XMC1000, XMC4000

32-bit Microcontroller Series for Industrial Applications

Working with DAVE™ APPs and moving from DAVE™ v3 to v4

AP32295

Application Note

About this document

Scope and purpose

This document introduces DAVE™ Version 4.0 and provides a step by step guide on how to move existing DAVE™ v3 projects to the new DAVE™ setup.

When moving a project from DAVE™ v3 to v4, you will create a new v4 project and, with the instructions in this guide, derive the configuration settings to use from the existing v3 project, and manually select which DAVE™ v4 APPs to use to replace the ones from DAVE™ v3.

Note: It is not compulsory to move your projects between the DAVE™ versions. DAVE™ v3 is, at the time of writing, still intended to be supported for a significant period of time.

Applicable Products

- XMC1000 and/or XMC4000 Microcontrollers Family
- DAVE ™ Version 4.0

References

Infineon: DAVE™ **http://www.infineon.com/DAVE**

Infineon: XMC Family **http://www.infineon.com/XMC**

Table of Contents

Table of Contents

1 DAVE™ v4 IDE key features

DAVE™ v4 is a high-productivity development platform for the XMC microcontroller families to simplify and shorten software (SW) development. With DAVE™, developers can generate the SW library to efficiently use the innovative application-optimized peripherals. Code generation is based on pre-defined and tested application-oriented SW components, called DAVE™ APPs.

DAVE™ v4 provides the following enhancements:

- Faster and more robust design.
  - The time for code generation, responses to user interactions in the graphical user interfaces, and the build time, are all improved significantly.
- DAVE APPs are based on low level peripheral drivers (XMC Lib), which makes generated code more efficient and more readable.
- A set of low level peripheral drivers (XMC Lib) is available and is used by DAVE APPs. The XMC Lib can also be used independent of DAVE or DAVE APPs with any other tool chain that supports XMC (Atollic, Altium, Keil, IAR, and Rowley).

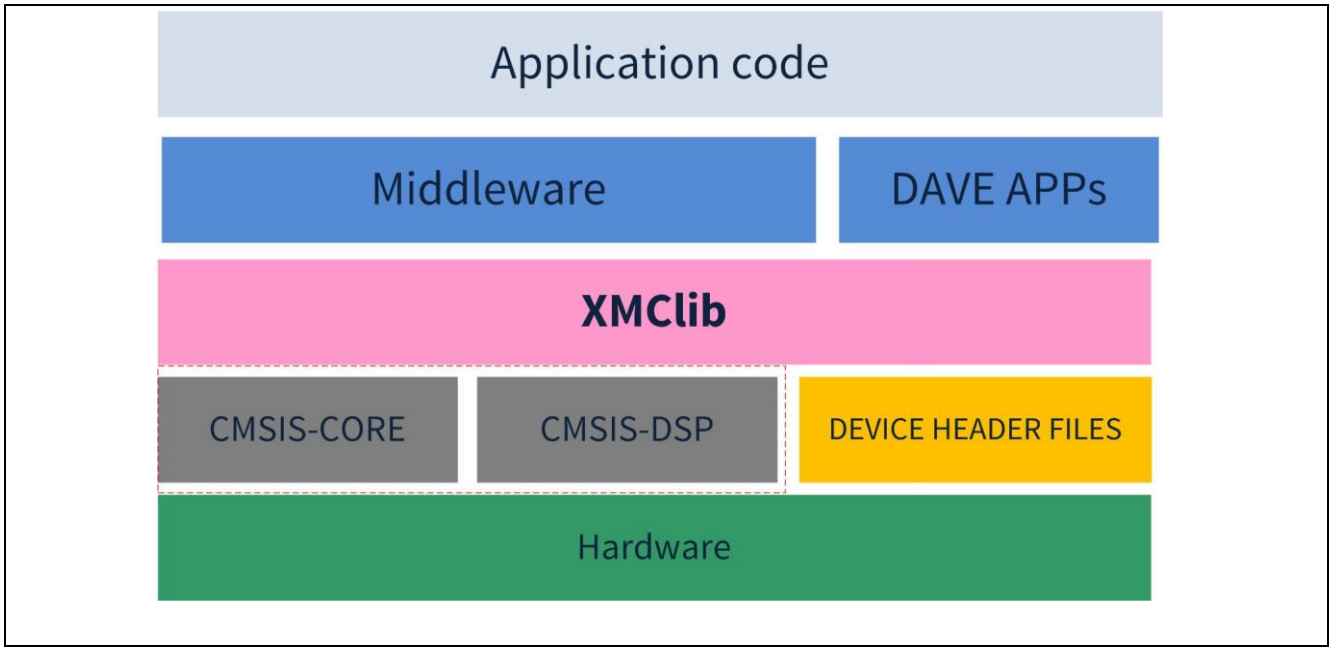


Figure 1 DAVE™ Architecture

This section is divided into Updated features and New features.

