Getting started with PSOC™ HV MS family

About this document

Scope and purpose

AN234115 briefly introduces you to PSOC™ high-voltage (HV) mixed signal (MS) MCU family, an Arm® Cortex®-M0+ based microcontroller. This application note also guides you on setting up the PSOC™ HV MS development environment.

Intended audience

This document is intended for anyone using PSOC™ HV MS family MCUs.

Table of contents

About this document **1**
Table of contents **2**
1 Introduction **3**
2 Feature set **4**
2.1 Function summary 5
3 Development environment and tools **9**
3.1 Evaluation board 9
3.1.1 CYHVMS-64K-56-001 board connections 10
3.1.2 Connection diagram with tools 12
3.2 Automotive Peripheral Driver Library (AutoPDL) 12
3.2.1 AutoPDL code examples release files 12
3.2.2 CPU (CM0+) startup sequence for PSOC™ HV MS 16
3.2.3 Startup flow example 17
3.2.4 Sample code 19
3.3 Peripheral drivers 21
3.4 Example usage of AutoPDL with IAR 23
3.4.1 Environment for AutoPDL 23
3.4.2 Install IAR Embedded Workbench and AutoPDL 23
3.4.2.1 Download AutoPDL package and license 23
3.4.2.2 Standard installation procedure 24
3.4.2.3 IAR EWARM flash patch configuration update 31
3.4.3 Example usage of AutoPDL 32
3.4.4 Other sample codes 41
3.5 Development tools 42
3.6 Flash programming tools 43
4 Summary **44**
5 Glossary **45**
References **47**
Revision history **48**
Disclaimer **49**

1 Introduction

This application note provides an overview of the device feature set, and describes the development environment and tools to get started with the CY8C41xxLxE-HVSxxx series from the PSOC™ HV MS family. PSOC™ HV MS is a fully integrated programmable embedded system for several automotive HMI, body, and powertrain applications. The system features an Arm® Cortex® M0+ processor with programmable and reconfigurable analog and digital blocks. PSOC™ HV MS is available in a 32-/48-/56-/64-QFN package with wettable flanks. To get a better understanding of the functionality described and terminology used in this application note, it is a good idea to read the “Section A: Overview” chapter of the architecture reference manual [3].

2 Feature set

PSOC™ HV MS (64K/128K) is a fully integrated programmable embedded system for several automotive HMI, body, and powertrain applications. The system features an Arm® Cortex®-M0+ processor, and programmable and reconfigurable analog and digital blocks.

PSOC™ HV MS devices have these characteristics:

- High-performance, 24 MHz/48 MHz Arm® Cortex-M0+® CPU with MPU and DMA controller
- 5th-generation multi-sense converter (MSC) supporting capacitive sensing (CAPSENSE™)
- 12-bit SAR ADC
- High-voltage subsystem (tolerant up to 42 V)
- Selectable integrated LIN/CXPI transceiver
- High-precision clock sources
- Configurable timer/counter/PWM block
- Configurable serial communication block (SCB) with I²C, SPI, UART, and LIN slave operating modes
- Low-power operating modes: sleep and deep sleep
- Functional safety for ASIL-B per ISO 26262
- AEC-Q100 qualified

Figure 1 and Figure 2 shows the major components of the PSOC™ HV MS-64K and MS-128K architecture.

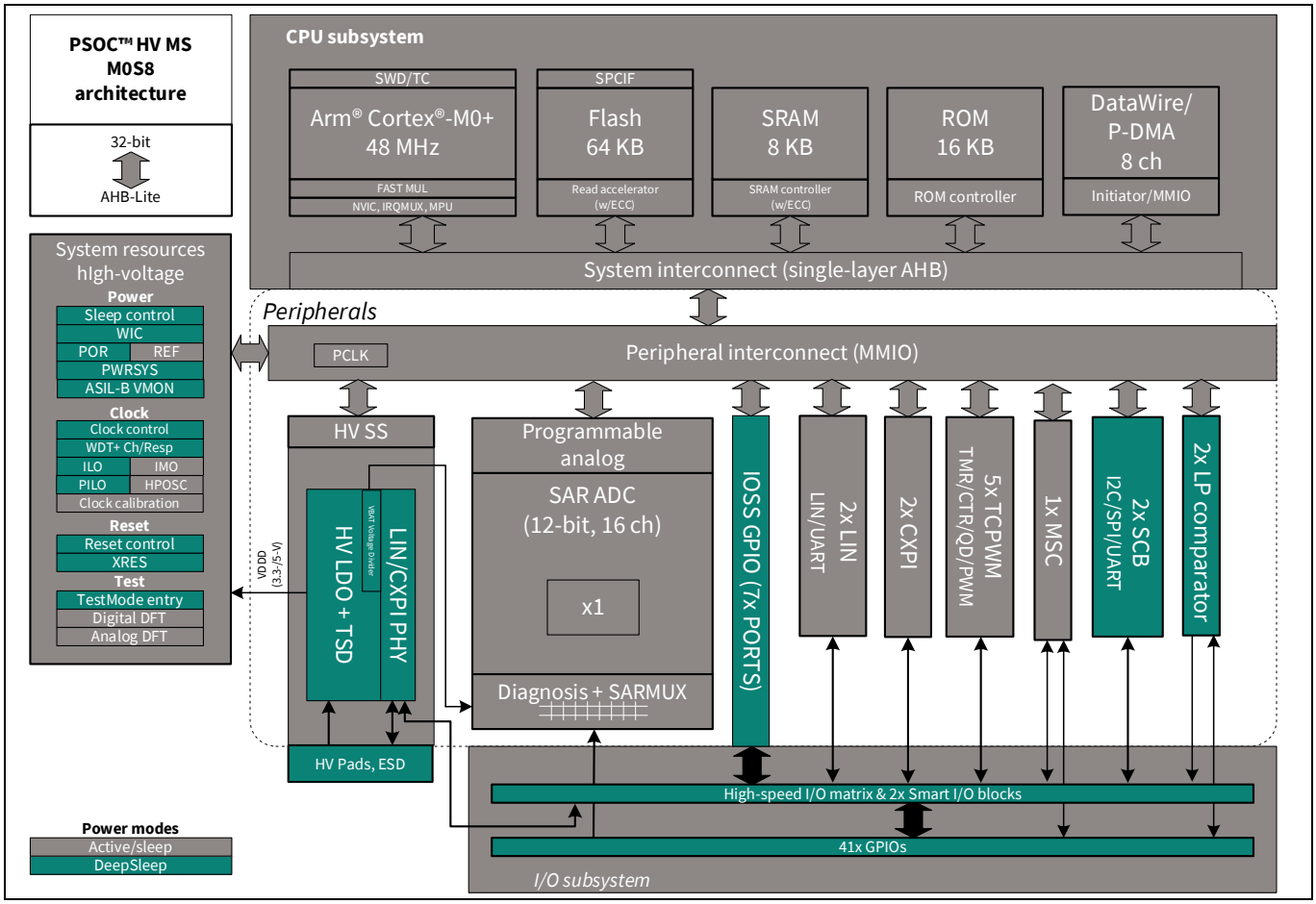

Figure 1 PSOC™ HV MS-64K block diagram

Feature set

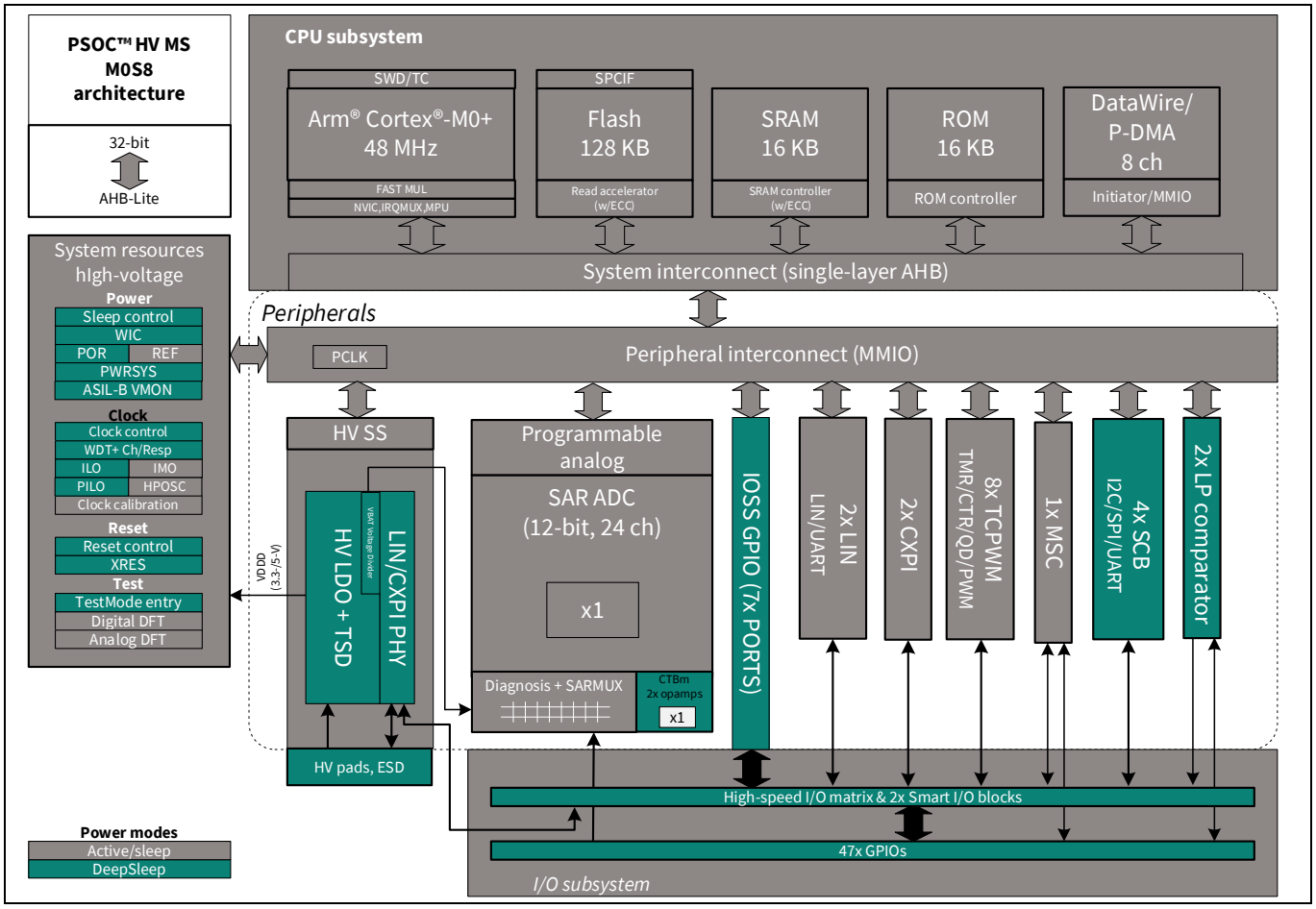

Figure 2 PSOC™ HV MS-128K block diagram

2.1 Function summary

- 32-bit CPU subsystem
  - Up to 48 MHz Arm® Cortex-M0+® CPU with MPU and DMA controller
  - Up to 64 KB (MS-64K)/128 KB (MS-128K) of code flash with ECC
  - Up to 8 KB (MS-64K)/16 KB (MS-128K) of SRAM with ECC
- Programmable analog
  - 12-bit, 1-Msps SAR ADC with differential and single-ended modes, and channel sequencer with signal averaging
  - Diagnostic multiplexer that measures various supply voltages, reference voltages, and grounds on-chip
  - Temperature sensor built into a SAR ADC
  - Low-power comparators that operate in deep sleep low-power mode
  - Two opamps (for MS-128K) with reconfigurable high-drive external and high-bandwidth internal drive, comparator modes and ADC input buffering capability. Opamps can operate in deep sleep low-power mode

Feature set

- High-voltage subsystem
  - Regulator output voltage supports 3.3 V or 5 V selectable with ±2% trimmed accuracy
  - Regulator current: up to 60 mA
  - Thermal shutdown
  - Operates directly off 12 V/24 V battery (tolerates up to 42 V)
  - Integrated Local Interconnect Network (LIN) or Clock Extension Peripheral Interface (CXPI) transceivers
- CAPSENSE™ (multi-sense converter) block
  - Multi-sense converter (MSC) provides best-in-class signal-to-noise ratio (SNR) (>5:1) and water tolerance for capacitive sensing
- Functional safety for ASIL-B
  - Development process per ISO 26262 for ASIL-B
  - Memory protection unit (MPU)
  - Window watchdog timer (WDT) with challenge-response functionality
  - Supply monitoring; detection of overvoltage and brownout events for 3.3 V/5 V, and 1.8 V supplies
  - Hardware error detection and correction with ECC: single error correction and double error detection (SECDED) on all safety-critical memories (SRAM, flash)
- Timing and pulse-width modulation
  - 16-bit timer/counter/pulse-width modulator (TCPWM) blocks (5x: MS-64K, 8x: MS-128K)
  - Center-aligned, edge, and pseudo-random modes
  - Comparator-based triggering of kill signals for motor drive and other high-reliability digital logic applications
  - Quadrature decoder
- Clock sources
  - ±2% 24 MHz to 48 MHz internal main oscillator (IMO)
  - ±1% 2 MHz high-precision oscillator (HPOSC)
  - ±5% 32 kHz precision low-power oscillator (PILO)
  - ±1% accuracy on IMO and PILO when software calibrated to the HPOSC
  - 40 kHz internal low-speed oscillator (ILO)
- Communication
  - Independent runtime reconfigurable serial communication blocks (SCBs) with reconfigurable I^2C, SPI, UART, or LIN slave functionality
  - Up to two independent LIN channels compliant with LIN 2.2A and ISO 17987 standards
  - Up to two independent CXPI channels compliant with the ISO/WD 20794 standard
- I/O
  - Up to 41(MS-64K)/47 (MS-128K) GPIOs
  - Smart I/O block that provides the ability to perform Boolean functions on port I/Os
- Package
  - 32-QFN with wettable flanks (6 x 6 mm) (for MS-64K)
  - 48-QFN with wettable flanks (7 x 7 mm)
  - 56-QFN with wettable flanks (8 x 8 mm)
  - 64-QFN with wettable flanks (9 x 9 mm) (for MS-128K)

Feature set

Table 1 shows the PSOC™ HV MS devices summary.

Table 1 PSOC™ HV MS device summary

Feature	PSOC™ HV MS-64K	PSOC™ HV MS-128K
CPU		
Core	32-bit Cortex®-M0+ CPU	32-bit Cortex®-M0+ CPU
Maximum frequency	24 MHz to 48 MHz	24 MHz to 48 MHz
DMA	8	8
Interrupt	32	32
SysTick timer	Available	Available
Fault subsystem	Available	Available
Memory		
Flash with ECC	Up to 64 KB	Up to 128 KB
SRAM with ECC	Up to 8 KB	Up to 16 KB
Programmable analog		
12-bit 1-Msps SAR ADC	1 (16-ch)	1 (24-ch)
Low-power comparator	2	2
Continuous-time block (CTBm)	Not included	1
Analog / diagnostic multiplexed bus	1	1
Temperature sensor	1	1
Special function peripheral		
CAPSENSE™ (multi-sense converter) block	1 (supported on all GPIO pins)	1 (supported on all GPIO pins)
High-voltage subsystem		
Operation voltage	3.6 V to 28 V Remains functional up to 42 V	3.6 V to 28 V Remains functional up to 42 V
High-voltage regulator	3.3 V or 5.0 V	3.3 V or 5.0 V
Selectable LIN/ CXPI transceiver	1	1
VBAT voltage divider	1	1
Thermal shutdown (TSD)	1	1
Digital system		
16-bit timer/counter/pulse-width modulator (TCPWM)	5	8
Serial communication block (I2C, SPI, UART, or LIN)	2	4
LIN channel	2	2
CXPI channel	2	2
Clock sources		
Internal main oscillator (IMO)	±2% up to 48 MHz	±2% up to 48 MHz
High-precision oscillator (HPOSC)	±1% 2 MHz	±1% 2 MHz

Feature set

Feature	PSOC™ HV MS-64K	PSOC™ HV MS-128K
Precision internal low-power oscillator (PILO)	±5% 32 kHz	±5% 32 kHz
Internal low-speed oscillator (ILO)	40 kHz	40 kHz
Functional safety for ASIL-B		
Memory protection unit (MPU)	Available	Available
Basic watchdog timer (WDT)	1	1
Challenge-response watchdog timer (CRWDT)	1	1
Lifetime counter	1	1
Power supply monitoring	Power-on-reset (POR)/ brownout detection (BOD)/ overvoltage detection	Power-on-reset (POR)/ brownout detection (BOD)/ overvoltage detection
Hardware error correction	All safety-critical memories	All safety-critical memories
System		
GPIOs	Up to 41	Up to 47
Smart I/O block	2	2
Debug interface	SWD	SWD
Package	32-QFN with wettable flanks (6 x 6 mm) 48-QFN with wettable flanks (7 x 7 mm) 56-QFN with wettable flanks (8 x 8 mm)	48-QFN with wettable flanks (7 x 7 mm) 56-QFN with wettable flanks (8 x 8 mm) 64-QFN with wettable flanks (9 x 9 mm)
Operating temperature	–40°C to +125°C	–40°C to +125°C

3 Development environment and tools

3.1 Evaluation board

The CYHVMS-64K-56-001 Evaluation Board (Figure 3), which supports 56-QFN package of PSOC™ HV MS-64K and MS-128K devices, provides access to all peripherals, resulting in a fully featured evaluation platform. CYHVMS-64K-56-001 has Arm® standard MIPI 10-pin connector for the debug interface by third-party interfaces, and Infineon KitProg3 low-level communication interface for programming and debugging. The evaluation board can be used for development of software. For more information on board details, see the PSOC™ HV MS Evaluation Board user guide [5].

