Mx Library (`mtb-dsl-psc1m3`) used by all examples in this guide.

Component	Device/Value	Purpose
MCU	PSC1M3x device on KIT_PSC1M3_EVAL (see Section 11.3 for the exact ordering code/ package)	Target MCU — 200 KB Flash, 16 KB SRAM
Debug MCU	XMC4200-QFN48 (Cortex®-M4F)	On-board debugger supporting SWD/SPD
CAN Transceiver	TLE9351BVSJ	CAN network testing via MultiCAN+
3.3 V Regulator	TLS205B0EJV33	3.3 V power rail
5V Regulator	TLS205B0LD V50	5 V power rail
Crystal	12 MHz (AEC-Q200)	External clock source
Debug Interface	SWD/SPD	Via XMC4200 on-board debugger
USIC	4 serial channels	Configurable UART/SPI/I2C/I2S
ADC	12-bit, up to 12 channels, 2 S/H stages, up to 1.1 MS/s	Current/voltage sensing
CCU4/CCU8	16 + 8 slices at 96 MHz	PWM and capture

11.2 Pin count reduction guidance

When migrating from PSC3M3xD (VQFN-48/E-LQFP-48/VQFN-64/E-LQFP-64, up to 64 pins, 39 GPIOs and up to 18 analog-capable pins) to PSC1M3x (TSSOP-38/VQFN-40/VQFN-48/LQFP-48, up to 43 I/Os and up to 12 analog inputs), a systematic pin reduction is required:

1. Identify all PSOC™ Control C3 pins and classify each as: power, ground, GPIO, analog, communication, debug, reset, or no-connect
2. Map each used pin to the corresponding PSOC™ Control C1 VQFN-40 or VQFN-48 function using the PSOC™ Control C1 datasheet pin assignment table
3. Smart I/O Logic — any combinatorial logic implemented in PSOC™ Control C3 Smart I/O (LUTs) must be re-implemented externally in logic ICs or in firmware
4. Analog pin count — verify that all required analog input channels fit within the available 12–14 VADC inputs
5. Power pins — verify VDD (1.8 V–5.5 V), VDDP (I/O power), VSS, VSSA decoupling on the new footprint

12 Boot and programming migration

12.1 PSOC™ Control C3 MCU boot process

PSOC™ Control C3 MCU uses a multi-step acquisition algorithm:

1. Check boot IDLE state
2. Acquire in test mode
3. Acquire using vector catch
4. Security initialization (TrustZone® provisioning)

The PSC3M3xD provides built-in Device Firmware Update (DFU) support in boot ROM via:

- UART — P8.1 (TX), P8.3 (RX) with pull resistors to VDDIO_0
- I2C — P8.1 (SDA), P8.3 (SCL) with 1 kΩ pull-ups to VDDIO_0
- SPI — P8.0/8.1/8.2/8.3 with CS, MOSI, MISO, CLK

12.2 PSOC™ Control C1 MCU Boot Mode Index (BMI)

PSOC™ Control C1 MCU boot mode is controlled via the Boot Mode Index (BMI) register:

Table 19 Boot Mode Index

BMI Value	Boot Mode
`0xFFC0`	Boot from Flash (normal operation)
`0xFF40`	Boot from pin (alternate, for debugging)
`0xFFE0`	Boot via UART/ASC (in-system programming)

```
/* PSOC™ Control C1 - BMI Change via External Trigger ISR */
void IRQ_Hdlr_4(void)
{
    /* Change to alternate boot-from-pin mode on external trigger */
    XMC1000_BmiInstallationReq(0xFF40U);
}

/* PSOC™ Control C1 - Standard Flash Boot Selection */
XMC1000_BmiInstallationReq(0xFFC0U);

/* PSOC™ Control C1 - Linker Script Change for Custom Start Address */
/* Change code start address to 0x10020000 in linker script  */
/* for alternate flash location.                              */
```

12.3 Programming interface mapping

Programming and debugging both devices is done from ModusToolbox™ via OpenOCD. The probe and protocol mapping is:

- PSOC™ Control C3 MCU: KitProg3 (on-board) or MiniProg4 over SWD/JTAG
- PSOC™ Control C1 MCU (KIT_PSC1M3_EVAL): on-board debug MCU exposing SWD (primary) and SPD (single-pin alternate)
- Production programming: gang programmers supporting SWD; the BMI (Boot Mode Index) on PSOC™ Control C1 MCU must be set to a debug-enabled value (see Section 12.2) before first programming

12 Boot and programming migration

Table 20 Programming interface mapping

DFU/Programming Interface	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU
UART	Baud=115200, 8N1, P8.1/P8.3	BMI UART mode: USIC ASC
I2C	Speed=400K, addr=0x35, P8.1/P8.3	Not natively supported in BMI
SPI	Via SCB SPI	Not natively supported in BMI
Debug SWD	KitProg3	J-Link/DAP-miniWiggler
SPD (Single Pin)	Not available	Supported on PSOC™ Control C1

13 API reference mapping

13.1 Complete API migration table

The following table summarizes the most frequently used PDL/HAL APIs on PSOC™ Control C3 MCU and their PSC1Mx Library (`mtb-dsl-psc1m3`) equivalents on PSOC™ Control C1 MCU. Use this table as a search-and-replace checklist when porting application code.

Table 21 API migration table

Function Category	PSOC™ Control C3 MCU PDL API	PSOC™ Control C1 MCU PSC1Mx Library (mtb-dsl-psc1m3) API
System Init	`cybsp_init()`	`cybsp_init()`
GPIO Init	`Cy_GPIO_Pin_Init()`	`Cy_GPIO_Init()`
GPIO Set High	`Cy_GPIO_Set()`	`Cy_GPIO_SetOutputHigh()`
GPIO Set Low	`Cy_GPIO_Clr()`	`Cy_GPIO_SetOutputLow()`
GPIO Toggle	`Cy_GPIO_Inv()`	`Cy_GPIO_ToggleOutput()`
GPIO Read	`Cy_GPIO_Read()`	`Cy_GPIO_GetInput()`
UART Init	`Cy_SCB_UART_Init()`	`Cy_UART_CH_Init()`
UART Enable	`Cy_SCB_UART_Enable()`	`Cy_UART_CH_Start()`
UART TX String	`Cy_SCB_UART_PutString()`	`Cy_UART_CH_Transmit()` (per byte)
UART RX Byte	`Cy_SCB_UART_Get()`	`Cy_UART_CH_GetReceivedData()`
SPI Init	`Cy_SCB_SPI_Init()`	`Cy_SPI_CH_Init()`
SPI Enable	`Cy_SCB_SPI_Enable()`	`Cy_SPI_CH_Start()`
SPI Write	`Cy_SCB_SPI_Write()`	`Cy_SPI_CH_Transmit()`
SPI Bus Busy	`Cy_SCB_SPI_IsBusBusy()`	Poll `TRANSMIT_SHIFT_INDICATION` flag
I2C Init	`Cy_SCB_I2C_Init()`	`Cy_I2C_CH_Init()`
I2C Enable	`Cy_SCB_I2C_Enable()`	`Cy_I2C_CH_Start()`
I2C Master Write	`Cy_SCB_I2C_MasterWrite()`	`Cy_I2C_CH_MasterTransmit()`
I2C Master Start	`Cy_SCB_I2C_MasterSendStart()`	`Cy_I2C_CH_MasterStart()`
I2C Master Stop	`Cy_SCB_I2C_MasterSendStop()`	`Cy_I2C_CH_MasterStop()`
ADC Global Init	`Cy_SAR2_Init()`	`Cy_VADC_GLOBAL_Init()`
ADC Group Init	`Cy_SAR2_Channel_Init()`	`Cy_VADC_GROUP_Init()`
ADC Start	`Cy_SAR2_SetChanMask()`	`Cy_VADC_GROUP_QueueAddChannelToSequence()`
ADC Read	`Cy_SAR2_Channel_GetResult()`	`Cy_VADC_GROUP_GetDetailedResult()`
ACMP Init	`Cy_HPPASS_CSG_CompInit()`	`Cy_ACMP_Init()`

(table continues...)