1.1 Updated features

- Project outline
- DAVE APP Selection View
- DAVE APP Dependency View
- DAVE APP HW Signal Connectivity View
- Instance Label of DAVE APPs

1.1.1 Project outline

Projects in DAVE v4 are made as self-contained as possible. This means all CMSIS header files, all device specific low level drivers, start-up files, and linker scripts are copied into the project. Also, all DAVE APPs will be fully included. The only remaining references are the standard header files of the compiler tool chain and the device model (Device Descriptions) that is used by the resource solver. With this Approach, a qualified project can be more easily frozen and archived.

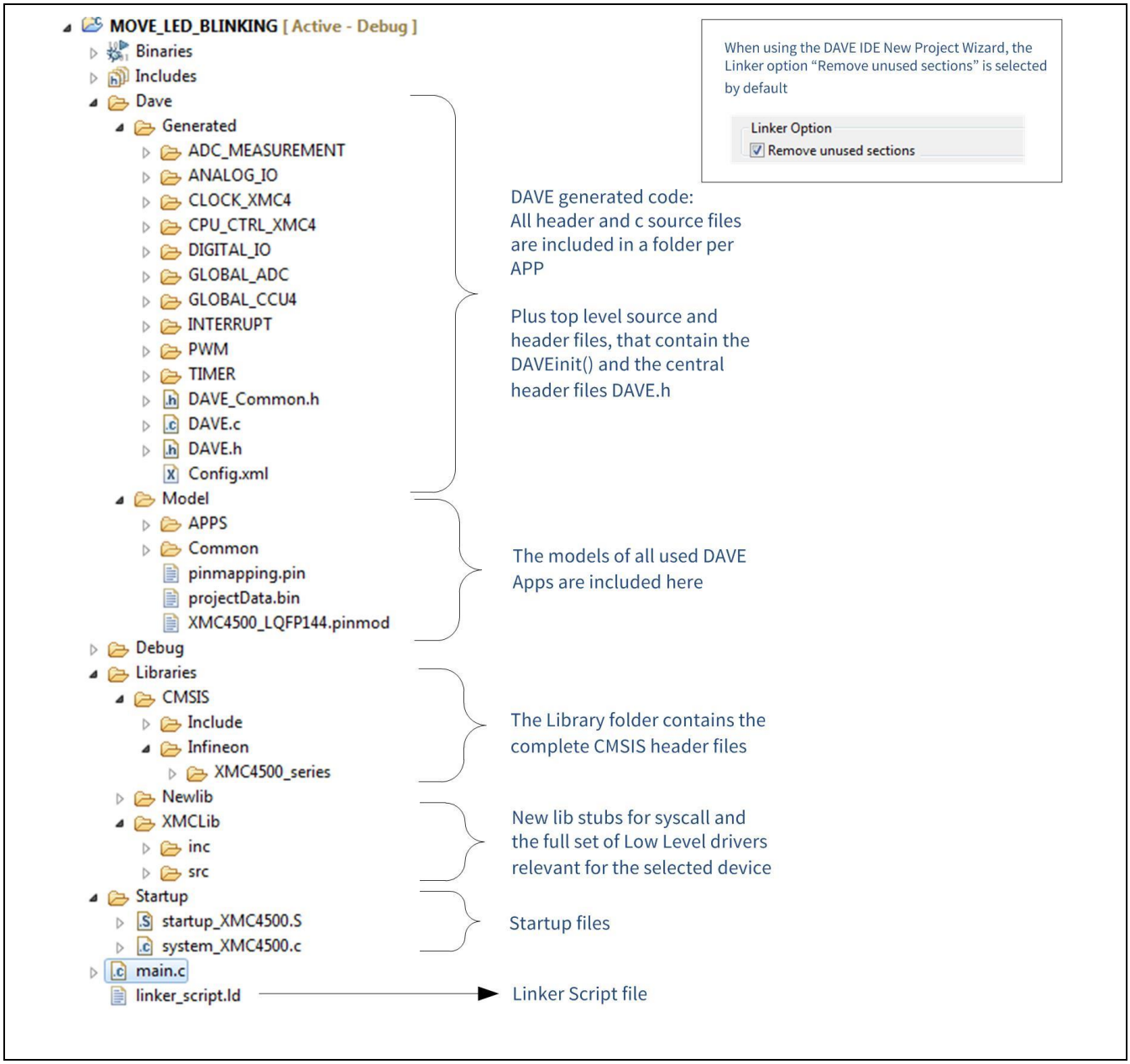


Figure 2 DAVE v4 Project outline

As the project may contain more files and functions than required by the application, the project option "remove unused section" is selected by default, to prevent dead code from using up memory space.

1.1.2 DAVE APP Selection View

The APP selection view with improved filter and categorization is now implemented as a modal view. This makes it easier to find the required DAVE APPs and save screen space when APP selection is not required.

The New APP selection view can be activated from any APP Dependency /Connectivity Views or the tool box bar. It should be closed after the required APPs are selected. It includes hidden categories for APPs that are normally not explicitly selected; e.g. APPs to configure Global registers.