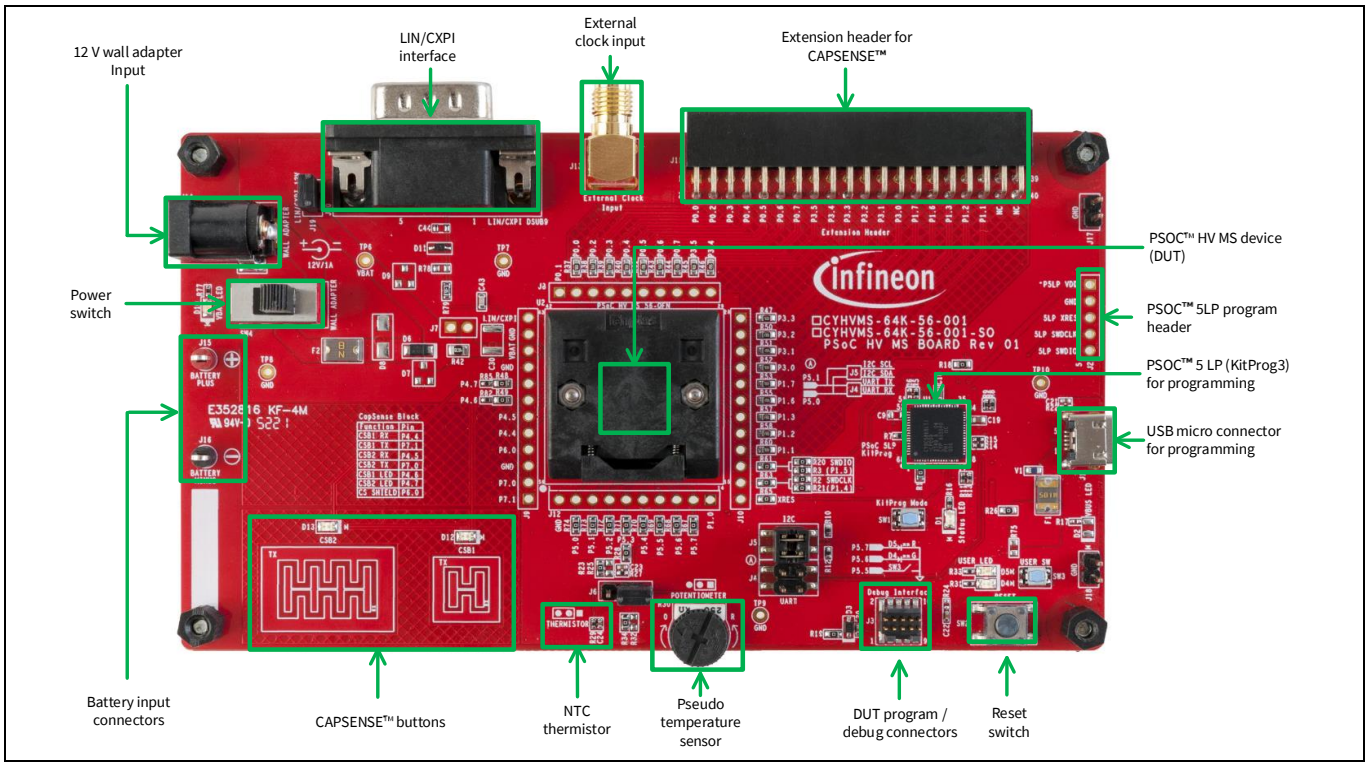

Figure 3 CYHVMS-64K-56-001 Evaluation Board – top view

Figure 4 shows the CYHVMS-64K-56-001 Evaluation Board block diagram.

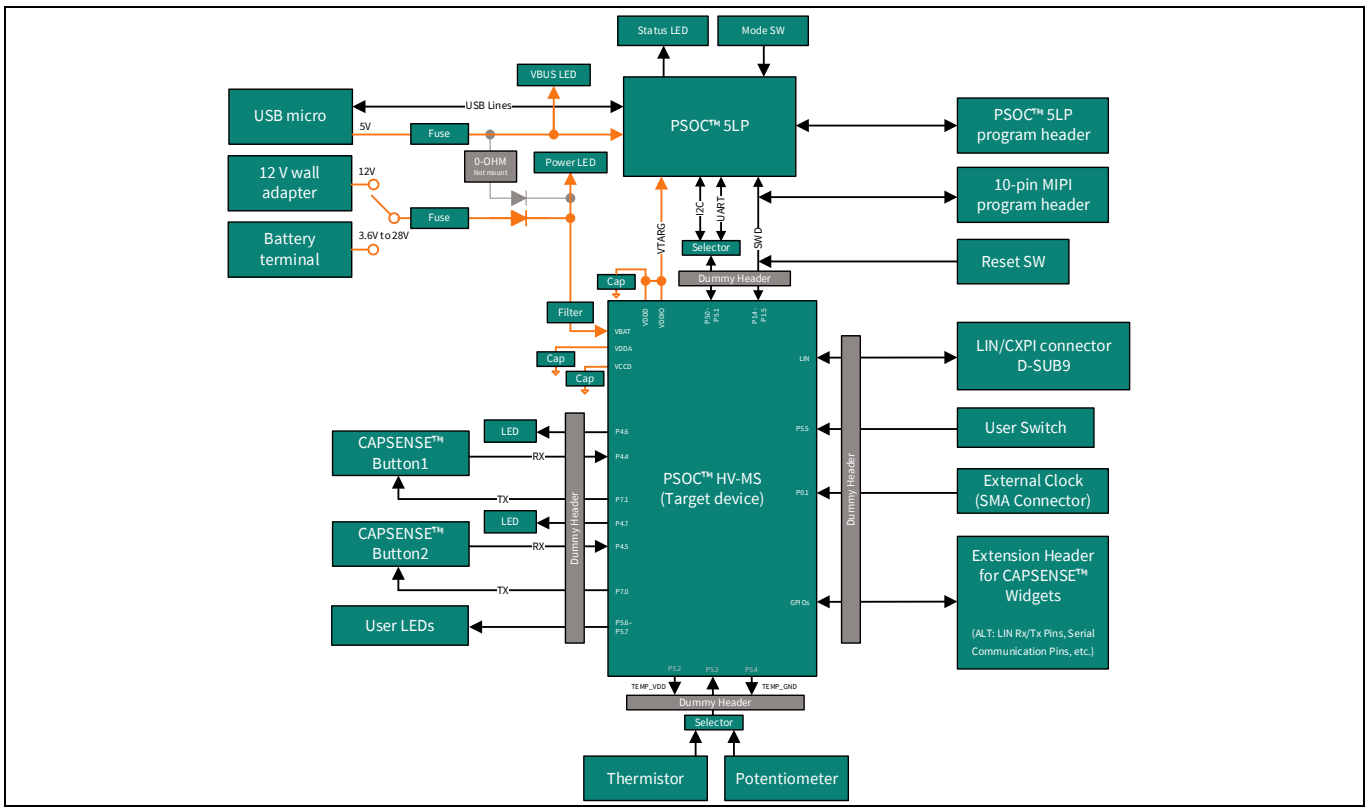

Figure 4 CYHVMS-64K-56-001 Evaluation Board block diagram

3.1.1 CYHVMS-64K-56-001 board connections

Table 2 Board connections

PSOC™ HV MS device: U2 (56-QFN package)		Connection/peripheral	Dummy header/test pin
Pin #	Pin name		
1	VDDA	Internal analog power (3.3 V or 5 V): 0.22 µF capacitor	TP5
2	VSSA	Analog GND	J9.3/9/11, J12.10
3	P7.0	CAPSENSE™ button 1 TX	J9.2
4	P7.1	CAPSENSE™ button 0 TX	J9.1
5	NC	–	–
6	P5.0	UART_TX/I2C_SDA for KitProg	J12.9
7	P5.1	UART_RX/I2C_SCL for KitProg	J12.8
8	P5.2	Thermistor VDD	J12.7
9	P5.3	Thermistor/potentiometer input	J12.6
10	P5.4	Thermistor GND	J12.5
11	P5.5	User switch	J12.4
12	P5.6	User LED 1: Green	J12.3
13	P5.7	User LED 2: Red	J12.2

Development environment and tools

PSOC™ HV MS device: U2 (56-QFN package)		Connection/peripheral	Dummy header/test pin
Pin #	Pin name		
14	P1.0	SAR ADC VREF	J12.1
15	P1.1	J11 extension header [36]	J10.9
16	VDDIO	I/O power connected to VDDD (3.3 V or 5 V): 0.1 µF + 3.3 µF capacitors	TP4
17	VSSD	Digital GND	J9.3/9/11, J12.10
18	XRES_L	Reset switch/program connector	J10.12
19	P1.2	J11 extension header [34]	J10.8
20	P1.3	J11 extension header [32]	J10.7
21	P1.4	SWD clock	J10.11
22	P1.5	SWD data	J10.10
23	P1.6	J11 extension header [30]	J10.6
24	P1.7	J11 extension header [28]	J10.5
25	P3.0	J11 extension header [26]	J10.4
26	P3.1	J11 extension header [24]	J10.3
27	P3.2	J11 extension header [22]	J10.2
28	P3.3	J11 extension header [20]	J10.1
29	P3.4	J11 extension header [18]	J8.10
30	P3.5	J11 extension header [16]	J8.9
31	P0.7	J11 extension header [14]	J8.8
32	P0.6	J11 extension header [12]	J8.7
33	P0.5	J11 extension header [10]	J8.6
34	P0.4	J11 extension header [8]	J8.5
35	P0.3	J11 extension header [6]	J8.4
36	P0.2	J11 extension header [4]	J8.3
37	P0.1	External clock input	J8.1
38	P0.0	J11 extension header [2]	J8.2
39	VDDIO	I/O power connected to VDDD (3.3 V or 5 V): 0.1 µF, 3.3 µF capacitors	TP4
40	VCCD	Internal digital power (1.8 V): 0.22-µF capacitor	TP1
41	VDDD	Internal digital power (3.3 V or 5 V): 0.1 µF, 3.3 µF capacitors	TP2
42	VSSD	Digital GND	J9.3/9/11, J12.10
43	VSSL	LIN / CXPI GND	J9.3/9/11, J12.10
44	LIN / CXPI	LIN / CXPI connector	J9.12
45	NC	–	–
46	VBAT	12-V power supply input	J9.10
47	NC	–	–

PSOC™ HV MS device: U2 (56-QFN package)		Connection/peripheral	Dummy header/test pin
Pin #	Pin name		
48	P4.7	CAPSENSE™ button 2 LED	J9.8
49	P4.6	CAPSENSE™ button 1 LED	J9.7
50	P4.5	CAPSENSE™ button 2 RX	J9.6
51	P4.4	CAPSENSE™ button 1 RX	J9.5
52	P4.3	CAPSENSE™ CMOD 4: 0.47-nF capacitor	–
53	P4.2	CAPSENSE™ CMOD 3: 0.47-nF capacitor	–
54	P4.1	CAPSENSE™ CMOD 2: 2.2-nF capacitor	–
55	P4.0	CAPSENSE™ CMOD 1: 2.2-nF capacitor	–
56	P6.0	CAPSENSE™ shield	J9.1

3.1.2 Connection diagram with tools

PSOC™ HV MS has an SWD interface to connect with third-party debuggers. The debug pin of SWD is shared with GPIO port (P1[4] and P1[5]). For more information, see *AN234116 – Hardware design guide for PSOC™ HV MS family*. Figure 5 show an example of the basic connection for PSOC™ HV MS devices. Note that the external resistors are optional.

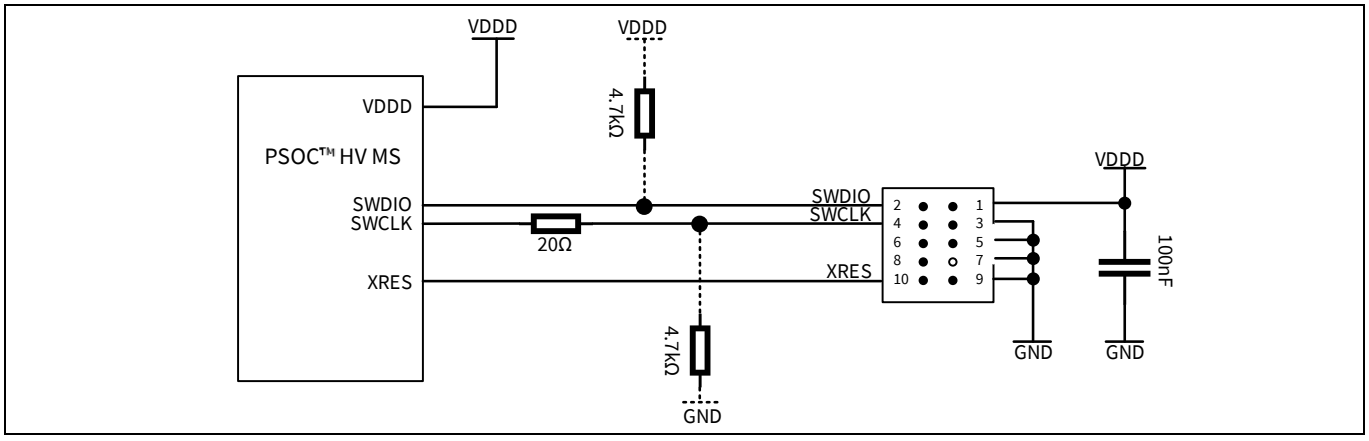

Figure 5 Basic connection diagram of MIPI 10-pin connector

3.2 Automotive Peripheral Driver Library (AutoPDL)

Automotive Peripheral Driver Library (AutoPDL) provides ISO 26262-compliant low-level drivers for all PSOC™ 4 HV peripherals.

The development environment is officially supported by AutoPDL is IAR EWARM. To use EWARM, obtain the appropriate license. See Section 3.3 for information regarding the software and license.

