Peripheral driver library (PDL) release notes

Version: 3.1.6

About this document

Scope and purpose

Thank you for your interest in the peripheral driver library (PDL) version 3.1.6. This document lists the content of the release package.

Intended audience

This document is intended for anyone who uses the PDL for PSoC™ Creator.

Table of contents

About this document **1**
Table of contents **2**
1 Tool information **3**
1.1 Overview 3
1.2 New features 3
1.2.1 Updated middleware and software 3
1.3 Limitations 4
1.4 Known issues 5
1.5 Device support 7
1.6 Peripheral drivers 7
1.7 Middleware and other software 9
1.8 Utilities 9
1.9 Tools 10
1.10 Supported toolchains 10
1.11 Folder structure 10
1.12 Documentation 11
1.13 Technical support 11

1 Tool information

1.1 Overview

PDL v3.x simplifies software development of the PSoC™ 6 MCU family of devices using PSoC™ Creator. This release provides the following features:

- PSoC™ 6 digital and analog peripheral drivers, which enable rapid peripheral software development in PSoC™ Creator and third-party IDEs
- The Arm® Cortex® Microcontroller Software Interface Standard (CMSIS) core access header files directly from the CMSIS 5.3.0 release
- CMSIS-compliant device header files, startup code (platform initialization), and device configuration header files
- Fully configurable Device Firmware Update (DFU, earlier bootloader) SDK ready to be integrated into any application
- FreeRTOS source code integrated with the PDL
- Bluetooth® Low Energy (LE) middleware and stack library
- Secure Image reference design, which demonstrates Trusted Execution Environment (TEE) execution on the CM0+ core
- PDL Application Programming Interface (API) reference
- PDL user guide

Note: *PDL v3.x does not support FM microcontroller devices – use PDL v2.1 to develop firmware for FM0+ and FM4 MCU devices. Use PDL v2.0 for FM3 MCU devices. All supported versions of the PDL are available through the PDL product page.*

Note: *While PSoC™ Creator remains fully supported and must be updated periodically, it is recommended all users move to ModusToolbox™ software for all future designs of PSoC™ 6 and future families of Infineon programmable devices. ModusToolbox™ software supports the new CY8C6xx8, CY8C6xx9, and CY8C6xxA series has new drivers for these series (such as SD Host and DMAC) and has new middleware for all the families such as CAPSENSE™ CSD ADC, and Bluetooth® LE. For more details, visit the ModusToolbox™ landing page at www.infineon.com/modustoolbox.*

1.2 New features

This release includes the following new and updated drivers and middleware. See the driver documentation sections of the PDL API reference for more details.

1.2.1 Updated middleware and software

- Bluetooth® LE 2.80. Bug fixing. For more detail, see the section “Changelog” of the Bluetooth® LE middleware library API reference.

1.3 Limitations

Table 1 Limitations

Limitation	Description
USB configurator	As in previous releases, supported only on the Windows x64 platform. Perform the development on the x64 platform. It also requires the Microsoft Visual C++ Redistributable packages for Visual Studio 2017. The installer provides the option to install that package.
RAM and flash memory consumption increase	As in previous releases, this version uses a register access layer and register access macros instead of direct register access. This makes the register access device independent. To support migration between devices without the need to recompile the PDL, the driver logic that handles the differences in silicon IP blocks is performed at runtime instead of compile time.
Reserved system resources	As in previous releases, the PDL reserves certain system resources for internal use. These resources include IPC resources, such as the first 16 IPC semaphores and interrupt lines to the Cortex® M0+ CPU. Using any of these resources in your design leads to unexpected behavior. See the PDL API reference for details.
SVD file schema	The PDL provides SVD files that use schema version 1.3. A few IDEs use a previous schema version. As a result, the IDE debugger does not display device registers. This issue disappears as third parties update their IDEs.
SYSPM 4.0 is not backward compatible	See the SYSPM driver documentation within the PDL API reference for the design impact, backward compatibility aspects, and migration guide
Cache coherency	PSoC™ 6 MCU does not support cache coherency. As a result, when a row of flash that executes instructions is written/updated, the updated information is not reflected in the cache. The cache should be invalidated in the firmware during such instances. This is applicable for both CM4 and CM0+ cache. In other words, the appropriate (CM0+ and/or CM4) cache should be invalidated.
Third-party IDE integration	The project templates for IAR and MDK are still provided and reflect the changes done in the PDL organization. The project templates for GCC are discontinued. For GCC, use ModusToolbox™ software, which has complete support of the MAKE command line flow for various compilers.
IPC and Crypto drivers no longer use configuration files	Older code examples and customer projects may have them. Move all needed configuration code from configuration files to the application source/header files.