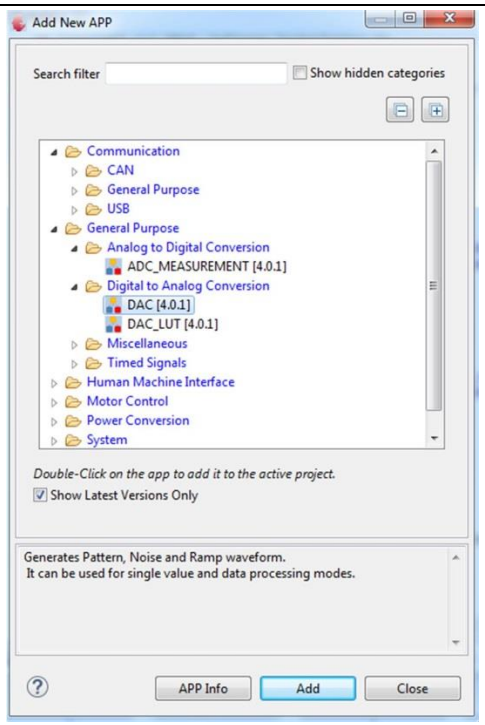


Figure 3 DAVE v4 APP selection view

1.1.3 DAVE APP Dependency View

To improve the visualization of dependencies between DAVE APPs, the APP Dependency view has been reworked in the following manner:

- Layered positioning of DAVE APPs with as many layers as required
- The top level APP of a dependency tree is positioned at the top, the next dependent DAVE APP on the next lower level and so on. The lowest level APP of a dependency tree is positioned on the lowest level of the view.
- The APPs cannot be moved outside of their assigned layer.
- The APPs can be ordered within the layer by the user, this order will be kept, also after starting the auto layout functionality.
- A new APP is always added from the left.

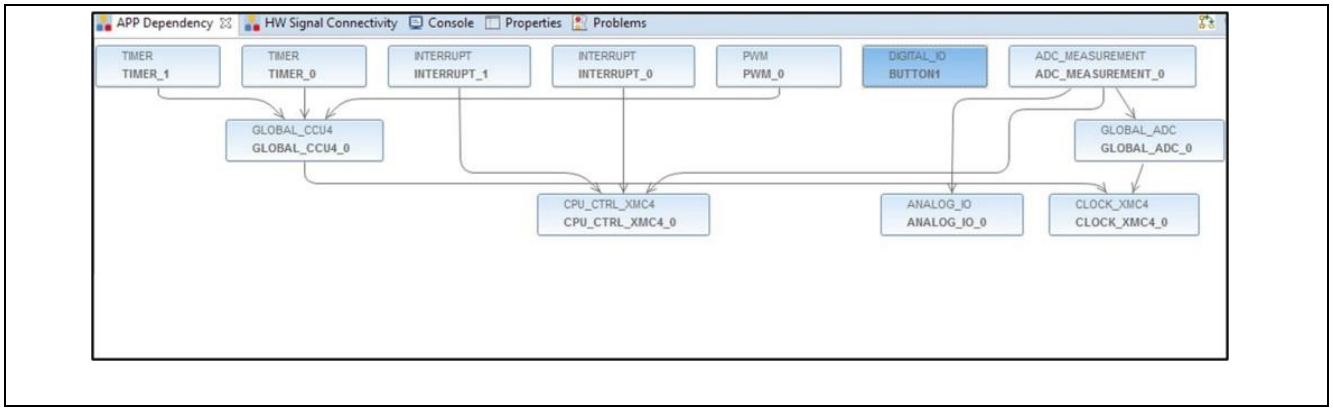
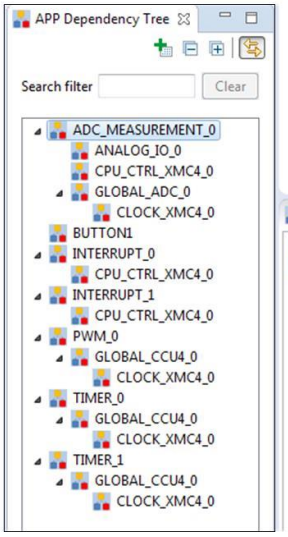


Figure 4 DAVE v4 APP dependency view

When a project has a large number of DAVE APPs are instantiated, the “APP Dependency Tree” view can be used to select an APP. When selected, the Dependency will focus on the selected APP (shaded blue) plus its dependency connections

“APP Dependency Tree” view:
ADC_MEASUREMENT_0 APP is selected.



In the APP Dependency” view:
The dependency will focus on the selected APP plus its dependency connections

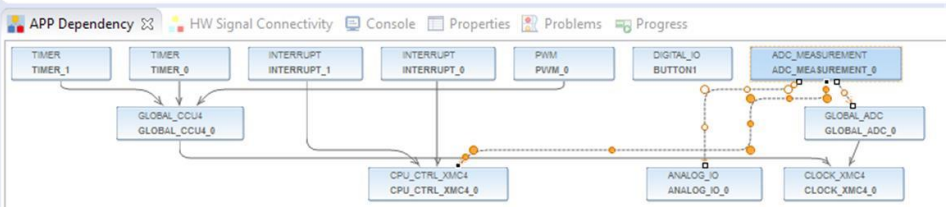


Figure 5 DAVE v4 APP Dependency Tree view

1.1.4 DAVE APP HW Signal Connectivity View

The HW Signal Connectivity view has been changed and represented by a schematic circuit style. Each APP is now represented as an object with all input and output signals displayed. Inputs are on the left side, outputs are on the right side. The green lines represent connections made by the manifest and cannot be modified by the user; the blue lines are user defined lines. The purposes of this view is to visualize signal and event connections graphically, the user cannot do any changes in this view.