3.2.1 AutoPDL code examples release files

The structure of folders and files released in code examples is as follows:

```
examples
|    readme.pdf
```

```
 +---adc *
 |    +---deep_sleep
 |    |       main_cm0plus.c
 |    +---die_temp
 |    |       main_cm0plus.c
 |    +---dual
 |    |       main_cm0plus.c
 |    +---multi
 |    |       main_cm0plus.c
 |    +---single
 |    |       main_cm0plus.c
 |    \---triple
 |            main_cm0plus.c
 +---canfd *
 |    +---canfd_with_pcan
 |    |       main_cm0plus.c
 |    +---canfdNode0
 |    |       main_cm0plus.c
 |    \---canfdNode1
 |            main_cm0plus.c
+---capsense
 |    |   main_cm0plus.c
 |    |   project_config.c
 |    |   project_config.h
 |    \---usercfg
 |        |   cy_cs_cfg.h
 |        +---CSX_buttons
 |        |       CSX_buttons.csproject
 |        |       cy_cs_cfg.h
 |        +---CSX_buttons_litekit **
 |        |       CSX_buttons.csproject
 |        |       cy_cs_cfg.h
 |        +---Slider_btn
 |        |       cy_cs_cfg.h
 |        |       Slider_buttons.csproject
 |        \---Touchpad
 |                8x9_touchpad.csproject
 |                cy_cs_cfg.h
 +---cxpi
 |    +---event_trigger
 |    |       main_cm0plus.c
 |    \---polling
 |            main_cm0plus.c
 +---demos *
 |    \---ibs
 |            main_cm0plus.c
 |            PSoCHVPA IBS demo for demo 1.0.exe
 +---devices
 |       bb_bsp.h
 |       bb_bsp_psoc4hvms128k.h
 |       bb_bsp_psoc4hvms128k_litekit.h
 |       bb_bsp_psoc4hvms64k.h
 |       bb_bsp_psoc4hvms64k_litekit.h
 |       bb_bsp_psoc4hvpa144k.h
 |       bb_bsp_psoc4hvpa144k_litekit.h
```

```
+---dma
|   +---1d_transfer
|   |       main_cm0plus.c
|   \---desc_chain
|           main_cm0plus.c
+---eeprom
|       main_cm0plus.c
 +---flash
|       main_cm0plus.c
+---gpio
|   +---all_leds
|   |       main_cm0plus.c
|   \---button
|           main_cm0plus.c
+---i2c
|   +---i2c_master
|   |       main_cm0plus.c
|   \---i2c_slave
|           main_cm0plus.c
+—iso uart *
|   +---host
|   |       main_cm0plus.c
|   \---node
|           main_cm0plus.c
+---lin
|   \---slave
|           main_cm0plus.c
|           PLinMaster.linproj
+---lpcomp
|       main_cm0plus.c
+---middlewares
|   +---adc
|   |       cy_adc_configuration.c
|   |       cy_adc_configuration.h
|   |       cy_adc_counts_to_volts.c
|   |       cy_adc_counts_to_volts.h
|   |       cy_adc_die_temp.c
|   |       cy_adc_die_temp.h
|   |       cy_adc_internal.h
|   |       cy_adc_rs_open.c
|   |       cy_adc_rs_open.h
|   +---cxpi
|   |       cy_cxpi_mw.c
|   |       cy_cxpi_mw.h
|   \---uart
|           cy_uart_print.c
|           cy_uart_print.h
+---safetlib
|   +---hvms
|   |       main_cm0plus.c
|   +---hvpa *
|   |       main_cm0plus.c
|   \---spmc *
|           main_cm0plus.c
+---sar
```

```
|       main_cm0plus.c
+---spi
|   +---highLevelDriverMaster
|   |       main_cm0plus.c
|   +---highLevelDriverSlave
|   |       main_cm0plus.c
|   +---lowLevelDriverMaster
|   |       main_cm0plus.c
|   \---lowLevelDriverSlave
|           main_cm0plus.c
+---sysclk
|       main_cm0plus.c
+---sysflt
|   +---flash
|   |       main_cm0plus.c
|   \---sram
|           main_cm0plus.c
+---sysint
|   \---button_interrupt
|           main_cm0plus.c
+---syspm
|   +---deepsleep_gpio
|   |       main_cm0plus.c
|   \---sleep_gpio
|           main_cm0plus.c
+---sysreset
|       main_cm0plus.c
+---systick
|       main_cm0plus.c
+---syswdt
|   +---crwdt
|   |       main_cm0plus.c
|   +---interrupt
|   |       main_cm0plus.c
|   +---lifetime_count
|   |       main_cm0plus.c
|   +---polling
|   |       main_cm0plus.c
|   \---reset
|           main_cm0plus.c
+---tcpwm
|   +---capture
|   |       main_cm0plus.c
|   +---counter
|   |       main_cm0plus.c
|   +---pwm
|   |       main_cm0plus.c
|   +---pwm_deadtime
|   |       main_cm0plus.c
|   \---timer
|           main_cm0plus.c
+---Tools
|   +---IAR_EWARM_Patch_PSoC4HV
|   |   \---arm
|   +---psoc4hvms128k **
```

```
|   |   |   main_cm0plus.c
|   |   |   psoc4hvms128k_Eva_license.eww
|   |   |   psoc4hvms128k_flash_cm0plus_template.eww
|   |   |   PSOC4HVMS_128K.icf
|   |   |   startup.c
|   |   |   startup.h
|   |   |   startup_cm0plus.s
|   |   \---settings
|   +---psoc4hvms64k
|   |   |   main_cm0plus.c
|   |   |   psoc4hvms64k_Eva_license.eww
|   |   |   psoc4hvms64k_flash_cm0plus_template.eww
|   |   |   PSOC4HVMS_64K.icf
|   |   |   startup.c
|   |   |   startup.h
|   |   |   startup_cm0plus.s
|   |   \---settings
|   \---psoc4hvpa144k *
|       |   main_cm0plus.c
|       |   psoc4hvpa144k_Eva_license.eww
|       |   psoc4hvpa144k_flash_cm0plus_template.eww
|       |   PSOC4HVPA_144K.icf
|       |   startup.c
|       |   startup.h
|       |   startup_cm0plus.s
|       \---settings
+---trigmux
|   +---dma_uart_tx_rx
|   |       main_cm0plus.c
|   \---tcpwm_dma
|           main_cm0plus.c
\---uart
        main_cm0plus.c
```

Note: ***** *NOT available for PSOC™ HV MS.*

****** *NOT available for PSOC™ HV MS-64K.*

3.2.2 CPU (CM0+) startup sequence for PSOC™ HV MS

The following steps are involved in the device startup sequence:

1. System reset (@0x0000 0000)
2. CM0+ in SROM/boot:
   a) Device in privilege mode
   b) Perform trimming of the device based on the 64-bit key
   c) SWD initialized
   d) APIs executed from Arm® CM0+ non-maskable interrupt (NMI)
   e) APIs for flash program, erase, and read
   f) APIs to retrieve data from privilege registers
   g) Transfer control to the user program

h) Device in user mode

3. CM0+ executes the user application from the code flash region

The following steps are involved in the application startup sequence:

4. System reset (@0x0000 0000)
5. CM0+ executes “SystemInit”
   a) Disables WDT
   b) Applies flash wait states
   c) Clock selection
6. Enables generic system IRQ
7. Continues execution of the CM0+ user application

3.2.3 Startup flow example

The application startup sequence process is executed after the boot process completes. In the boot process, according to the lifecycle stage, protection status and debug access restriction setting are mainly performed, and application authentication is performed. In the application startup sequence process, initial settings necessary for program execution such as disabling the WDT, applying flash wait state, and clock settings are completed, and then execution jumps to the main routine. CM0+ executes the boot code after reset.

Figure 6 describes the PSOC™ HV MS device startup flow.

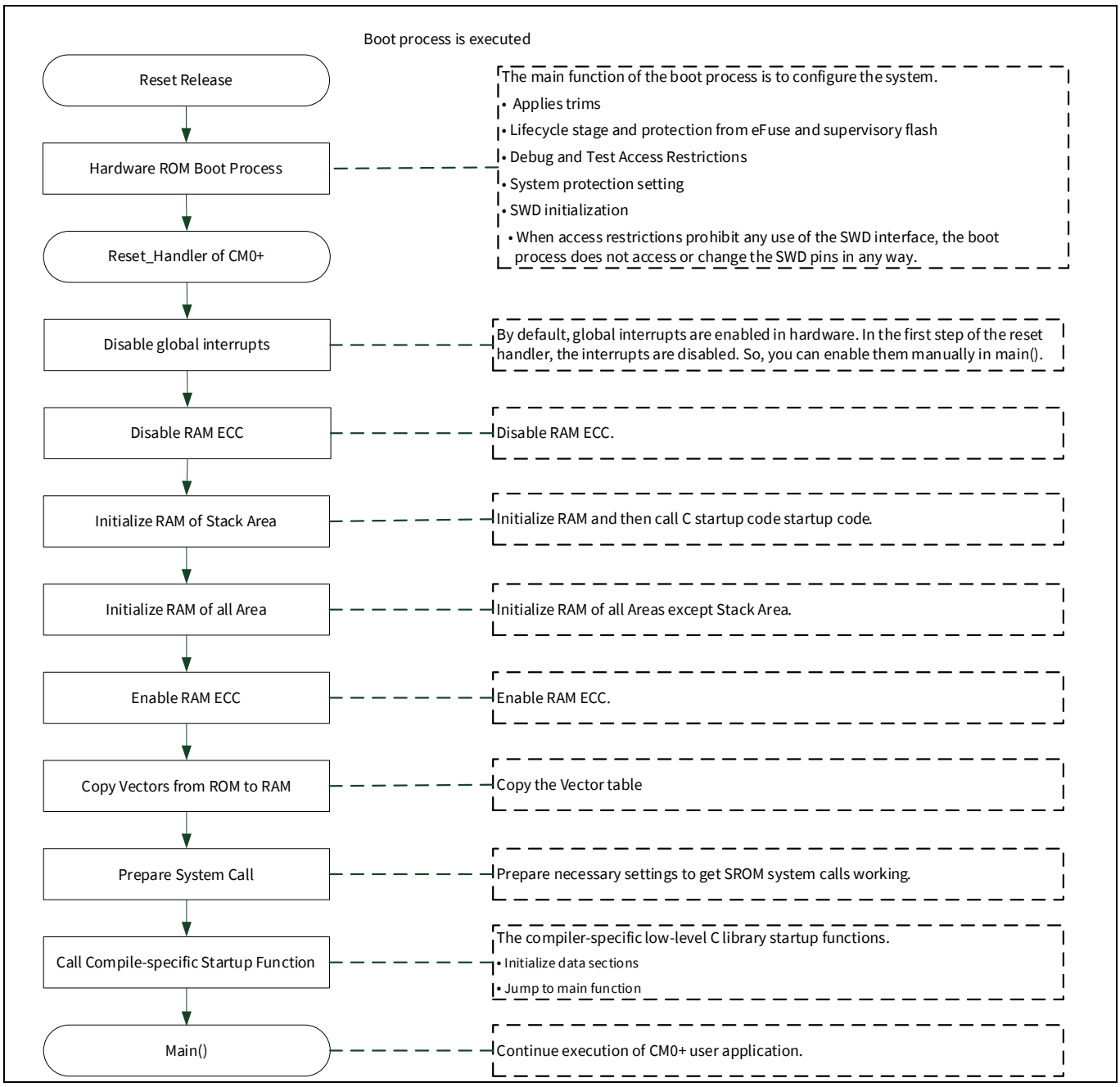

Figure 6 Device startup flow

Note: See architecture reference manual [3] for more details on lifecycle stages and protection bits.

3.2.4 Sample code

Code Listing 1 shows a sample code for CM0+ startup and Code Listing 2 shows a sample code for CM0+ main function.

Code Listing 1 Sample code for CM0+ startup

```
;*******************************************************************************
;* Start-up Code
;*******************************************************************************
        THUMB
        PUBWEAK Reset_Handler
        SECTION .text:CODE:REORDER:NOROOT(2)
Reset_Handler:

; Disable global interrupts
        CPSID   I

; Update Vector Table Offset Register with address of user ROM table
; (will be updated later to user RAM table address in C startup code)
        LDR  r0, =__vector_table
        LDR  r1, =VTOR
        STR  r0, [r1]
        DSB

; Initialize ECC of startup stack (needed for local variables in C startup code) by
processing 8 bytes per loop iteration.
; Prerequisite: Stack Pointer (SP) has not been modified (from the vector table
init value) by above code (otherwise code must be adapted)
        MOVS r0, #0 ; clear value
        MOVS r1, #0 ; clear value
        LDR  r2, Cy_u32StartupStackStartAddress
startup_stack_ecc_init_loop:
        STM  r2!, {r0, r1}
        CMP  r2, sp
        BNE  startup_stack_ecc_init_loop

; Call C startup code (no ANSI C context established yet!)
        LDR  r7, =Startup_Init
        BLX  r7

; Initialize data sections
        LDR  r7, =__iar_data_init3
```

Code Listing 1 Sample code for CM0+ startup

```
        BLX   r7

        LDR   r7, =__iar_program_start
        BLX   r7

; Note: Control flow does not necessarily return here.
; On some tool-chains (e.g. IAR) control flow will never return from
; the system library.
Cy_Main_Exited:
        B     Cy_Main_Exited
```

Code Listing 2 Sample code for CM0+ main function

```
#include "intrinsics.h" /* To use IAR embedded function */
#include "startup.h"
#include "bb_bsp.h"

#include "cy_common_def.h"
#include "cy_error_callout.h"
#include "cy_gpio.h"
#include "cy_sysclk.h"
#include "cy_syslib.h"

int main(void)
{
    uint8 flag = 1;
    SystemInit();
    __enable_interrupt(); /* Enable global interrupts. */

    /* Initialize USER LED1 port/pin */
    Cy_Gpio_SetPinMode(CY_USER_LED1_PORT_IDX, CY_USER_LED1_PIN, CY_USER_LED1_MUX);
    Cy_Gpio_SetDriveMode(CY_USER_LED1_PORT_IDX, CY_USER_LED1_PIN,
CY_GPIO_DRIVE_MODE_STRONG);

    /* Initialize USER LED2 port/pin */
    Cy_Gpio_SetPinMode(CY_USER_LED2_PORT_IDX, CY_USER_LED2_PIN, CY_USER_LED2_MUX);
```

Code Listing 2 Sample code for CM0+ main function

```
    Cy_Gpio_SetDriveMode(CY_USER_LED2_PORT_IDX, CY_USER_LED2_PIN,
CY_GPIO_DRIVE_MODE_STRONG);
    for(;;)
    {   // Wait 0.05 [s]
        Cy_SysLib_DelayUs(50000);
        switch(flag)
        {
        case 1:
          Cy_Gpio_Inv(CY_USER_LED1_PORT_IDX, CY_USER_LED1_PIN);
          flag = 2;
          break;
        case 2:
          Cy_Gpio_Inv(CY_USER_LED2_PORT_IDX, CY_USER_LED2_PIN);
          flag = 1;
          break;
        default:
          Cy_Gpio_Inv(CY_USER_LED1_PORT_IDX, CY_USER_LED1_PIN);
          break;
        }
    }
}
```

3.3 Peripheral drivers

Peripheral drivers are a set of firmware drivers that provide APIs for accessing hardware. These APIs perform initialization and control activities of each peripheral.

Table 3 lists the peripheral drivers that Infineon provides.

Table 3 Peripheral drivers

No.	Driver	Description	API functionality
1	CAPSENSE™	Capacitive sensing	Provides capacitive sensing functionality
2	CXPI	CXPI driver	Provides CXPI function
3	DMA	Direct memory access	Performs memory-to-memory (M-DMA) and peripheral-to-memory (P-DMA) (and vice versa) operations
4	EEPROM	Electrically Erasable Programmable Read-Only Memory	Provides APIs to read and write EEPROM
5	FLASH	Flash	APIs to handle flash memory

No.	Driver	Description	API functionality
6	GPIO	General-purpose I/O ports	Configures and access device input/output pins
7	I2C	Inter-Integrated Circuit	provides master and slave data transfer
8	LIN	Local Interconnect Network (LIN)	Provides master and slave data transfer capabilities
9	LPCOMP	Low-power comparators	Provides access to the low-power comparators
10	SafeTLib	Safety library	Provides safety-related functionality
11	SAR	Successive Approximation Register	Provides analog-to-digital conversion functionality
12	SPI	Serial Peripheral Interface	Provides master and slave data transfer
13	SYSCLK	System clock	Provides APIs to control and read status of various clocking capabilities of the device
14	SYSFLT	System fault	Configures the fault reporting structure
15	SYSINT	System interrupt	Manages interrupts and exceptions, in conjunction with the CMSIS core NVIC API
16	SYSPM	System power modes	Controls device power modes
17	SYSRESET	System reset	Provides APIs for reading reset reason and clearing them
18	SYSTICK	System Tick timer	Manages a 24-bit down-counter timer
19	SYSWDT	Free-running watchdog timer	Provides control and status capabilities
20	TCPWM	Timer counter PWM	Manages a 16- or 32-bit periodic counter, PWM, quadrature decoder, shift register
21	TRIGMUX	Trigger multiplexer	Manages the multiplexing of trigger outputs to specific trigger inputs across multiple peripherals
22	UART	Universal Asynchronous Receiver-Transmitter	Provides serial communication

Table 4 Device-specific middleware

No.	Middleware	Description	API functionality
1	UART	UART COM configuration	UART configuration for ADC example projects
2	CXPI	CXPI configuration	CXPI configuration for CXPI example projects

3.4 Example usage of AutoPDL with IAR

3.4.1 Environment for AutoPDL

This section describes the operation with PSOC™ HV MS devices with AutoPDL as an example.