Table 21 (continued) API migration table

ACMP Start	`Cy_HPPASS_CSG_CompEnable()`	N/A
ACMP Read	`Cy_HPPASS_CSG_GetCompStatus()`	N/A
PWM Init	`Cy_TCPWM_PWM_Init()`	`Cy_CCU8_SLICE_CompareInit()`
PWM Enable Clock	`Cy_TCPWM_PWM_Enable()`	`Cy_CCU8_EnableClock()`
PWM Start	`Cy_TCPWM_TriggerReloadOrIndex_Single()`	`Cy_CCU8_SLICE_StartTimer()`
PWM Set Period	`Cy_TCPWM_Counter_SetPeriod()`	`Cy_CCU8_SLICE_SetTimerPeriodMatch()`
PWM Set Duty	`Cy_TCPWM_PWM_SetCompare0()`	`Cy_CCU8_SLICE_SetTimerCompareMatch()`
Timer Init (CCU4)	`Cy_TCPWM_Counter_Init()`	`Cy_CCU4_SLICE_CompareInit()`
WDT Init	`Cy_WDT_Init()`	`Cy_WDT_Init()`
WDT Service	`Cy_WDT_ClearWatchdog()`	`Cy_WDT_Service()`
WDT Enable	`Cy_WDT_Enable()`	`Cy_WDT_Start()`
NVIC Enable	`NVIC_EnableIRQ()`	`NVIC_EnableIRQ()`
NVIC Priority	`NVIC_SetPriority()`	`NVIC_SetPriority()`
SysTick Init	`Cy_SysTick_Init()`	`SysTick_Config()`
Critical Section Enter	`Cy_SysLib_EnterCriticalSection()`	`__disable_irq()`
Critical Section Exit	`Cy_SysLib_ExitCriticalSection()`	`__enable_irq()`

14 Migration checklist

14.1 Pre-migration hardware review

- [ ] Select target device: confirm PSC1M3FBLGQ0AT meets pin count, peripheral count (USIC/CCU4/CCU8/VADC/ACMP/CAN) and analog channel count needs
- [ ] Audit all PSOC™ Control C3 MCU peripheral usage against PSOC™ Control C1 MCU availability
- [ ] Confirm total code/data size fits within 200 KB Flash and 16 KB SRAM
- [ ] Identify all FPU/DSP/CORDIC-dependent code sections — convert to fixed-point
- [ ] Identify TrustZone®/SAU usage — remove completely (not available on PSC1M3x)
- [ ] Identify CryptoLite (AES-128, SHA-256, TRNG) usage — replace with software
- [ ] Verify supply voltage range: PSC1M3x require 1.8 V to 5.5 V (vs. 1.71 V to 3.6 V on PSC3M3xD)
- [ ] Verify CAN dependency — PSOC™ Control C1 MCU (PSC1M3) provides Classical CAN 2.0 A/B via the MultiCAN+ block; CAN-FD is not available
- [ ] Verify CAN-FD dependency — not supported on PSOC™ Control C1 MCU
- [ ] Verify ISO 7816 dependency — not available on PSC1M3
- [ ] Verify DeepSleep SCB dependency — not available on PSC1M3
- [ ] Review DMA channel count — PSC1M3 have only 4 channels vs. 32 on PSOC™ C3 MCU
- [ ] Review PCB layout requirements for VADC and oscillator per AP32316
- [ ] Verify QFN exposed pad is connected to board ground
- [ ] Select appropriate package: VQFN-40 (compact), VQFN-48 (more I/O), or LQFP-48 (prototyping)
- [ ] Map all PSOC™ C3 MCU Smart I/O LUT logic to external logic or firmware