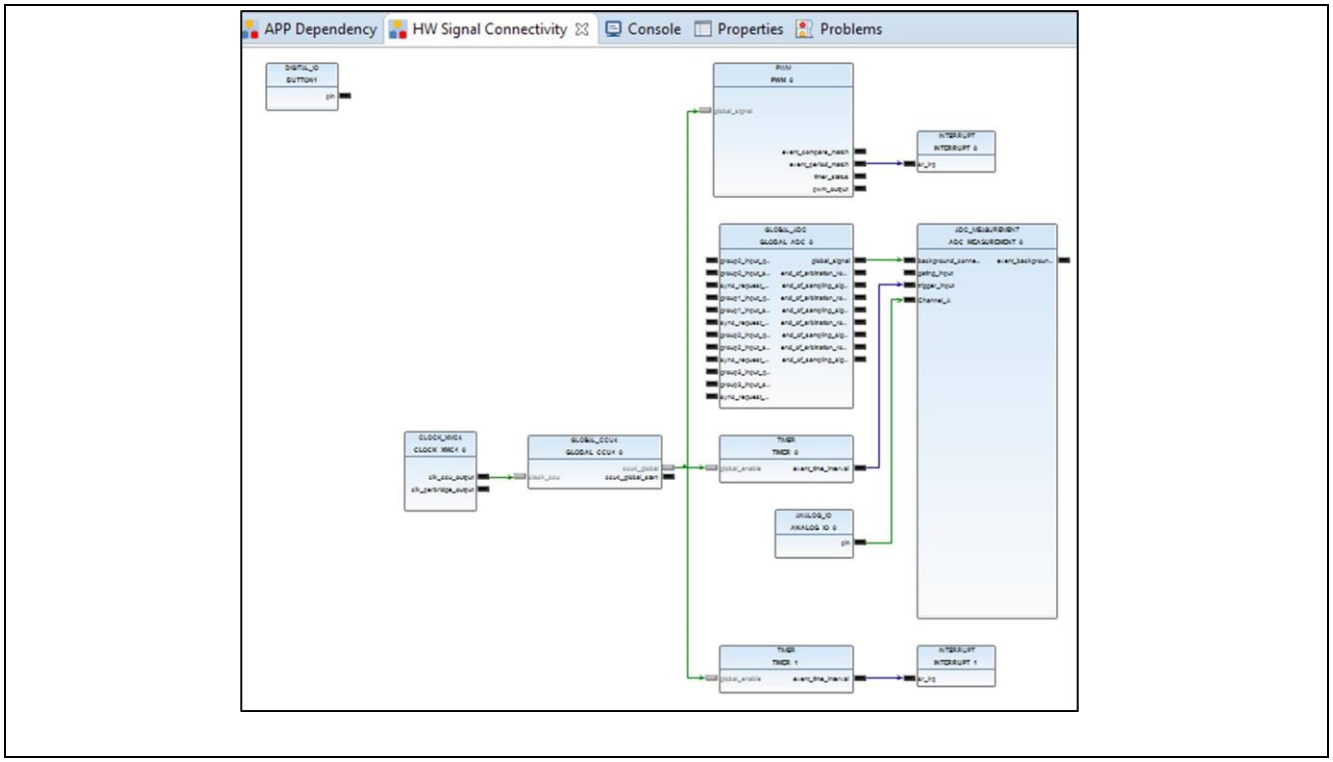


Figure 6 DAVE v4 APP HW Connectivity view

1.1.5 Instance Label of DAVE APPs

DAVE™ v4 provides the option to change the default instance label to a user defined symbol. This symbol can be used in the APIs of the APP to refer to the respective APP instance. With this, user code can easily be ported to other projects that use a certain set of APPs. It is just required that both projects use the same user defined instance label.