AutoPDL operational conditions:

- CPU: PSOC™ HV MS devices (56-QFN)
- Target board: CYHVMS-64K-56-001 Evaluation Board
- IDE: IAR Embedded Workbench for Arm® 8.50.10
- ICE: IAR I-jet Debugger or onboard KitProg3

3.4.2 Install IAR Embedded Workbench and AutoPDL

Install IAR Embedded Workbench 8.50.10 in advance and ensure that the licenses are properly enabled.

The following sections provide information on the installation procedures and licensing details for AutoPDL.

3.4.2.1 Download AutoPDL package and license

Follow these steps to download AutoPDL and get the License.

1. Go to Infineon Developer Center
2. Under **Software**, find **PSOC™ 4 HV Family AutoPDL and middleware** and click **Request** to get access to the AutoPDL software

   Once the order request is approved, you will receive an email from Infineon Developer Center (**R-IFX-DeveloperCent@infineon.com**). Product installer key (**License Key**) and a **link to download** the software are included in the email

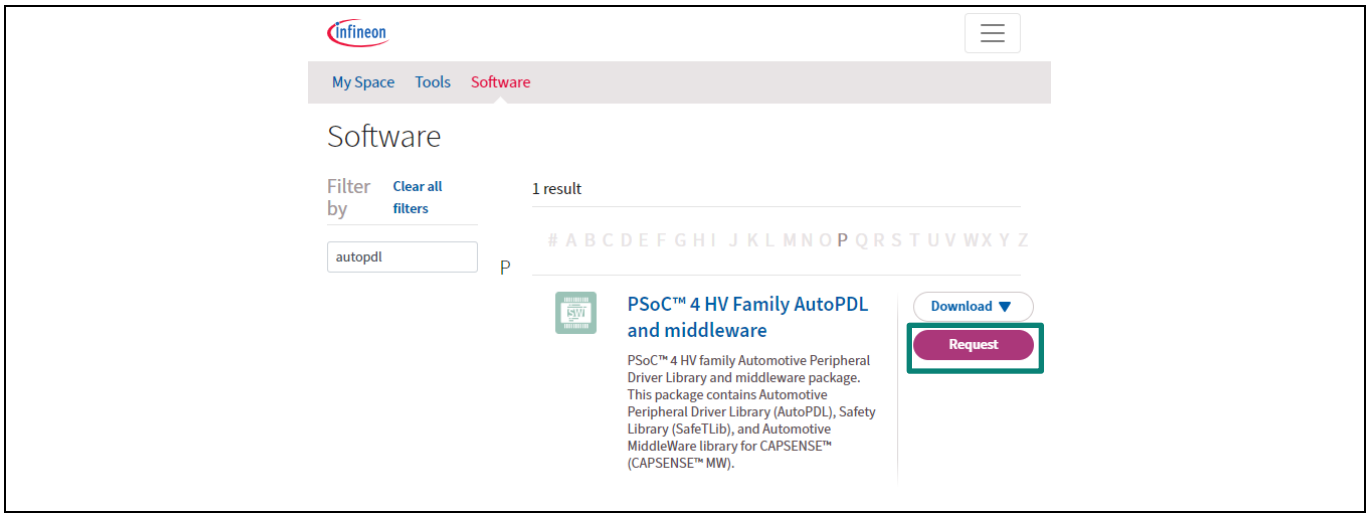

Figure 7 Requesting PSOC™ 4 AutoPDL and middleware from IDC

3.4.2.2 Standard installation procedure

After you download the software and extract the archive, follow the instructions to install AutoPDL.

AutoPDL Middleware package

1. Run the *PSOC4HV_AutoPDL-MW_V1.x.x_BulkInstall.bat* in file the *\WWxxxx_PSOC4HV_AutoPDL-MW_Vx.x.x* folder. After the setup wizard appears, click **Next**

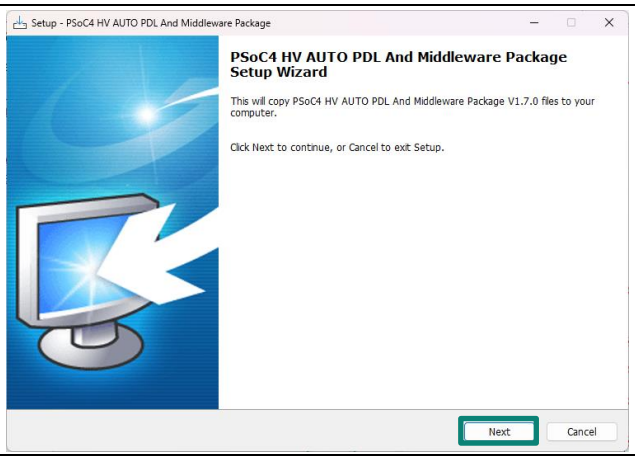

Figure 8 Setup wizard

2. Enter the **License key** which is included in the email from Infineon Developer Center and click **Next**

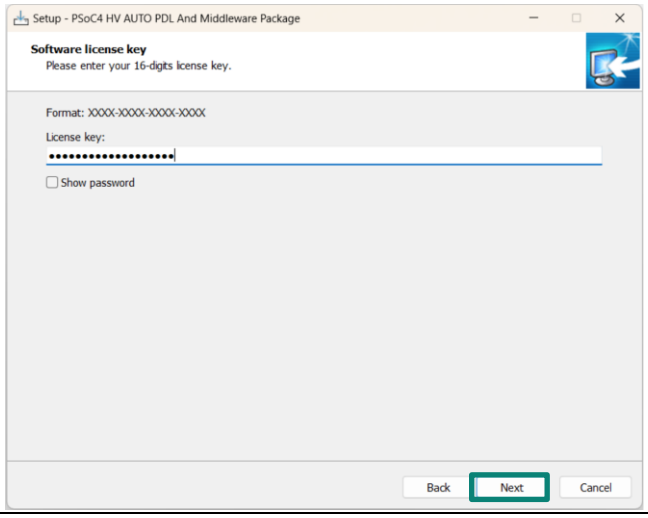

Figure 9 Adding license key

3. Read the license agreement. Select **I accept the agreement** and click **Next**

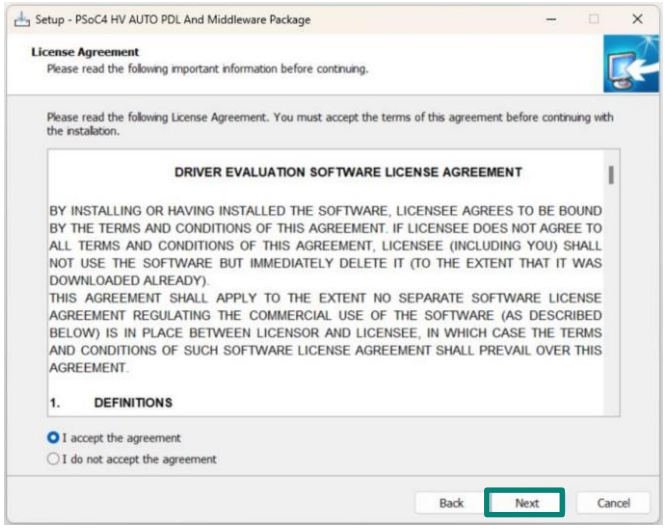

Figure 10 Accepting license agreement

4. Read the information and click **Next**

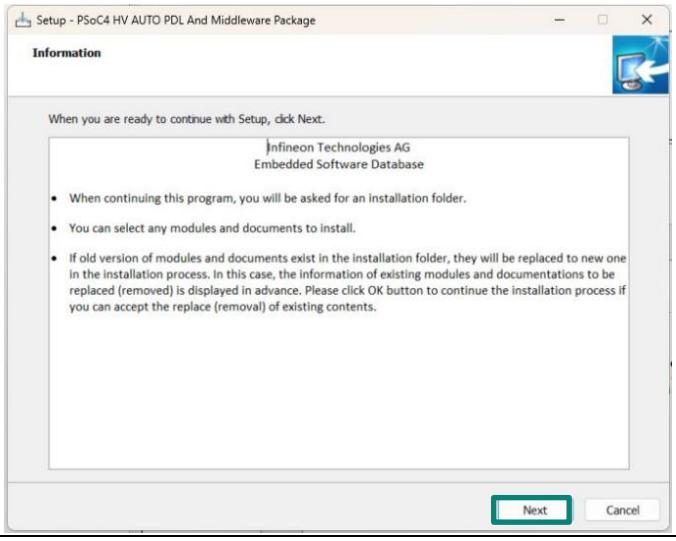

Figure 11 Reading the information for package

5. Select the install folder and click **Next** to continue

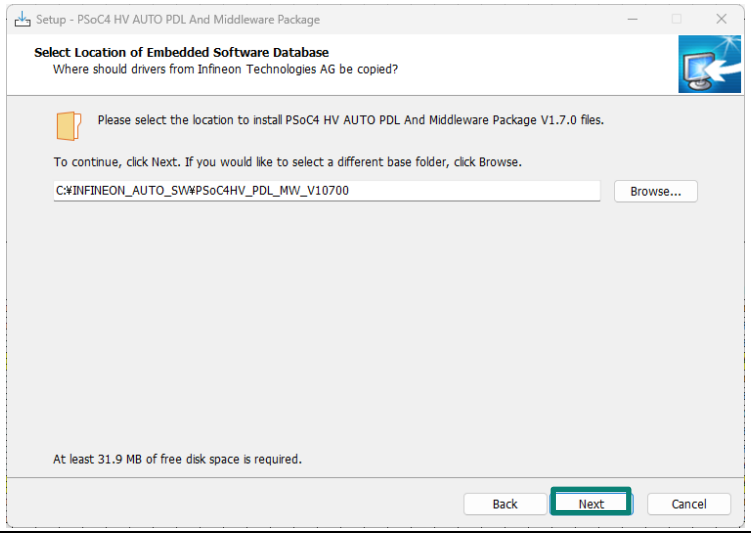

Figure 12 Selecting the installation folder

6. Select the components that you want to install and click **Next** to continue

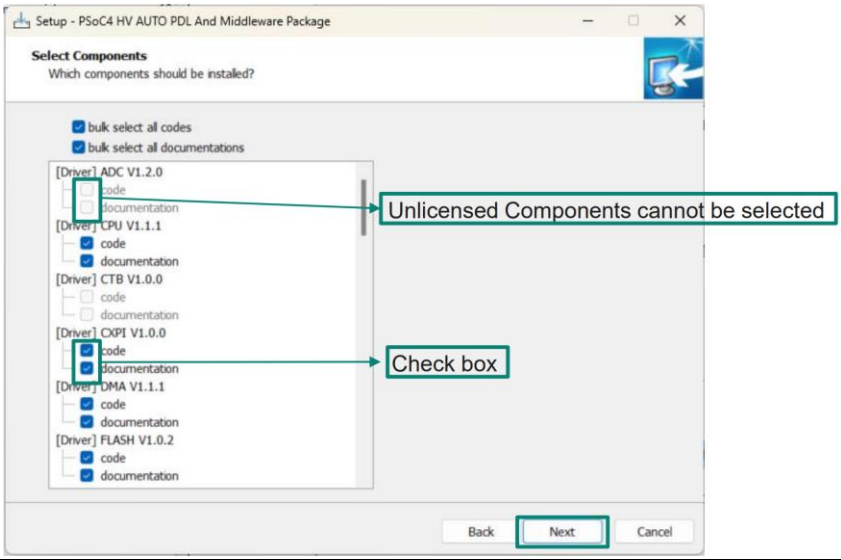

Figure 13 Selecting components

7. The AutoPDL and middleware package installer checks for the existing AutoPDL drivers/middleware and documents in the installation folder. The existing drivers/middleware and documents are displayed. If you want to replace the existing contents, click **OK**

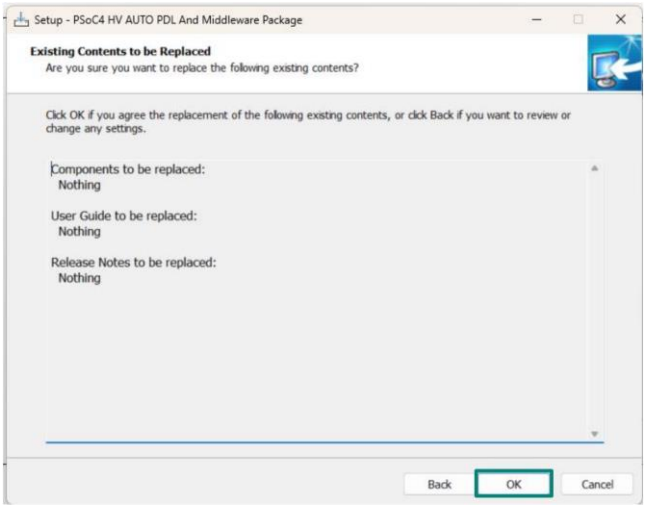

Figure 14 Replacing the existing contents

8. Confirm the details and click **Install**

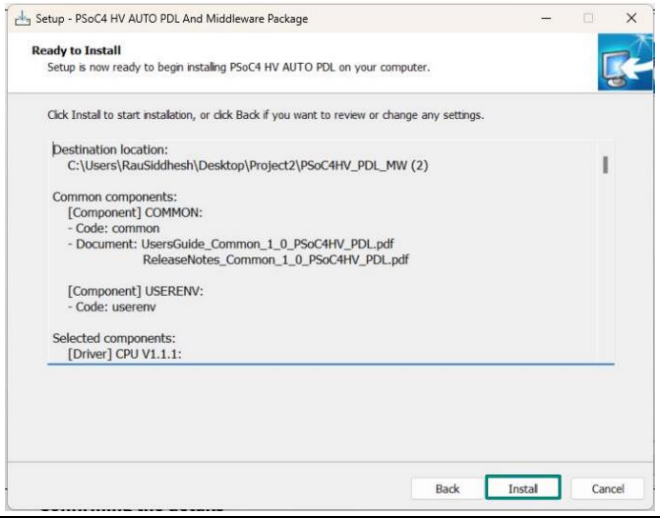

Figure 15 Installing AutoPDL and middleware

9. Read the Precautions for Use and click **Next**

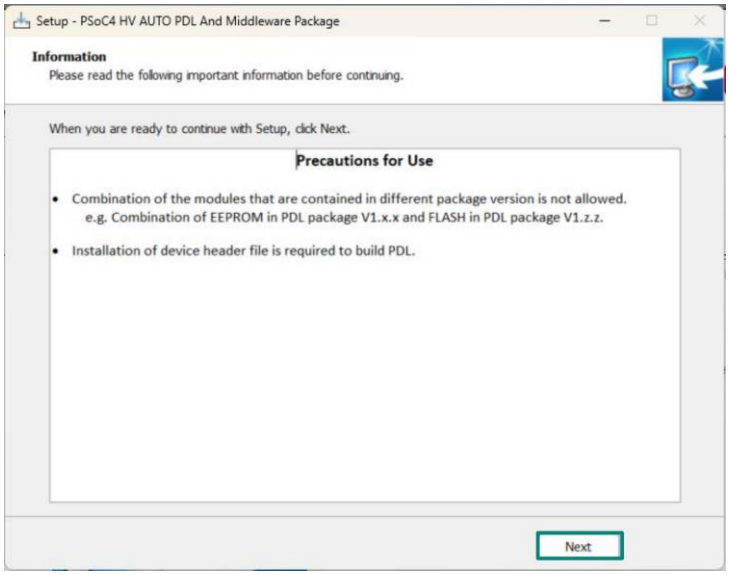

Figure 16 Reading the precautions for using the package

10. Click **Finish** to complete the installation

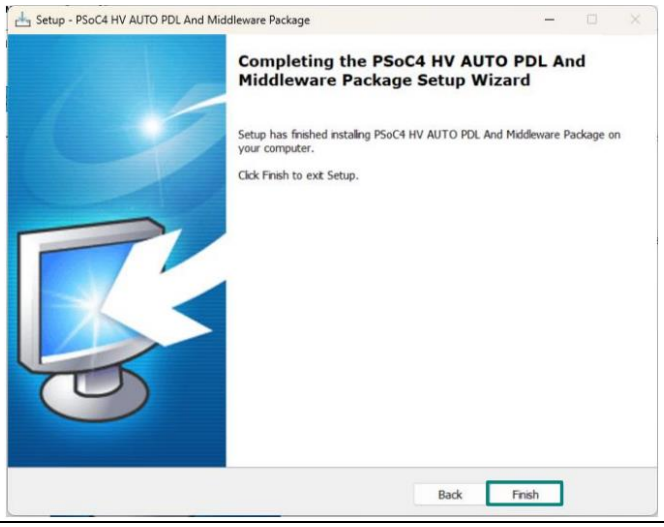

Figure 17 Finishing the installation

Device header files

1. The setup wizard for device header files appears automatically
2. In the setup wizard, click **Next**
3. Read the license agreement. Select **I accept the agreement** and click **Next**
4. Read the information and click **Next**
5. Select the install folder as the same as selected while installing the middleware. Click **Next** to continue