14.2 ModusToolbox™ project setup

- [ ] Create new PSOC™ Control C1 MCU project in ModusToolbox™ (depending on device selection)
- [ ] PSOC™ Control C1 MCU: Select BSP `TARGET_KIT_PSC1M3_EVK` (VQFN-48 PSC1M3FBLGQ0AT)
- [ ] If targeting a custom PSC1M3 board, start from `TARGET_KIT_PSC1M3_EVK` and adjust pins, clocks, and the Device Configurator to match the production schematic
- [ ] Add PSC1Mx Library (mtb-dsl-psc1m3) (`mtb-dsl-psc1m3`) via Library Manager
- [ ] Configure MCLK = 48 MHz, PCLK = 96 MHz via `Cy_SCU_CLOCK_Init()`
- [ ] Configure Boot Mode Index (BMI) if custom boot mode needed
- [ ] Set up debug interface (SWD primary, SPD as alternate)
- [ ] Recreate all peripherals in Device Configurator
- [ ] Verify FreeRTOS configuration if used (stack sizes must be reduced for 16 KB SRAM)

14.3 Firmware code migration

- [ ] Replace all `Cy_` PDL APIs with `Cy_` PSC1Mx Library (mtb-dsl-psc1m3) APIs (see Section 11.1)
- [ ] Replace all `cyhal_` HAL APIs with PSC1Mx Library (mtb-dsl-psc1m3) equivalents
- [ ] Convert all floating-point code to Q-format fixed-point (no FPU on PSC1M3)
- [ ] Remove all TrustZone® (SAU, secure/non-secure, NS_CALL) code
- [ ] Remove or replace AES-128 and SHA-256/TRNG with software crypto library
- [ ] Reduce NVIC priority levels: maximum 4 levels on PSC1M3 (from 8 on PSOC™ C3 MCU)
- [ ] Migrate clock init: replace DPLL-based config with `Cy_SCU_CLOCK_Init()`

- [ ] Apply USIC ASC corrected init per errata USIC_AI.D001
- [ ] Apply I2C slave recovery sequence per IIC lock-up errata
- [ ] Review CCU8 PWM Active/Passive Rules (CCU8_AI.D001 errata)
- [ ] Review ANAOSCHPCTRL register corrections (OSC_CM.D001 errata)
- [ ] Review oscillator watchdog errata (SCU_CM.022, SCU_CM.023)
- [ ] Review all USIC_AI errata items (AI.008, .014, .017, .018, .021)
- [ ] For PSOC™ Control C1 MCU with CAN: Review MultiCAN+ errata items (H005, H006, H007, H009, TC.H003, TC.H004)
- [ ] Review WDT Pre-warning Mode correction (WDT_CM.D001)
- [ ] Update linker scripts for PSC1M3 memory map
- [ ] Update interrupt vector table and ISR naming conventions

14.4 Verification and testing

- [ ] Compile project without errors or warnings
- [ ] Test all USIC channels: UART, SPI, I2C — verify baud rates and protocol timing
- [ ] Validate VADC (ADC) performance with correct PCB layout per AP32316
- [ ] Validate CCU4/CCU8 PWM output timing and waveforms
- [ ] For PSOC™ Control C1 MCU: Verify MultiCAN+ 2.0 A/B operation and message object handling (Classical only)
- [ ] Verify NVIC priority assignments and interrupt latency under load
- [ ] Test Sleep and DeepSleep power modes with correct wake sources
- [ ] Run full functional regression against original PSOC™ Control C3 MCU test suite

15 Summary

Migrating from the PSC3M3xD (Cortex®-M33 at 100 MHz, HPPASS 6-Msps SAR ADC, TrustZone® + CryptoLite, 256 KB Flash/64 KB SRAM) to the PSC1M3x (Cortex®-M0 at 48 MHz, USIC/CCU4/CCU8/VADC 1.1 MS/s, MultiCAN+ Classical) is best approached as a hardware-first activity: lock down the pinout against the PSC1M3x VQFN-48 datasheet, audit memory and clock budgets against the 200 KB Flash/16 KB SRAM/48 MHz MCLK/96 MHz PCLK envelope, and remove every dependency on FPU, DSP, CORDIC (the PSC1M3x MATH coprocessor is 24-bit only), TrustZone, and CryptoLite before touching the firmware.