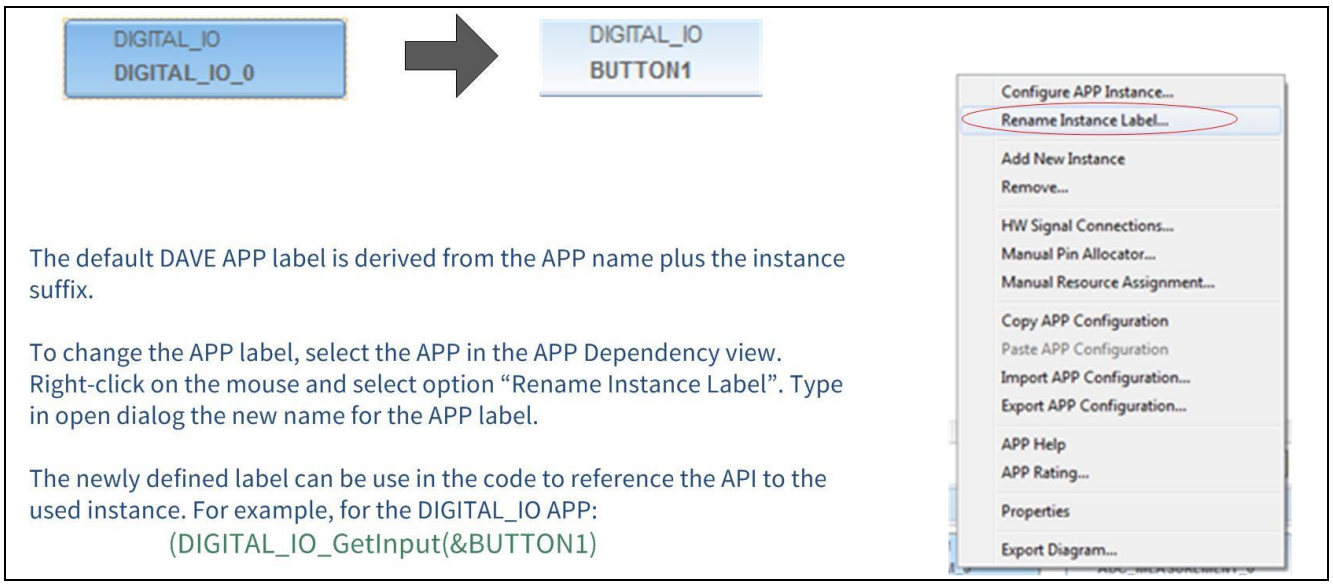


Figure 7 DAVE v4 Rename Instance Label

1.2 New features

- Graphical Pin Mapping Functionality
- Global Interrupt Editor
- Copy and Paste for DAVE APP Configurations

1.2.1 Graphical Pin Mapping Functionality

Prior to DAVE v4, manual pin assignments were presented in table view format. A graphical pin mapping functionality for manual pin assignments has been implemented within its own dedicated perspective window called Pin Mapping.