1.4 Known issues

Table 2 List of known issues

Known issue	Workaround

The function `malloc()` does not return an error when the allocation size is bigger than the heap size	Correct operation of malloc and related functions depends on the working implementation of the 'sbrk' function. Newlib-nano (default C runtime library used by the GNU Arm® Embedded toolchain) provides a weak 'sbrk' implementation that does not check for heap and stack collisions during excessive memory allocations. To ensure the heap always remains within the range defined by `__HeapBase` and `__HeapLimit` linker symbols, provide a strong override for the 'sbrk' function:
	`#include <errno.h>`
	`#include <malloc.h>`
	
	`void *_sbrk_r(struct _reent *reent, ptrdiff_t diff)`
	`{`
	`(void) reent;`
	
	`extern uint32_t __HeapBase;`
	`extern uint32_t __HeapLimit;`
	
	`static uint8_t *_cur_brk = (uint8_t *)&__HeapBase;`
	`static uint8_t *_heap_end = (uint8_t *)&__HeapLimit;`
	
	`uint8_t *_old_brk = _cur_brk;`
	`if ((_cur_brk + diff) > _heap_end)`
	`{`
	`return (void *)-1;`
	`}`
	`_cur_brk += diff;`
	`return _old_brk;`
	`}`

Preproduction PSoC™ 6 devices do not support changing core voltage if the protection context (PC) does not equal 0	See SYSPM driver API reference for details. This also impacts dependent drivers and middleware: FLASH, Bluetooth® LE, Emulated EEPROM, and DFU.
The Watchdog Timer (WDT) lock state is not retained during the system's Deep Sleep power mode. After wakeup from the system Deep Sleep, the WDT is locked.	To change the configuration of WDT, Clk LF, ILO, or PILO, unlock the WDT (call `Cy_WDT_Unlock()`) after a wakeup from the system Deep Sleep

The `Cy_SysClk_EcoEnable` and `Cy_SysClk_EcoGetStatus` functions return `success` status when ECO is disconnected	Check the ECO_OK bitfield after the return success status:
	`cy_en_sysclk_status_t status = Cy_SysClk_EcoEnable(3000);`
	`if(status == CY_SYSCLK_SUCCESS)`

Known issue	Workaround

	`{`
	`  if(0UL == (SRSS_CLK_ECO_STATUS & SRSS_CLK_ECO_STATUS_ECO_OK_Msk))`
	`  {`
	`  /* There is something wrong with ECO */`
	`  status = CY_SYSCLK_TIMEOUT;`
	`  }`
	`}`

Configuration of PDL build options in PSoC™ Creator is reset for every updated middleware	Open the Peripheral Driver Library tab from the **Project > Build Settings...** menu, note the configuration for PDL 3.0.4, and select the missing checkboxes for PDL 3.1 manually. The affected components are Crypto, DFU SDK, emWin, Retarget I/O, and FreeRTOS.
Arm® MDK linker returns an error during the build of DFU SDK with a custom DFU linker script	Remove the `__asm void cy_DFU_mdkAsmDummy(void);` function prototype from the *dfu_mdk_common.h* header file
The USB device ignores LPM requests after wake up from Deep Sleep.	Call USBFS driver `Cy_USBFS_Dev_Drv_Lpm_SetResponse()` after calling `Cy_USBFS_Dev_Drv_Resume()` to restore response to the LPM packets
The USB device modes with DMA do not work after wake up from Deep Sleep, because of an incorrect restore of the ARB_CFG register	Save the ARB_CFG values before entering Deep Sleep and restore after calling `Cy_USBFS_Dev_Drv_Resume`.
Em_EEPROM Read function may become unresponsive in the infinite loop	See KBA227502
Em_EEPROM Read function returns incorrect data when restored from a redundant copy	See KBA227530
The USB configurator supports import from the HID descriptor tool. Current version 2.4 contains an error related to strings.	Before importing, check the values of String Index, String Minimum, and String Maximum items according to spec HID 1.11 and fix these items manually in the file generated by the HID descriptor tool.
The CY_BLE_EVT_STACK_BUSY_STATUS event may not return a CY_BLE_STACK_STATE_FREE state if the application initiates an active connection (Peripheral/Central GAP role) along with scan activity (GAP Observer) with a high-duty cycle (scan window value is close to scan interval). Such behavior is observed only when Bluetooth® LE is operated in dual CPU mode.	Increase the scan interval and reduce the scan window values to have a ratio of at least 1:2
In a few rare cases, waking the system from Deep Sleep could potentially cause a hard fault. This is because the system requests early access to flash/Sflash (before they are ready to be accessed).	See KBA229335