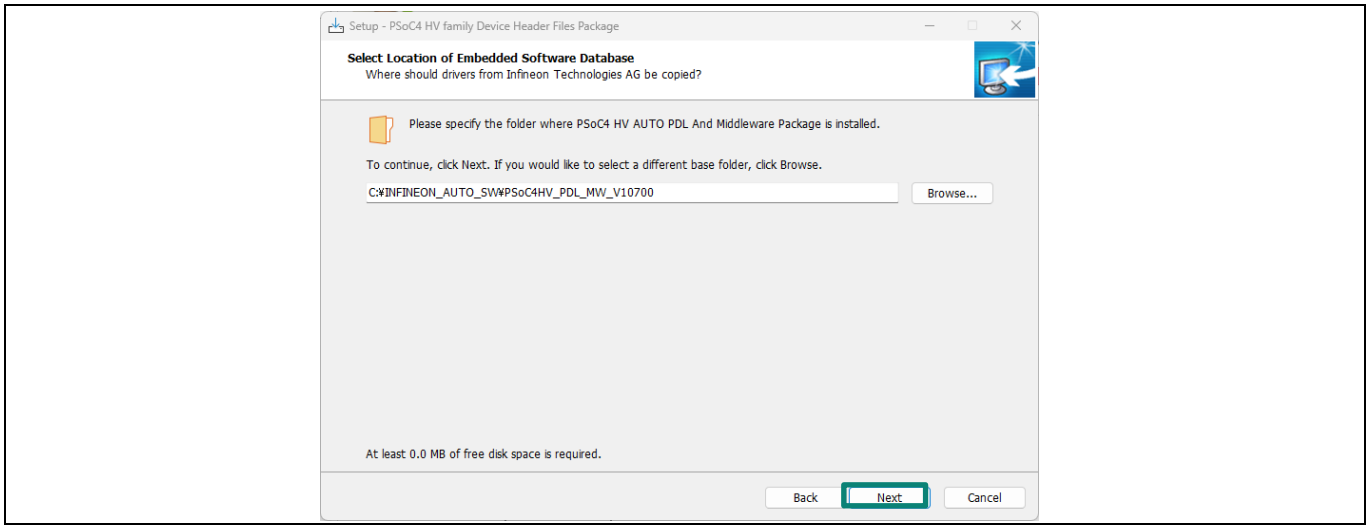

Figure 18 Selecting the installation folder

6. Select the PSOC™ HV device for which you want to install headers and click **Next** to continue

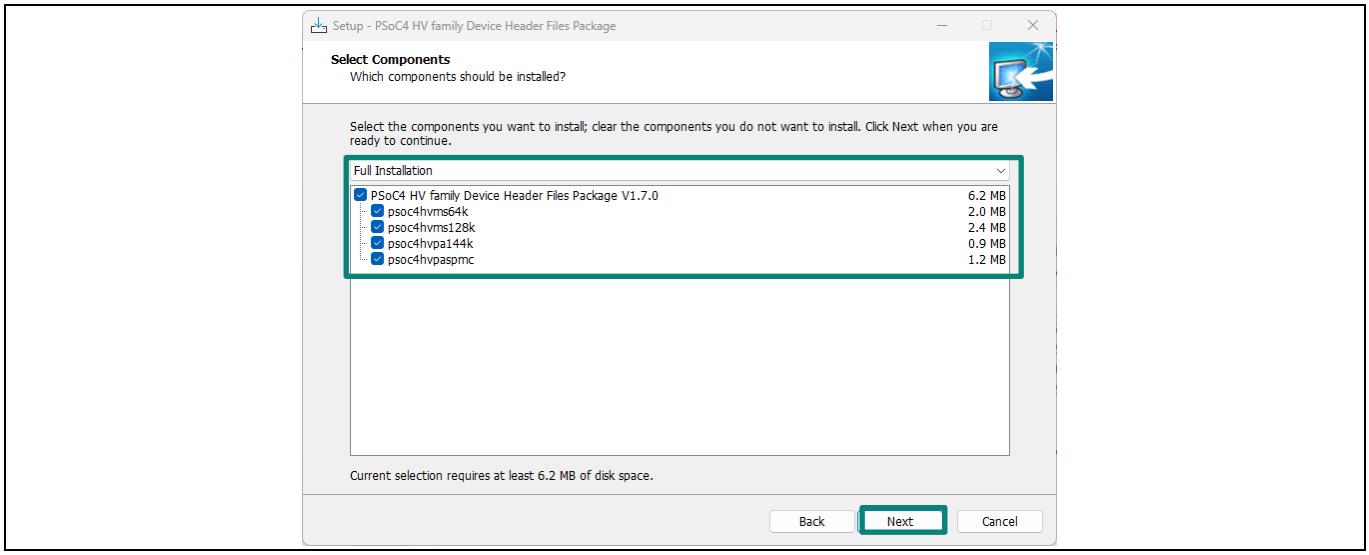

Figure 19 Selecting headers

7. Check your selection and click on **install**

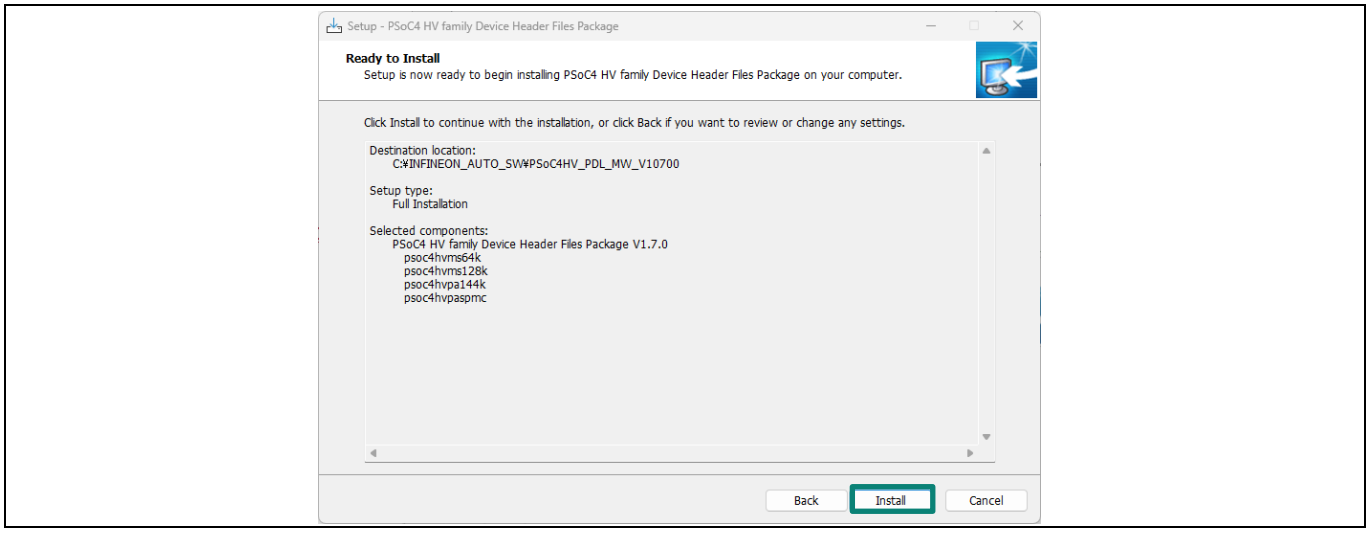

Figure 20 Installing the package

8. Click **Finish** to complete the installation

Code examples

The setup wizard for code examples automatically appears.

1. In the setup wizard, click **Next**
2. Read the license agreement. Select **I accept the agreement** and click **Next**
3. Read the information and click **Next**
4. Select the install folder as the same as selected while installing the middleware and device header. Click **Next** to continue

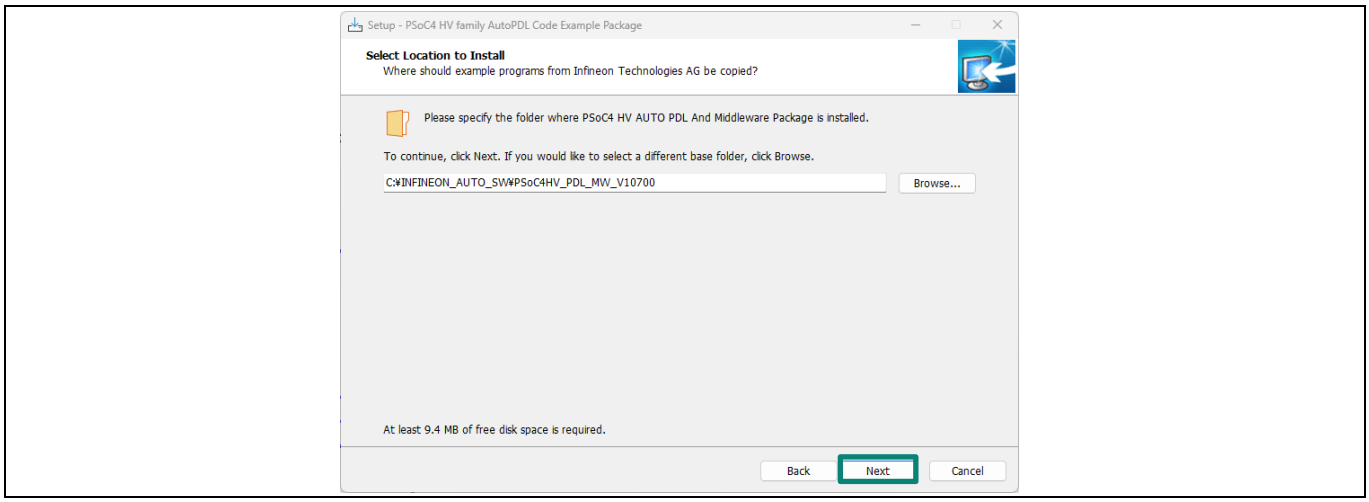

Figure 21 Selecting the installation folder

5. Select the PSOC™ HV device for which you want to install code example from the drop down as shown in the following figure. Click **Next** to continue

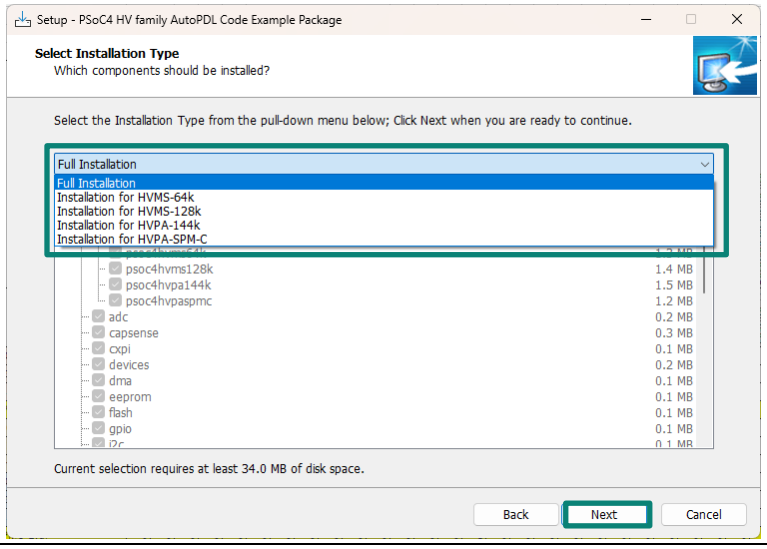

Figure 22 Selecting the required code example

6. Check your selection and click **Install**

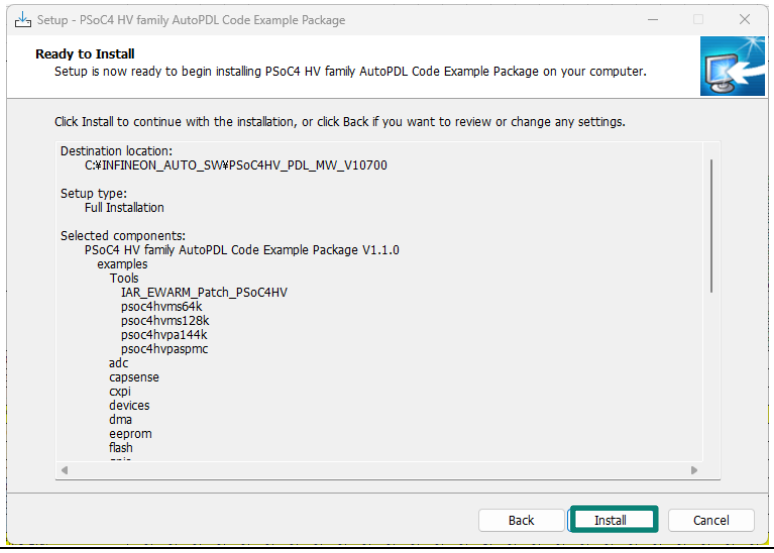

Figure 23 Ready for installation

7. Click **Finish** to complete the installation

3.4.2.3 IAR EWARM flash patch configuration update

Copy and paste (or replace) the flash loader files (in the *arm* folder) to the IAR installation folder.

1. Copy from AutoPDL installation:
   - *C:\INFINEON_AUTO_SW\PSOC4HV_PDL_MW_Vxxxxx\examples\Tools\IAR_EWARM_Patch_PSoC4HV*
2. Paste (or replace) to IAR installation:
   - *C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.4 EWARM FS 8.50.10*

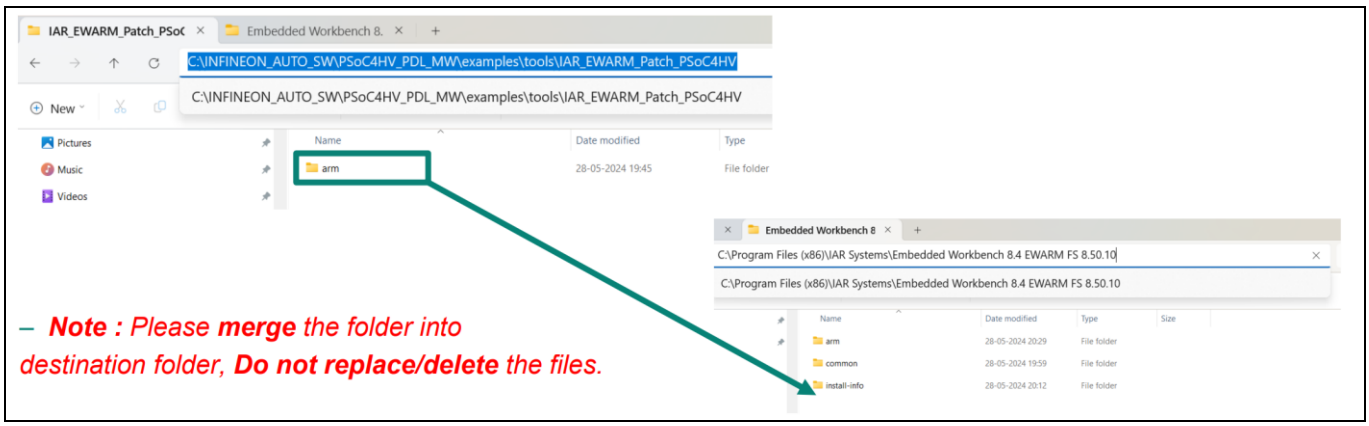

Figure 24 Copying the folder

3.4.3 Example usage of AutoPDL

The following steps are for PSOC™ HV MS-64K. In case of using PSOC™ HV MS-128K, adjust the number and follow the same step.