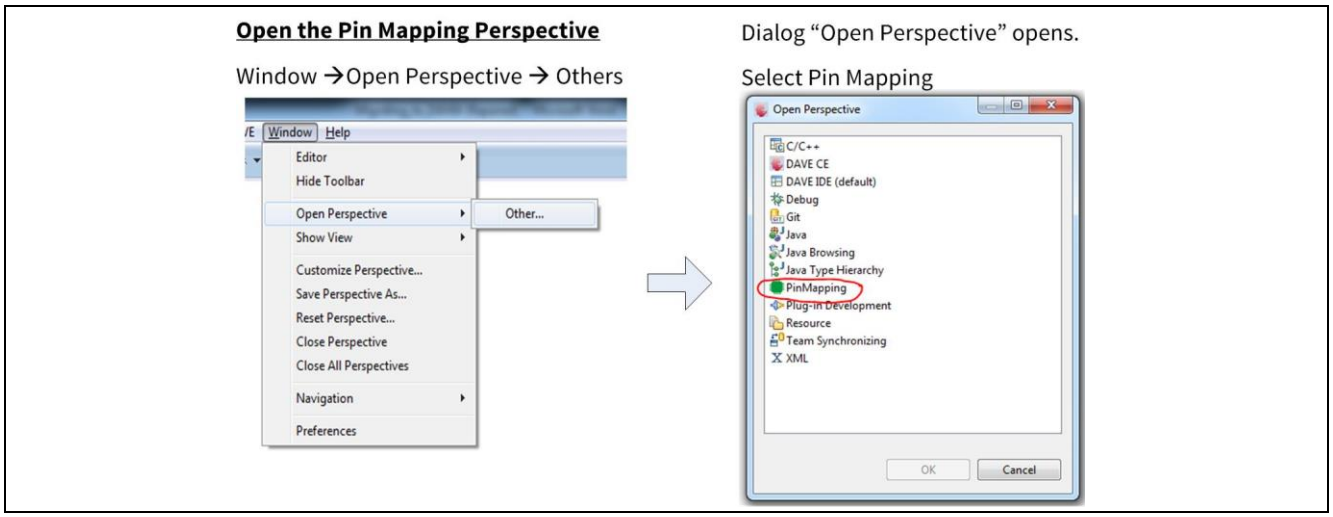


Figure 8 Enabling the Pin Mapping Perspective

DAVE™ v4 IDE key features

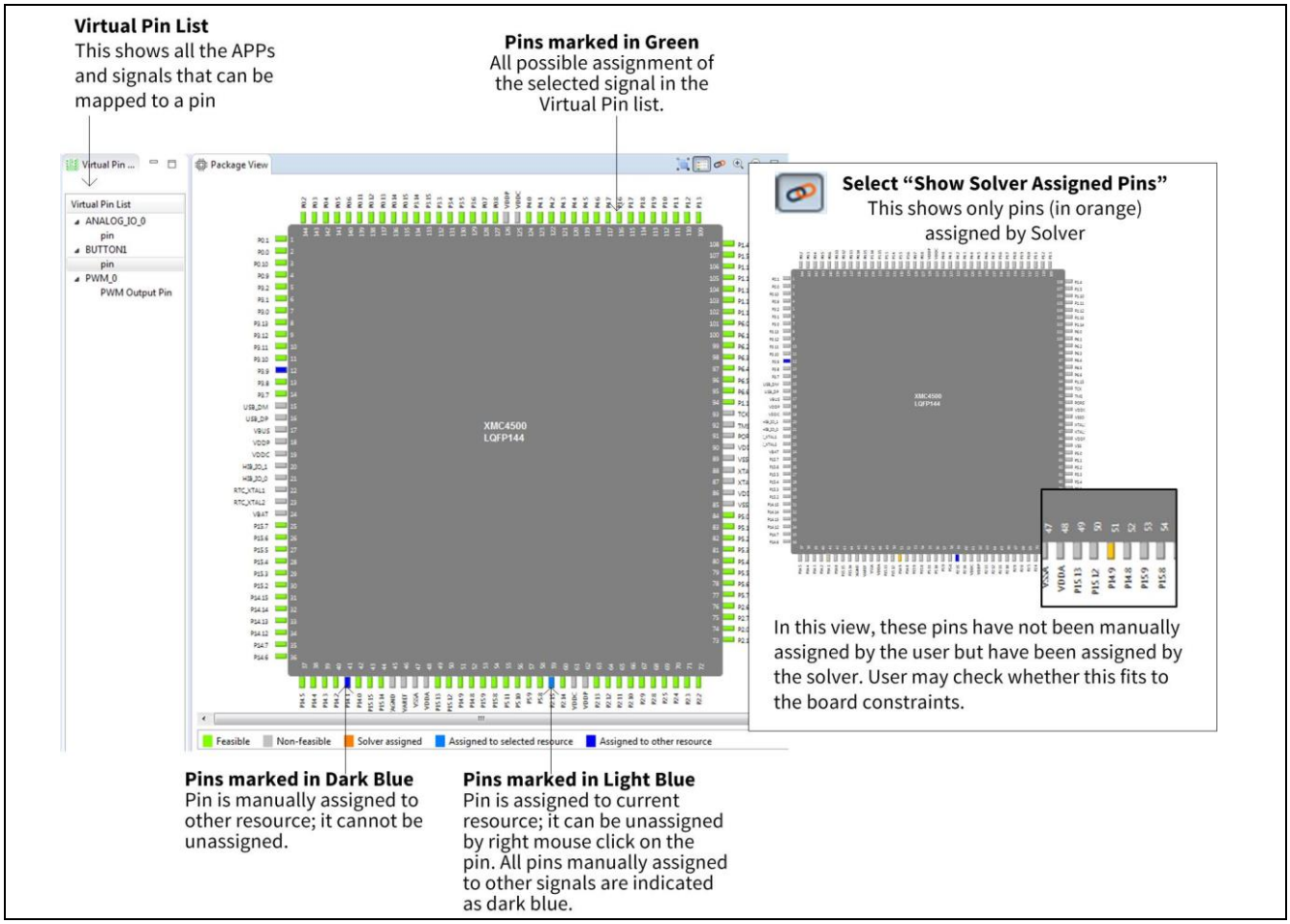


Figure 9 Graphical Pin Mapping perspective

1.2.2 Global Interrupt Editor

The Global Interrupt Editor shows all defined interrupts in a global User Interface (UI) and allows you to change the priority of all defined interrupts.