Known issue	Workaround
The GCC_Arm® linker script does not ensure the .data section fits into the flash memory, so no linker error is generated in this case while the project becomes non-functional.	Manually ensure the .data section fits into the flash memory using the project .map file. The issue in the GCC_Arm® linker script results in silent failure.

1.5 Device support

The PDL includes:

- Device-specific header files that define all peripheral registers and bits in the device
- CMSIS-compliant startup code to initialize the system after a device reset and transfer the code execution to the `main()`
- Linker files for each supported device and toolchain
- SVD files with a detailed description of peripherals, registers, fields, and bit values

This release supports devices in the PSoC™ 6 MCU architecture such as PSoC™ 62 and PSoC™ 63. It does not support CY8C6xx8, CY8C6xx9, and CY8C6xxA. Use the Eclipse IDE for ModusToolbox™ software, which contains the PDL and middleware within the PSoC™ 6 software library.

1.6 Peripheral drivers

The PDL provides a high-level API to configure, initialize, and use a peripheral driver. The drivers are designed for peripheral IP blocks. They work on any PSoC™ 6 device that instantiates that IP block.

Table 3 Peripheral drivers

Driver	Description	API functionality
CRYPTO	Cryptographic operations	Perform cryptographic operations on user-designated data
CTB	Continuous time block	Configure and access the analog CTB
CTDAC	Continuous-time DAC	Generate a 12-bit DAC output voltage from the reference
DMA	Direct memory access	Perform direct memory transfers
EFUSE	Electronic Fuses	Read the customer-accessible electronic fuses
FLASH	Flash memory	Manage flash memory operations
GPIO	General purpose I/O ports	Configure and access device input/output pins
I2S	Inter-IC Sound	Manage digital audio streaming to external I2S devices
IPC	Inter-process communication	Manage data transfer between CPUs or processes in a device
LPCOMP	Low-power comparator	Fast detection of voltage changes in both normal and ultra-low power operation
LVD	Low-voltage detection	Monitor whether the VDDD voltage level is above the configurable threshold
MCWDT	Multi-counter watchdog timer	Manage counters to create a free-running timer or periodic interrupts
PDM_PCM	PDM to PCM converter	Convert one-bit digital audio streaming data to PCM data
PROFILE	Energy profiler	Measure relative energy consumption of monitored operations

Tool information

Driver	Description	API functionality
PROT	Memory protection	Manage the MPU, Shared MPU (SMPU), and Peripheral Protection Unit (PPU)
RTC	Real-time clock	Manage calendar dates and clock time
SAR ADC	SAR ADC subsystem	Manage a fast 12-bit multichannel SAR ADC with a sample rate of 1 Msps
SCB	Serial Communication Block	Manage serial communication as I2C, SPI, or UART
SMIF	Serial Memory Interface	Manage an SPI-based interface to external memory devices
SYSANALOG	System analog reference	Generate highly accurate reference voltages and currents for the analog subsystem
SYSCLK	System clock	Manage system and peripheral clocks
SYSINT	System interrupt	Manage interrupts and exceptions, in conjunction with the CMSIS core NVIC API
SYSLIB	System library	Utility functions to handle delays, register read/write, asserts, software reset, silicon unique ID, and more
SYSPM	System power modes	Manage power modes and get power mode status
SYSTICK	Systick timer	Manage a 24-bit down-counter timer
TCPWM	Timer/counter-pulse width modulation	Manage a 16- or 32-bit periodic counter, PWM, or Quadrature decoder
TRIGMUX	Trigger multiplexer	Manage the multiplexing of trigger outputs to specific trigger inputs across multiple peripherals
WDT	Watchdog timer	Manage a watchdog timer