1. Navigate to the AutoPDL folder shown below and open the *psoc4hvms64k_Eva_license.eww* file:
   C:\INFINEON_AUTO_SW\PSOC4HV_PDL_MW_Vxxxxx\examples\Tools\psoc4hvms64k

Note: If you use the source code license, open the psoc4hvms64k_flash_cm0plus_template.eww *file.*

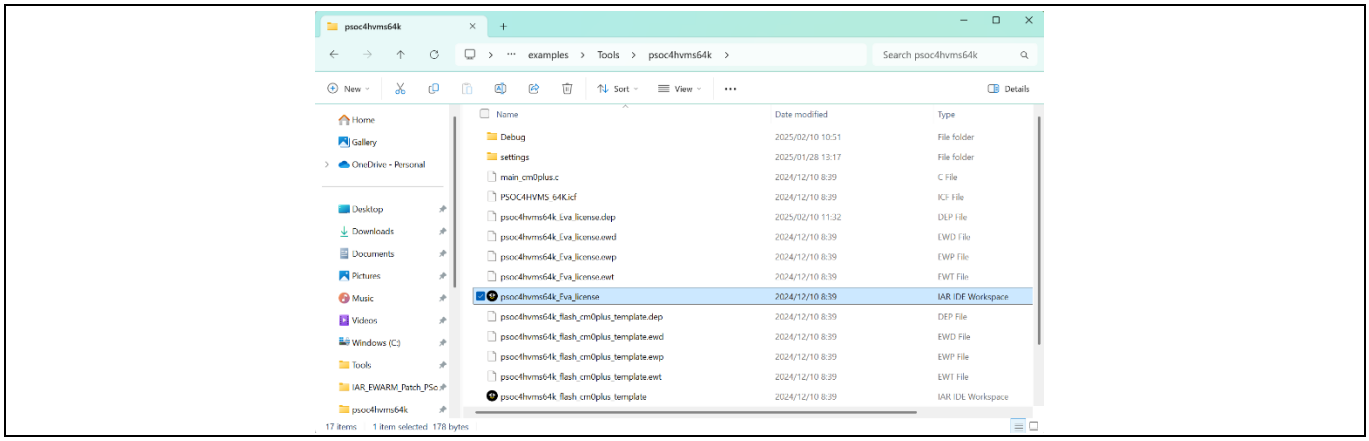

Figure 25 Opening the license file

2. Double-click on *main_cm0plus.c* in the left panel. Details of the sample code are shown in Figure 26

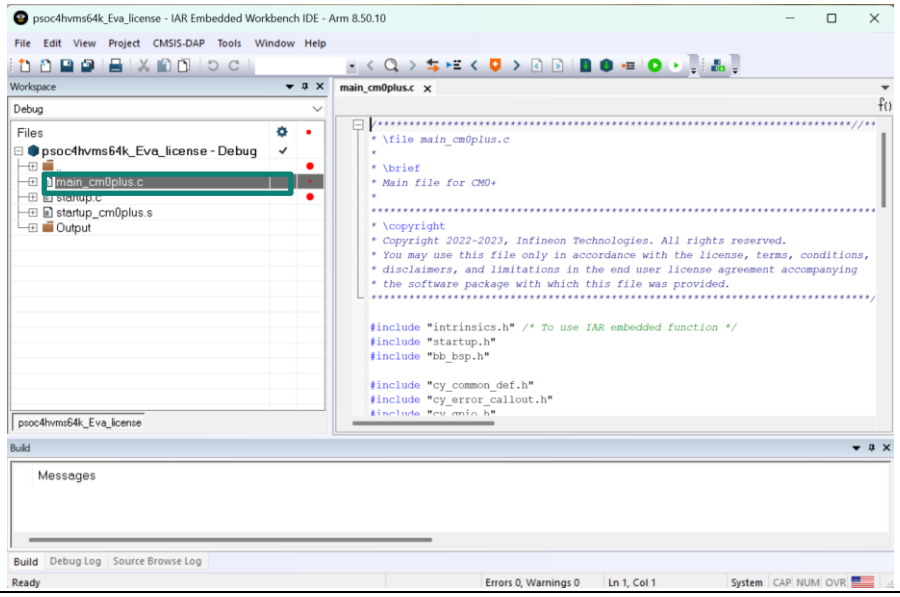

Figure 26 Sample code for main_cm0plus.c

3. Write the define value for your target device:

   a) Open *C:\INFINEON_AUTO_SW\PSoC4HV_PDL_Vxxxxx\source\userenv\cy_pdl_config.h*

   The device names supported by the standard evaluation board are as follows:

   `HVPA-144K: CY8C4147LCE_HV423` (evaluation board) or `CY8C4147LCE_HV413` (lite-kit board)

   `HVMS-64K: CY8C4146LWE_HVS115`

   `HVMS-128K: CY8C4147LWE_HVS135X`

   `HVPA-SPM-C: CY8C4147LDE_HV734`

   b) Select and set the define value that matches your board (**CY8C4146LWE_HVS115** or **CY8C4147LWE_HVS135X**)

   If you use the lite-kit board, also **add** the following definitions:

   `HVPA-144K_LITE: KIT_PSOC4_HVPA_144K_LITE`

   `HVMS-64K_LITE: KIT_PSOC4_HVMS_64K_LITE`

   `HVMS-128K_LITE: KIT_PSOC4_HVMS_128K_LITE`

   For example, if you are using **HVMS-64K_LITE**, change ***cy_pdl_config.h*** as follows:

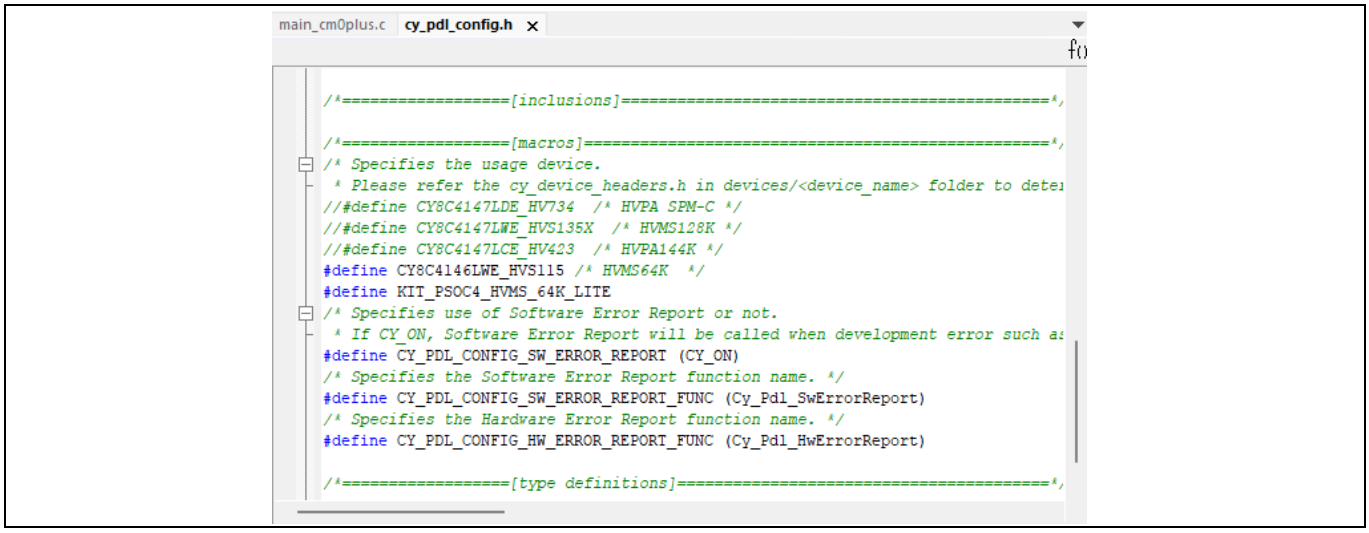

Figure 27 Changing cy_pdl_config.h

4. Select **Project** > **Rebuild All** to build the project

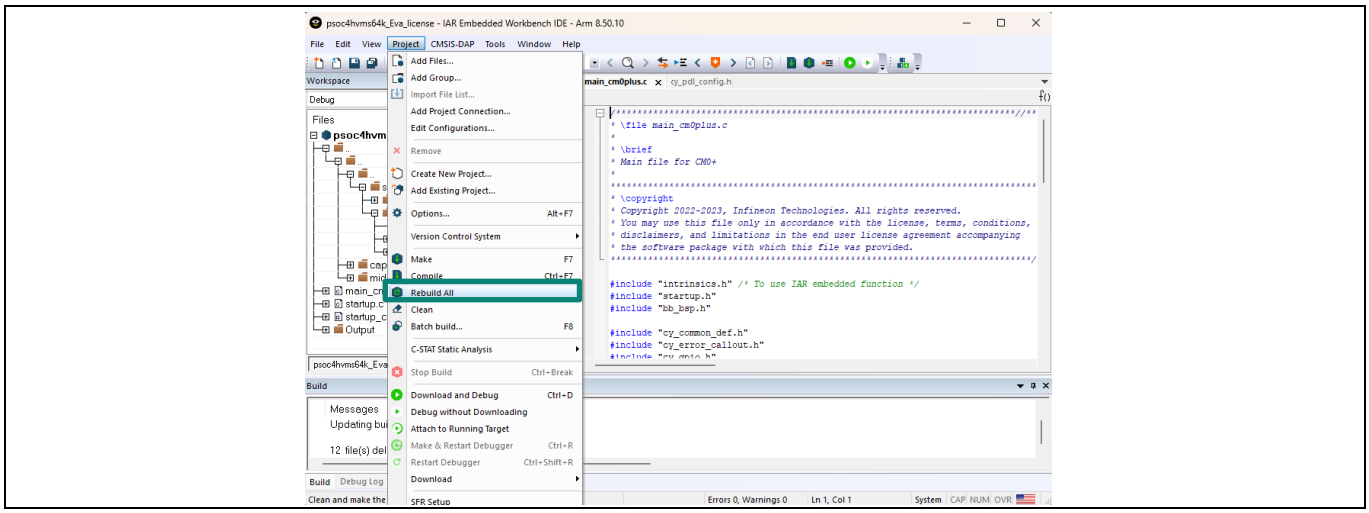

Figure 28 Rebuilding the project

Note: If you did not request a SafeTlib license, two errors will occur. If you see the errors, remove the cy_pstl_user_callout.c *and* libpstl.a *files and then select* ***Rebuild all****. To remove the* cy_pstl_user_callout.c *file, right-click the file and then click* ***Remove****. To remove the* libpstl.a *file, navigate to* ***Project*** *>* ***Options*** *>* ***Linker*** *>* ***Library*** *tab and click the ellipses button next to the* ***Additional libraries*** *pane. Select* libpstl.a *and click remove (×).*

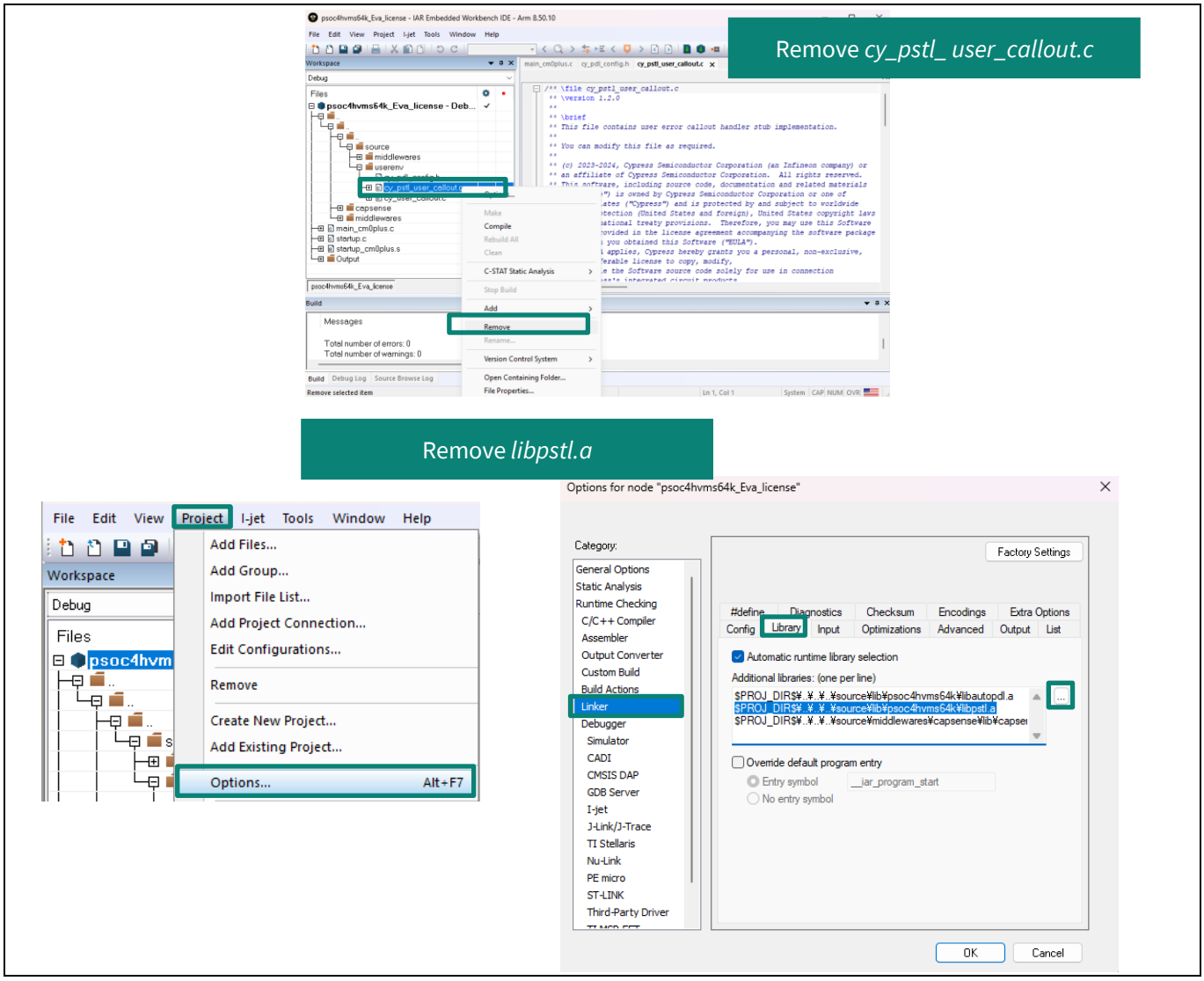

Figure 29 Fixing SafeTlib license errors

5. Confirm the initial jumper and switch settings (See Figure 30):
   - SW4 – Battery side
   - J19 – Short
   - J6 – Right side (potentiometer)
   - J5 – Short (I2C)

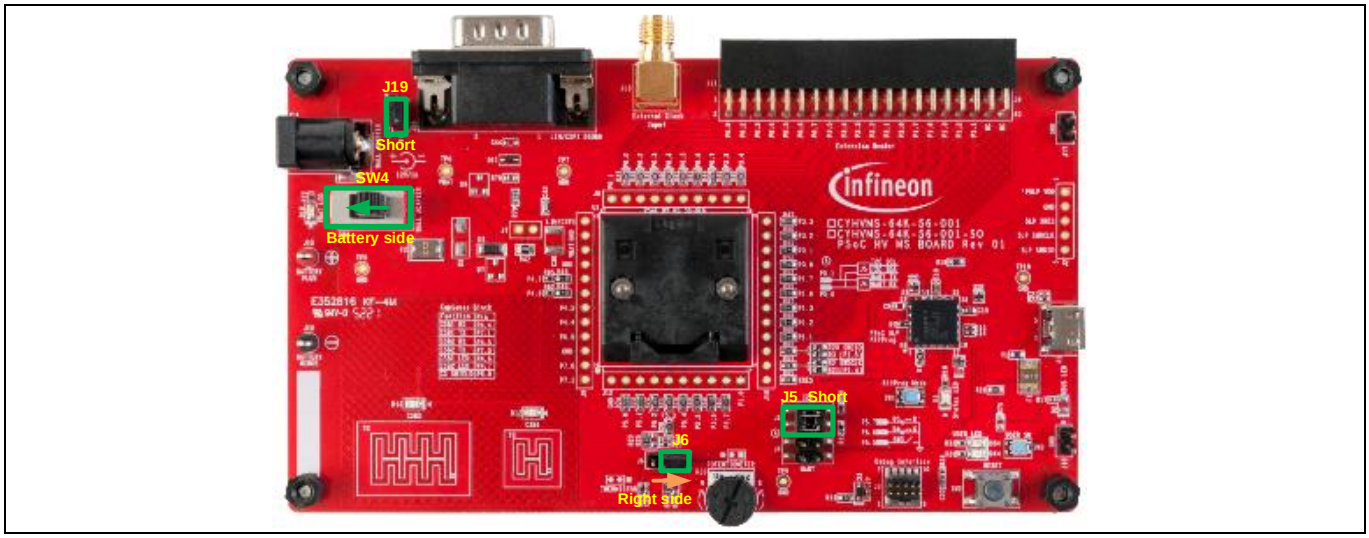

Figure 30 CYHVMS-64K-56-001 initial jumper and switch settings

Note: If you do not have the IAR I-jet Debugger, skip steps 9 to 12.