On the firmware side the migration becomes mechanical once the BSP switches to `TARGET_KIT_PSC1M3_EVK` and the `mtb-dsl-psc1m3` library is added through the Library Manager: PDL `Cy_SCB_*` calls are replaced by `Cy_UART_CH_*`, `Cy_SPI_CH_*` and `Cy_I2C_CH_*`; `Cy_TCPWM_*` is replaced by `Cy_CCU4_*`/`Cy_CCU8_*`; `Cy_SAR2_*` is replaced by `Cy_VADC_*`; clock initialization moves from the auto-configured DPLL to `Cy_SCU_CLOCK_Init()` (MCLK = 48 MHz, PCLK = 96 MHz); and GPIO interrupts move from direct port/pin configuration to the ERU.

Apply the errata workarounds called out in Sections 4–9 (notably USIC_AI.D001 for ASC mode, the I2C slave-recovery sequence, the ANAOSCHPCTRL register corrections and the WDT pre-warning fix) and run the firmware against the verification checklist in Section 14.4 before release. With these steps complete, an application originally targeted at PSOC™ Control C3 MCU can be brought up on KIT_PSC1M3_EVAL and progressed to a custom PSC1M3 board with confidence.

Table 22 Migration summary

Migration topic	PSOC™ Control C3 MCU	PSOC™ Control C1 MCU	Key Migration Action
CPU Core	Cortex®-M33 at 100 MHz	Cortex®-M0 at 48 MHz	Remove FPU/DSP/CORDIC dependency; convert to Q-format
Security	TrustZone® + CryptoLite (AES-128, TRNG)	CRC engine only	Remove TrustZone® code; replace crypto with software
Memory	1 MB Flash/256 KB SRAM	200 KB Flash/16 KB SRAM	Mandatory code and data size optimization
GPIO	HSIOM + Smart I/O (LUTs)	Fixed pin mux	Remap all pins; reimplement LUT logic
GPIO Interrupt	Direct pin interrupt	ERU-based	Configure ERU for each external interrupt
Serial I/F	SCB (UART/SPI/I2C)	USIC — 4 channels	Remap SCB→USIC; apply USIC errata
ADC	HPPASS 6-Msps SAR	VADC 1.1 MS/s (PSC1M3x) — 2 kernels	Redesign high-speed sampling; check timing
Comparators	5× HPPASS CSG (with DAC)	4× ACMP (no DAC)	PSC1M3: Use software or external; PSOC™ Control C1 MCU: Map to ACMP
Timers/PWM	TCPWM 32-bit/HRPWM	CCU4 + CCU8 16-bit	Map TCPWM→CCU; review CCU errata
CAN	2× CAN-FD at 8 Mbps	MultiCAN+ Classical at 1 Mbps	Select PSOC™ Control C1 MCU if CAN required; no CAN-FD support

(table continues...)

15 Summary

Table 22 (continued) Migration summary

Interrupt System	8-level NVIC	4-level NVIC	Consolidate priorities; review ISR dependencies
Clock	2× DPLL at 100 MHz	DCO1/SCU at 48 MHz maximum	No PLL; reconfigure SCU; check OSC errata
Power Modes	Active/Sleep/DeepSleep/ Hibernate	Run/Sleep/DeepSleep	Remove Hibernate; redesign low-power wakeup
Boot/Programming	TrustZone® DAP + DFU	BMI + UART/SWD/SPD	Reconfigure boot mode; update debug setup
Motor Control	FPU FOC + HPPASS + MOTIF	Fixed-point FOC + VADC + ACMP + POSIF	PSC1M3: Software OCL; PSOC™ Control C1 MCU: Use ACMP + library
APIs	`Cy_` PDL/`cyhal_` HAL	`Cy_` PSC1Mx Library (mtb-dsl-psc1m3)	Full API replacement (see Section 11)
PCB	PSOC™ C3 MCU layout	PSOC™ Control C1 MCU hardware design guide	Follow AP32316 for VQFN-40/VQFN-48/ LQFP-48
Errata	Refer to Section 10, PSOC™ Control C3 MCU Entry Line datasheet	Multiple AA-step items	Review all errata before production

Revision history

Document revision	Date	Description of changes
**	2026-06-29	Initial release
*A	2026-07-28	Web release

Trademarks

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 2026-07-28

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-epz1777548293178

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.