1.7 Middleware and other software

The PDL includes the following middleware components:

Table 4 Middleware and other software

Middleware	Description
Bluetooth® Low Energy (LE)	Bluetooth® Low Energy (LE) stack, along with a comprehensive set of APIs to configure the Bluetooth® LE stack and the underlying hardware. The Bluetooth® LE middleware also provides a general interface between the Bluetooth® LE application and the Bluetooth® LE stack module.
Emulated EEPROM	Emulated EEPROM middleware provides an API that allows creating an Emulated EEPROM in flash, with the ability to do wear leveling and restore corrupted data from a redundant copy
Embedded graphic library (emWin)	emWin is an embedded graphic library and graphical user interface (GUI) framework designed to provide an efficient, processor- and LCD controller-independent GUI for any application that operates with a graphical display. It is compatible with single-task and multitasks environments. Developed by SEGGER Microcontroller, Infineon has licensed the high-performance emWin library from SEGGER and offers it for free to customers.
USB device	The USB device middleware provides a full-speed USB 2.0 Chapter 9 specification-compliant device framework. It uses the USBFS driver from PDL to interface with the hardware. The middleware provides support for Audio, CDC, and HID classes. In addition, it allows the implementation of other class support. The standalone USB Device Configuration tool is shipped in PDL to make it easy to construct the USB Device descriptors.
DFU SDK	The Device Firmware Update (DFU, earlier bootloader) SDK allows you to design flexible bootloading applications with varying levels of complexity
RTOS	FreeRTOS v10.0.1 source code is integrated with the PDL
Secure Image	The Secure Image is a project template intended to run only on the CM0+ secure processor in a multi-processor system. It sets up hardware and software protection for the system, validates the user application, and jumps to its starting point. The template provides a basic secure system sufficient for most applications. You may modify or replace the template to match the requirements of a specific user system.

1.8 Utilities

The utilities directory contains source files you can use to redirect standard I/O to the user-defined target hardware.

1.9 Tools

The tools directory contains applications you can use to configure a software component or to perform post-build processes for all supported toolchains.

Table 5 Tools

Tool	Description
CyMCUElfTool	The build process for PSoC™ 6 devices uses the CyMCUElfTool. The tool post-processes linked ELF images to add data necessary for the boot process, perform security checking, and merge images for multiple cores into a complete image for an entire application. It also supports the PSoC™ 6 Bootloader SDK.
SMIF Configuration Tool	The SMIF Configuration Tool consists of two applications: SMIF Configuration GUI Tool and the command line SMIF Generation Tool. This tool allows you to generate input structures for the SMIF driver memslot API.
USB Configurator	The USB Configurator is also part of the ModusToolbox™ software. Use this tool to configure the USB Device descriptors. The USB Configurator generates header and source files that store the USB Device descriptors and other information used by USB Device middleware.

1.10 Supported toolchains

- PSoC™ Creator 4.4
- IAR embedded workbench for Arm® 8.32
- Keil embedded development tools for Arm® 5.25

1.11 Folder structure

The PDL is organized into several folders. The following table shows the PDL folder structure.

Table 6 Folder structures

Path\folder	Description
cmsis	CMSIS core access headers and DSP library
devices	Device header files, startup code, linker file, flash loader implementation, and the CMSIS SVD file for each device series
dfu	DFU SDK
doc	PDL and other documentation
drivers	Driver source code and headers
examples	CE217911 – PSoC™ 6 FreeRTOS Example Project
middleware	Firmware development stacks, such as Bluetooth® LE
rtos	RTOS source code supported by the PDL, such as FreeRTOS
security	Basic secure system project template
tools	User-level applications; for example, to configure a software component or to perform post-build processing
utilities	Various utility files, such as standard I/O support

1.12 Documentation

PDL user guide and API reference are in the *doc* folder of the PDL installation directory.

1.13 Technical support

For assistance, go to the Infineon support page.

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

Edition 2023-02-02

Published by

Infineon Technologies AG
81726 Munich, Germany

© 2023 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?
Email: erratum@infineon.com

Document reference
002-37026 Rev. **

Important notice
The information contained in this document is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this document must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this document.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.