6. Connect the IAR I-jet Debugger to J3 (MIPI 10-pin) of the board with the 12 V power supply, and slide SW4 to Wall Adapter side (See Figure 31). See the PSOC™ HV MS Evaluation Board user guide [5] for more detailed information

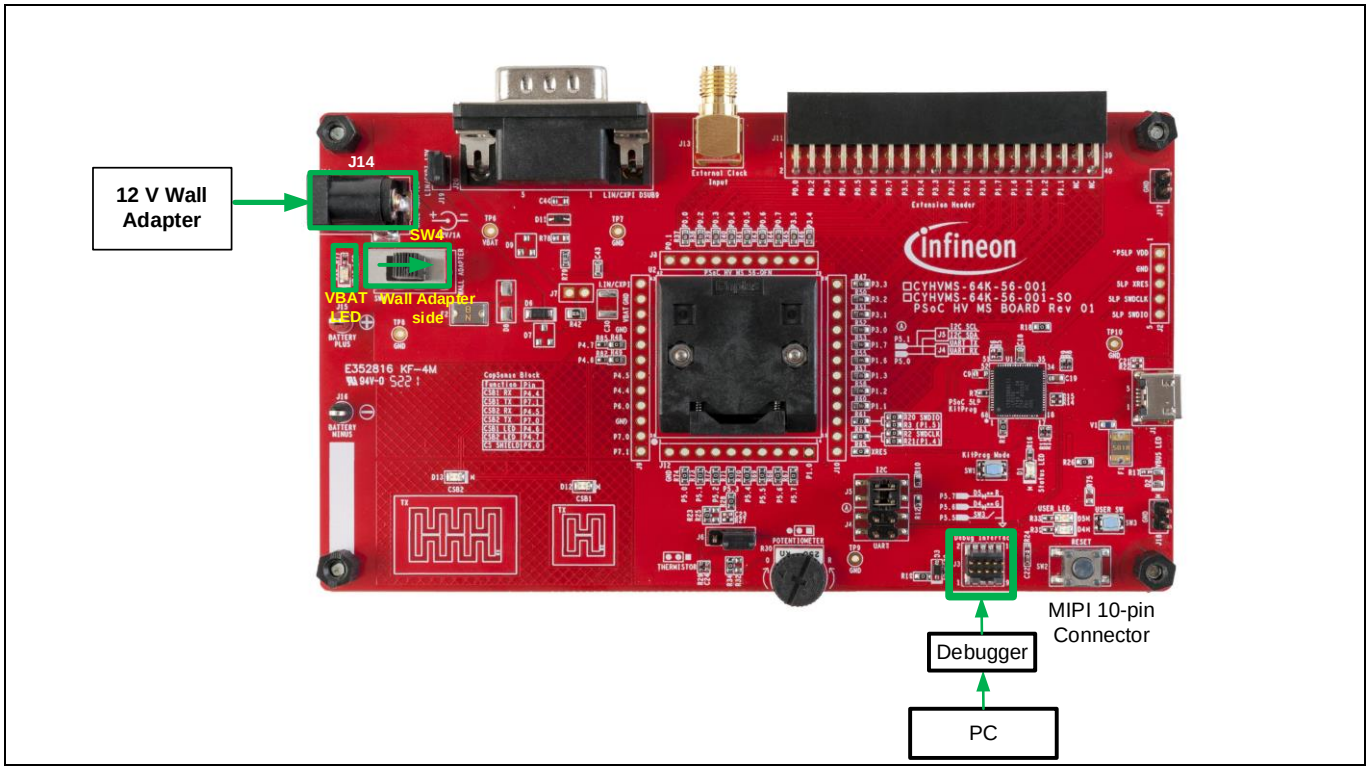

Figure 31 Connection of IAR I-jet Debugger

7. Check the debugger settings. You can see the configured debugger in the menu

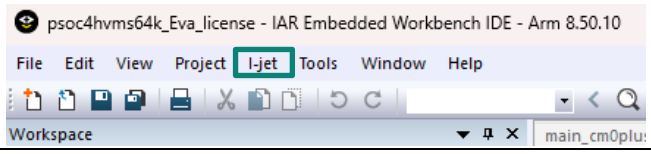

Figure 32 Configuring the debugger

8. To set correct debugger, open **Project** > **Options**

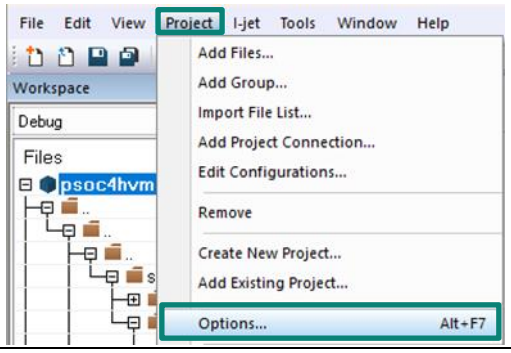

Figure 33 Options for debugger

9. Find **Debugger** in **Category**, and select **I-jet** in the **Driver** column

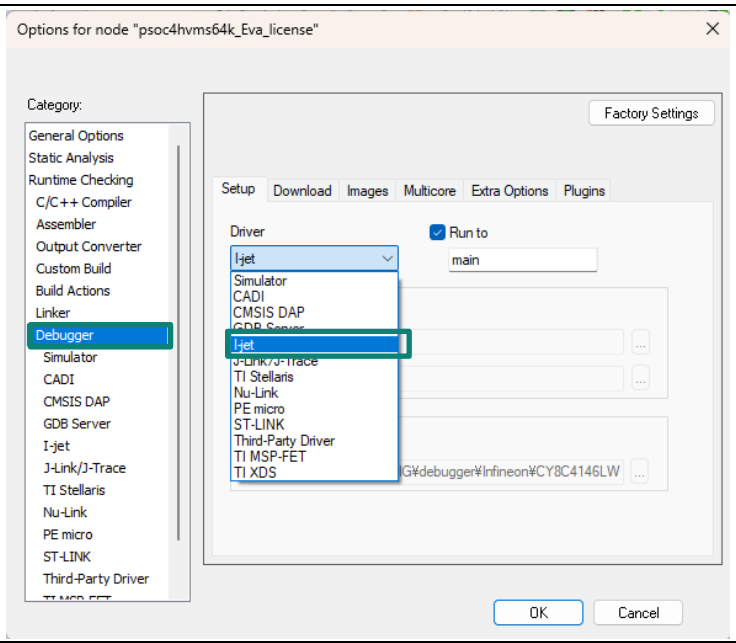

Figure 34 Finding debugger

10. Download the project to PSOC™ HV MS device by selecting **Project** > **Download** > **Download active application**

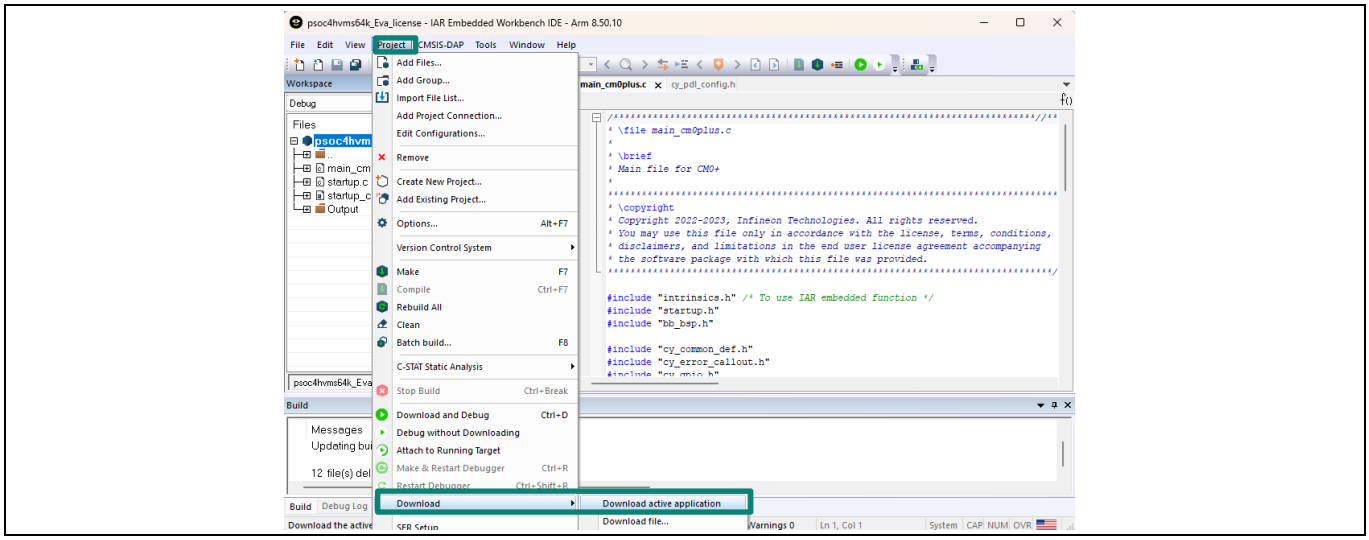

Figure 35 Downloading the application

11. Push the reset switch (SW2) on the board, and then confirm that the LEDs (D4, D5) are blinking
12. To download application using onboard KitProg3, do the following:
    a) Connect the USB cable to J1 of the CYHVMS-64K-56-001 Evaluation Board with the 12 V power supply from J14 (See Figure 36).
    b) Slide to the Wall Adapter side (SW4), and make sure the LEDs (D1, D10) are on. Refer to the PSOC™ HV MS Evaluation Board user guide [5] for more information.

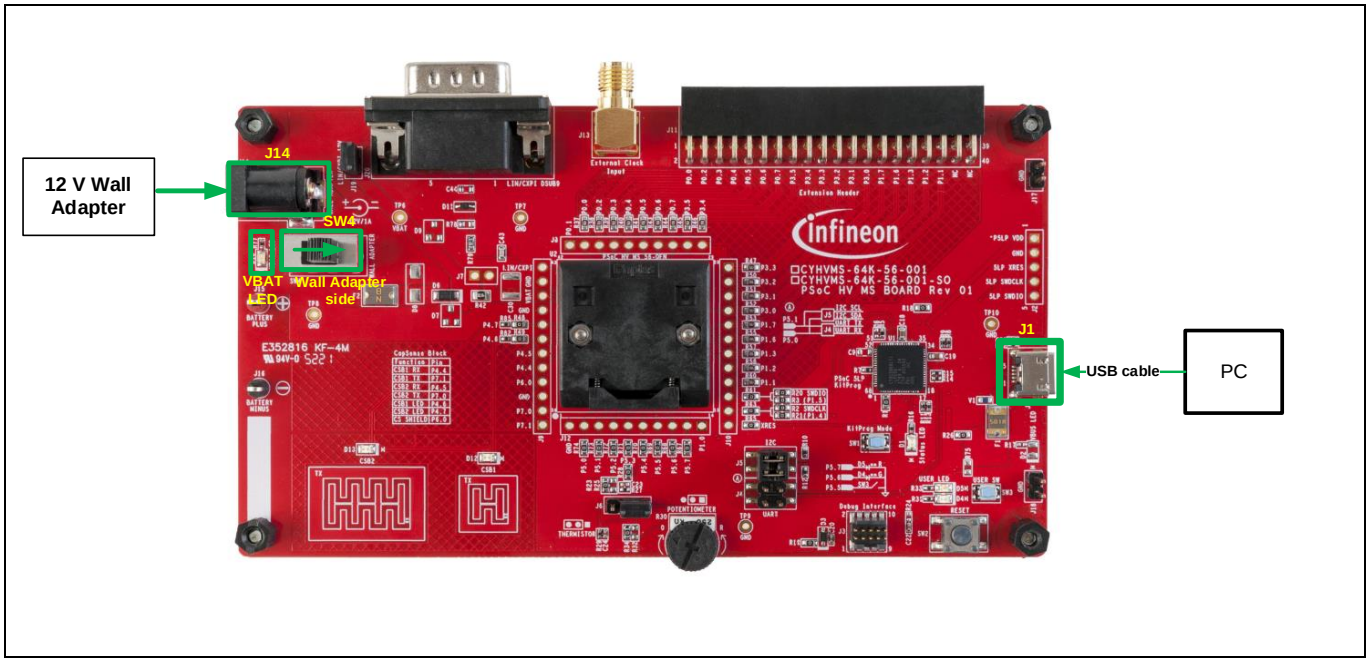

Figure 36 Connection of USB cable

13. Check the debugger settings. You can see the configured debugger in the tab. If using KitProg3, verify that **CMSIS-DAP** is set

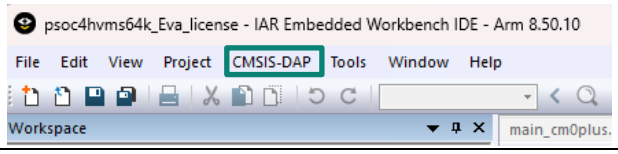

Figure 37 Checking the debugger settings

14. Select **Project** > **Options** to change the settings

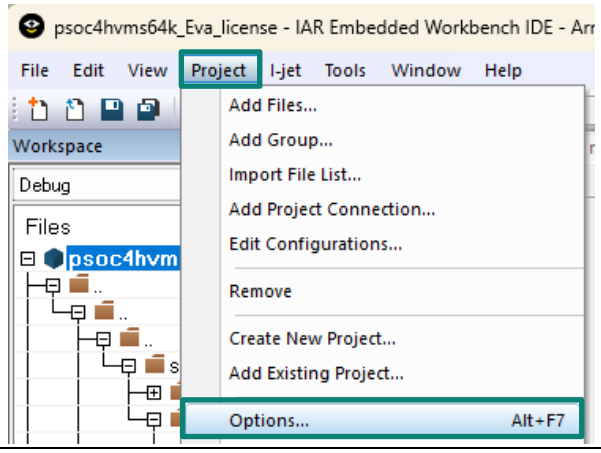

Figure 38 Options to change settings

15. Select **CMSIS DAP** from **Category**, and click the **Interface** tab. Check that the settings are as follows:
    - Probe config: Auto
    - Interface speed: 2 MHz
16. Click **OK** to reflect the change and leave the **Options** menu