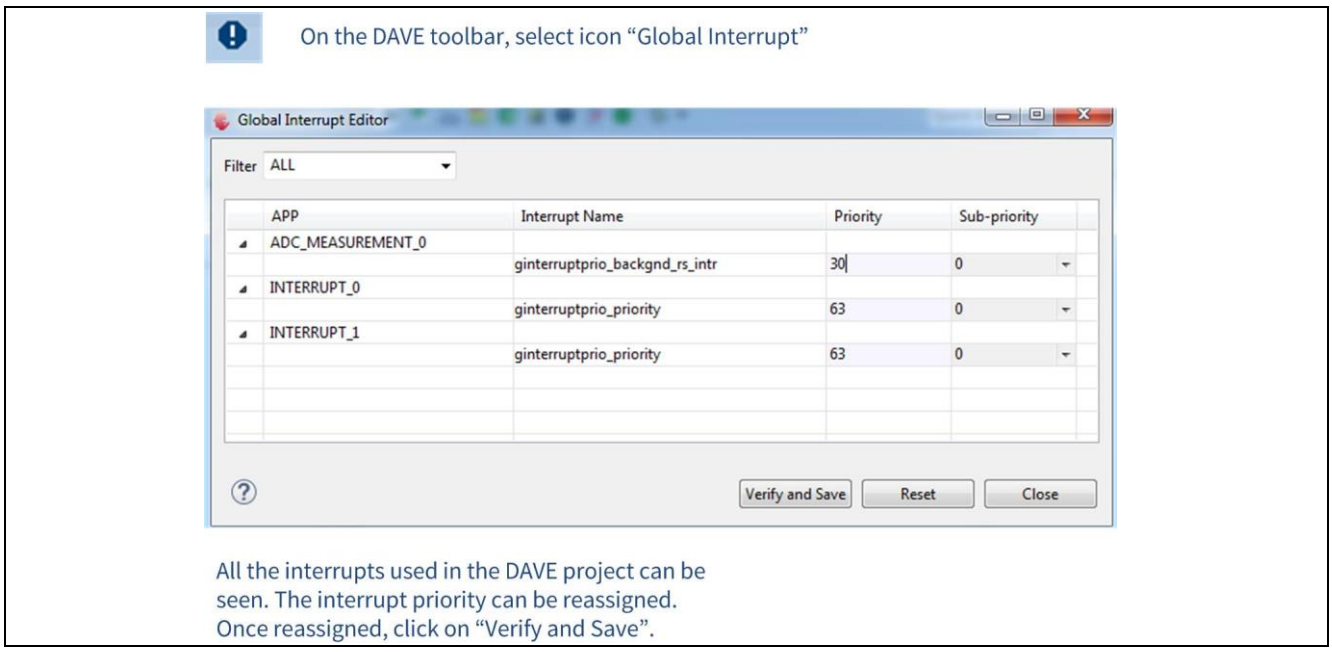


Figure 10 Global Interrupt Editor

1.2.3 Copy and Paste for DAVE APP Configurations

The existing configuration of any APP can be copied to another instance of this APP in the same project by using the Copy and paste functionality, accessible from a right-mouse click on an APP. The configuration of an APP can also be exported to a file (XML) and imported from there to another instance of this APP in any project.

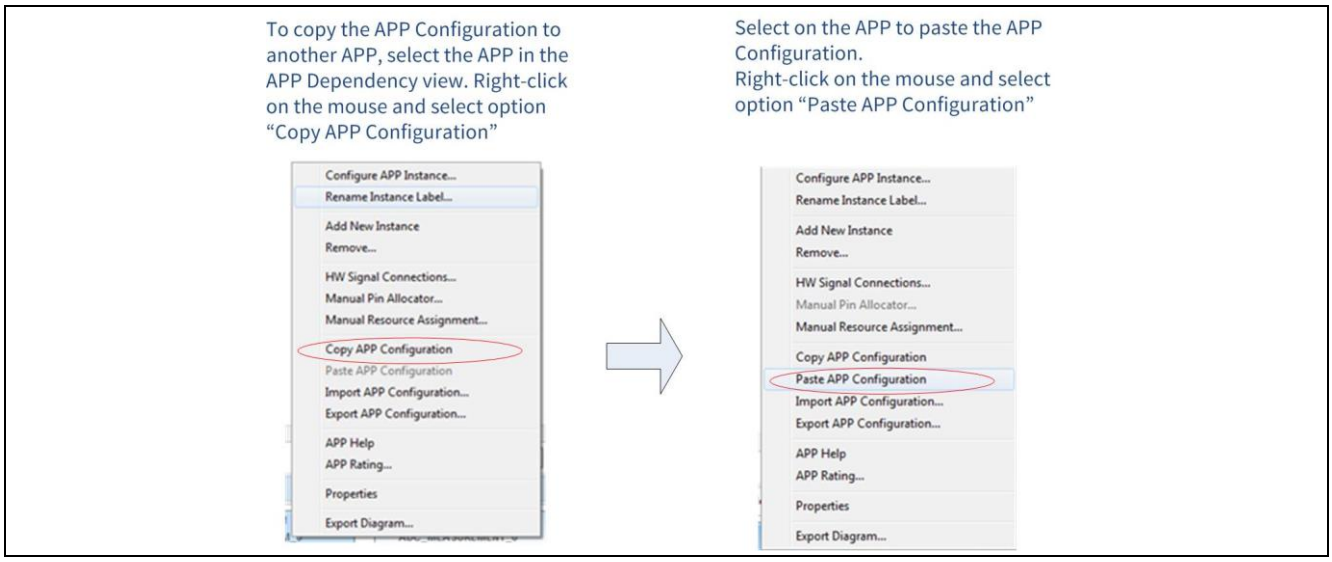


Figure 11 Copy and Paste for DAVE APP Configurations

2 APP changes from DAVE v3 to v4

DAVE APPs are application components that abstract a certain use case. There is no change in the general philosophy of DAVE v4 APPs in comparison to DAVE v3. However, there are two major changes in the implementation concept of DAVE v4 APPs:

- DAVE v4 APPs are built on top of the peripheral specific low level drivers (XMC Lib) introduced in this release.
  - With this change, DAVE v4 APPs do not generally, directly access registers of the MCU, but use the API of the XMC Lib to initialize Hardware (HW). DAVE v4 APPs essentially generate the configuration data structure based on the selected configuration options and call the appropriate 'Init' functions.
  - Depending on the use case of the DAVE v4 APPs, higher level API functions are created to implement the defined use cases, by using the APIs provided by the LLDs. With this, code generation becomes more transparent and robust and generated code can easily be qualified to be used in production.
- New Domain Specific Language (DSL) to define the GUI, GUI actions, instantiation actions, and code generation actions.
  - Groovy has been chosen as the DSL to define DAVE v4 APPs, related control functions and code generation templates. With this, the GUI design to configure DAVE APPs is more flexible and the APP design is more robust.