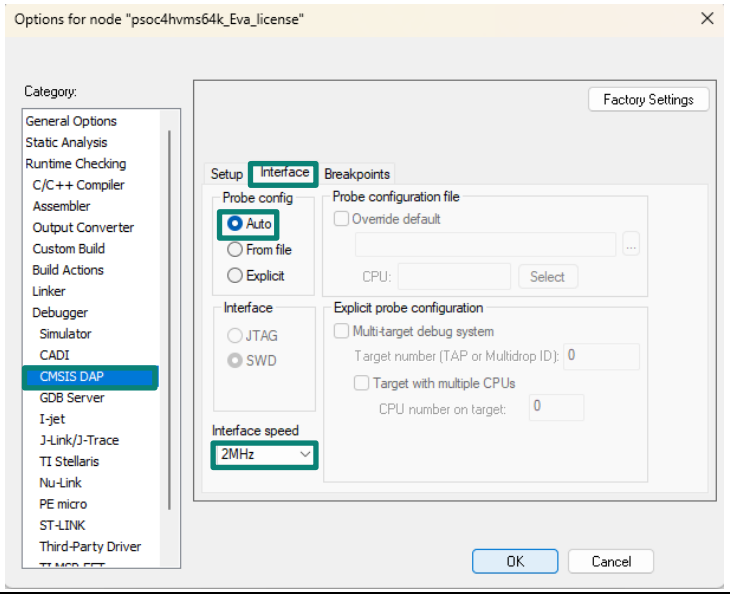

Figure 39 Applying the changes

17. Select **Project** > **Rebuild All** to build the project

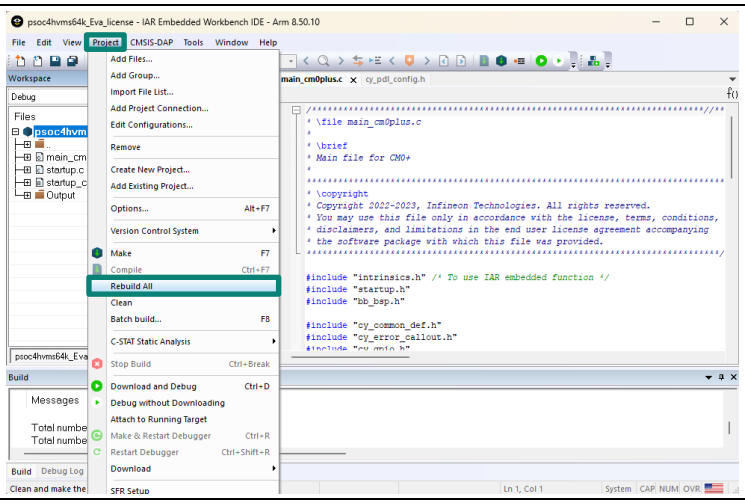

Figure 40 Rebuilding the project

18. Download the project to PSOC™ HV MS device by selecting **Project** > **Download** > **Download active application**

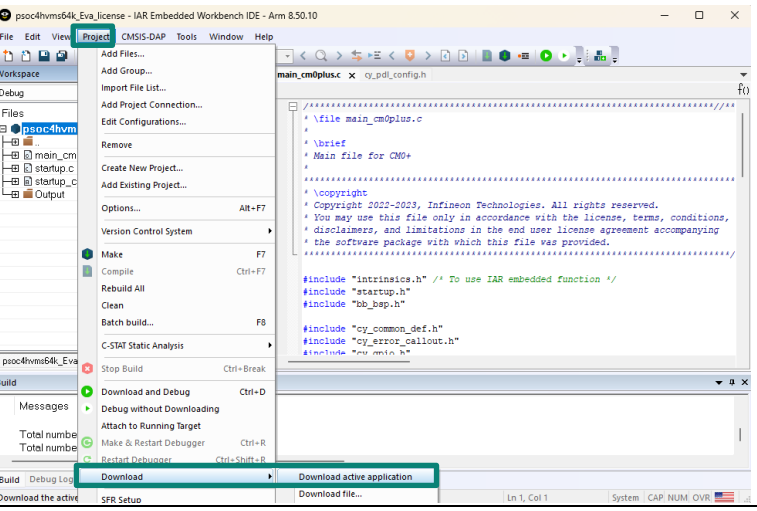

Figure 41 Downloading the application

19. Push the reset switch (SW2) on the board, and then confirm that the LED (D4, D5) is blinking
20. To debug the project using either IAR I-jet debugger or onboard KitProg3 (CMSIS DAP), select **Project** > **Download and Debug**. For more information on debugging, see the IAR EWARM user guide

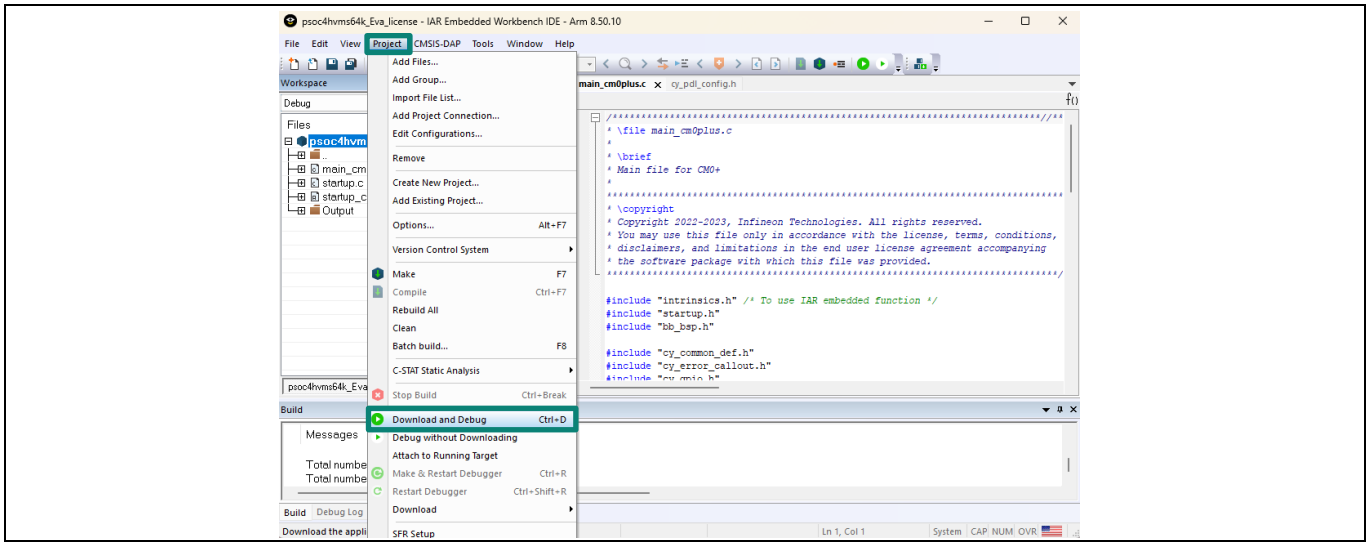

Figure 42 Debugging the project

3.4.4 Other sample codes

The other code examples are in this folder:
C:\INFINEON_AUTO_SW_\PSoC4HV_PDL_MW_Vxxxx\examples

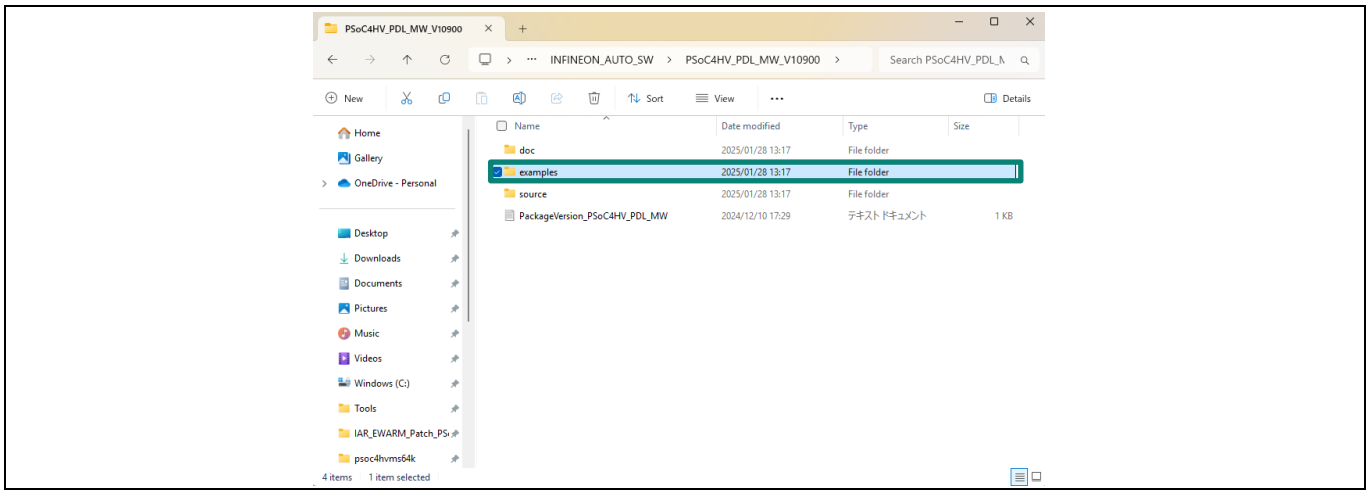

Figure 43 Example sample codes

1. Go to any code sample folder (e.g., *sar*) and copy the *main_cm0plus.c* file in the folder

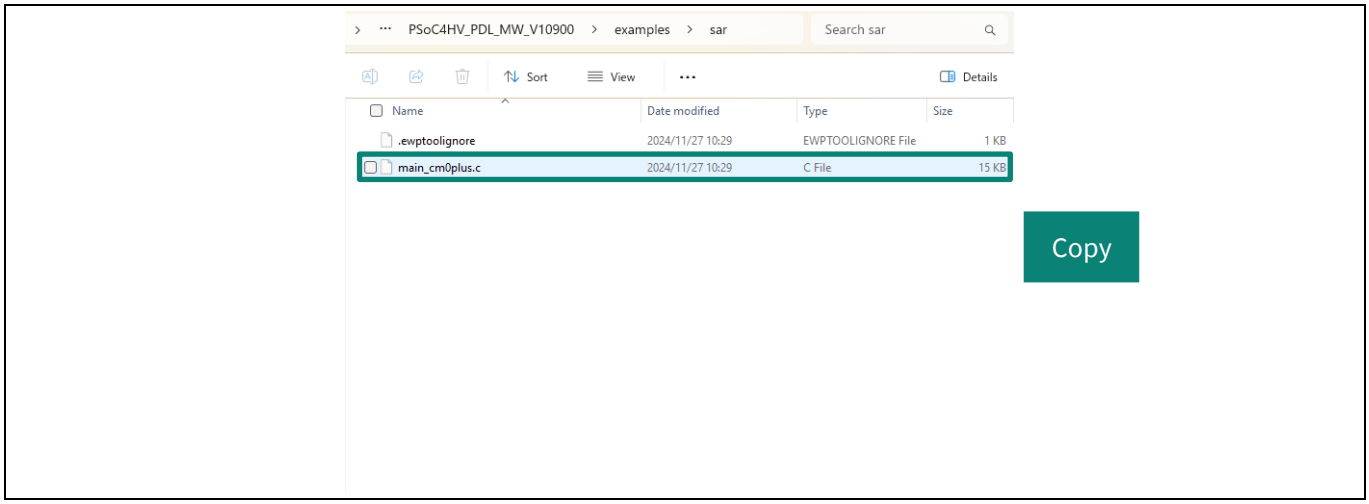

Figure 44 Copying main_cm0plus.c file

2. Navigate to *PSOC4HV_PDL_MW_Vxxxxx\examples\Tools\psoc4hvms64k* and replace the copied *main_cm0plus.c* file

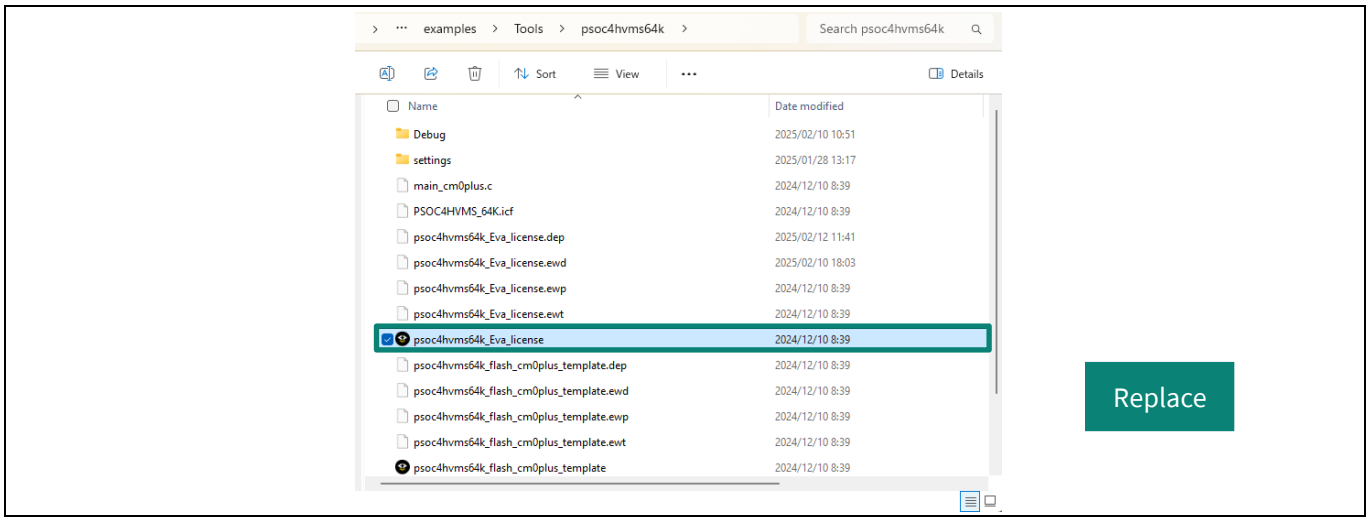

Figure 45 Copying the license file

3. From the same folder, open psoc*4hvms64k_Eva_license.eww* (or *qAW*psoc*4hvms*64*k_flash_cm0plus_template.eww* for source file license) to open the IAR EWARM sample project
4. Rebuild and download the project to the board.
   See the *C:\INFINEON_AUTO_SW\psoc4HV_PDL_MW_Vxxxxx\examples*readme.pdf file of the sample codes for a detailed explanation of the use case

3.5 Development tools

Table 5 Development tool

Vender	Emulators/probes	Compiler
IAR Systems	I-jet debugger Onboard KitProg3	EWARM (8.50.10)

3.6 Flash programming tools

Table 6 lists the supported flash programming tools. Infineon MiniProg4 programmer interface is for development purposes. For mass production purposes, JTAG programmer from a third-party is suitable. For Infineon Auto Flash Utility, contact Technical Support to obtain this software.

Table 6 Flash programming tools

Vender	Flash programmer	Software
Infineon	Miniprog4	Infineon Auto Flash Utility
	KitProg3	

Note: CYHVMS-64K-56-001 has the Cortex® SWD debug connector. SWD is compatible with MiniProg4.

Table 7 lists the pin assignment for the Cortex® debug connector.

Table 7 Cortex® debug connector

Pin	Signal
1	VDDD
2	SWDIO
3	GND
4	SWDCLK
5	GND
6	SWO (option)
7	KEY
8	NC
9	GND detect
10	nRESET

4 Summary

This application note guided you to set up PSOC™ HV MS development environment. Infineon provides an evaluation board and a wealth of sample software to help you get started with PSOC™ HV MS family. To evaluate the CYHVMS-64K-56-001 Evaluation Board, contact your local sales representative or Technical Support.

5 Glossary

Table 8 Glossary

Terms	Description
ADC	analog-to digital converter
AEC	Automotive Electronics Council
AHB	Advanced High-Performance Bus
API	application programming interface
AutoPDL	Automotive Peripheral Driver Library
BOD	brownout detector
CAPSENSE™	capacitive sensing
CM0+	Arm® Cortex®-M0+ (processor core name)
CPU	central processing unit
CPUSS	CPU subsystem
CXPI	Clock Extension Peripheral Interface
DAP	debug access port
DMA	direct memory access
DMAC	DMA controller
DUT	device under test
DW	data wire
ECC	error correction code (safety)
GPIO	general-purpose I/O
HPOSC	high-precision oscillator
HV	high-voltage
HVSS	high-voltage subsystem
HMI	human machine interface
I/O	input or output
IAR	IAR Systems (Company name)
ILO	internal low-speed oscillator
IMO	internal main oscillator
IOSS	I/O subsystem
IRC	interrupt controller
IRQ	interrupt request
ISR	interrupt service routine
LIN	Local Interconnect Network
MCU	microcontroller unit
MIPI	Mobile Industry Processor Interface
MPU	memory protection unit
MS	mixed signal
MSC	multi-sense converter

Glossary

Terms	Description
NMI	non-maskable interrupt
PILO	precision low-power oscillator
POR	power-on reset
PWM	pulse width modulation
QFN	quad flat no-lead
RAM	random access memory
ROM	read-only memory
SAR	successive approximation register
SCB	serial communication block
SNR	signal-to-noise ratio
SPI	serial peripheral interface
SRAM	static RAM
SRSS	system resource subsystem
SWD	serial wire debugging
TCPWM	timer, counter, and PWM
WDT	watchdog timer

References

The following are the PSOC™ HV MS family series application notes, datasheets, and reference manuals. Contact Technical Support to obtain these documents or get support.

[1] AN234116 – Hardware design guide for PSOC™ HV MS family

[2] PSOC™ HV mixed signal datasheet

[3] PSOC™ HV MS architecture reference manual

[4] PSOC™ HV MS registers reference manual

[5] PSOC™ HV MS Evaluation Board user guide

Revision history

Document revision	Date	Description of changes
**	2022-03-24	Initial release
*A	2023-04-11	Added PSOC™ HV MS-128K information in all chapters
*B	2023-09-18	Updated number of low-power comparator Corrected typos
*C	2025-02-20	Updated to use AutoPDL

Trademarks

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

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™.

Edition 2025-02-20

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2025 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

002-34115 Rev. *C

Important notice

The information contained in this application note 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 application note 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 application note.

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.