These changes mean that most of the new DAVE APPs are conceptually re-designed to improve ease of use, including improving their usability in productive code.

2.1 APP Mapping between DAVE™ v3 and DAVE™ v4

Note: There is no compulsory implementation need to migrate old projects between the two versions, because DAVE v3 and DAVE APPs for DAVE v3 will still be available and maintained for a significant period of time. However, if you do wish to, you can map your App from v3 to v4.

Because of the changes in concept for DAVE APPs and the data model of DAVE APPs to improve the speed and robustness of projects, DAVE APPs are not compatible between DAVE v3 and DAVE v4. However, it is possible to manually move projects from DAVE v3 to DAVE v4.

In DAVE v4, APPs have been re-defined to provide more ease of use and in some cases, consolidated into a single APP. In the following sections, the changes in the major groups of APPs are discussed, with a mapping of the APPs from DAVE v3 to DAVE v4.

2.2 General Purpose APPs

2.2.1 Generic APPs

Table 1 DAVE v4 Generic APPs

S/N	APP Name	Description	Available (2015-02-23)	New
1	CRC_SW	To calculate and validate CRC (Cyclic Redundancy Check) on a block of data using a software CRC implementation.	✓	-
2	CRYPTO_SW	To encrypt/decrypt data based on a user-defined key.	✓	-
3	DMA_CH	To perform single and multi-block data transfer using the GPDMA module on XMC4000.	✓	-
4	E_EEPROM_XMC1	Emulates a portion of flash as an EEPROM for data storage.	✓	-
5	GLOBAL_DMA	This is a DMA global APP. Aggregated by *DMA_CH.*	✓	✓
6	GUI_LCD	To provide an OLED display driver interface for Segger GUI library. OLED Display driver is interfaced with Segger GUI library using SPI_MASTER.	✓	-
7	GUI_SEGGERLIBRARY	This is a Segger Graphic library with graphical user interface in RTOS/Non-RTOS environment. It is designed to provide an efficient processor and display controller independent graphical user interface, for any application that operates with a graphical display.	✓	-

Table 2 Generic APPs: Main changes in DAVE v4 from DAVE v3

S/N	APP Name	DAVE v3 equivalent	Comments
1	CRC_SW	CRC001	-
2	CRYPTO_SW	AES001	-
3	DMA_CH	DMA002 DMA003 DMA004	-
4	E_EEPROM_XMC1	FEE001	-
5	GUI_LCD	GUILC001	-
6	GUI_SEGGERLIBRARY	GUISL002	• Added support for other LCD display types.

Table 3 Generic APPs: Mapping DAVE v3 to DAVE v4

S/N	DAVE v3	Equivalent APP for DAVE v4	Comments
1	AES001	CRYPTO_AES	-
2	CRC001	CRC_SW	-
3	DMA002	DMA_CH	-
4	DMA003	DMA_CH	-
5	DMA004	DMA_CH	-
6	FEE001	E_EEPROM_XMC1	-
7	GUILC001	GUI_LCD	-
8	GUISL002	GUI_SEGGERLIBRARY	-

2.2.2 Timer/PWM related APPs

Code changes for PWM related APPs.

Table 4 DAVE v4: PWM-related APPs

S/N	APP Name	Description	Available (2015-02-23)	New
1	COUNTER	To count the occurrence of external events using one timer slice of CCU4 or CCU4.	✓	-
2	GLOBAL_CCU4	To initialize CCU4 global register set and to support synchronous start of CCU4 slices.	✓	-
3	GLOBAL_CCU8	To initialize CCU8 global register set and to support synchronous start of CCU4 slices.	✓	-
4	PWM	To generate a PWM using one timer slice of CCU4 or CCU8.	✓	-
5	TIMER	To provide an accurate timer by using CCU timer. This can be used as trigger input to other peripherals or create an event.	✓	✓

Table 5 Main changes in DAVE v4 from DAVE v3: PWM-related APPs

S/N	APP Name	DAVE v3 equivalent	Comments
1	COUNTER	CNT001	• Added support for external gating event
2	GLOBAL_CCU4	GLOBAL_CCU4 CCU4ST01 CCUST01	• Added signal and API for global start control signal control. This allows a signal to be connected to CCU module external events.
3	GLOBAL_CCU8	GLOBAL_CCU8 CCU8ST01 CCUST01	• Added signal and API for global start control signal. This allows a signal to be connected to CCU module external events.
4	PWM	PWMSP001 PWMSP002	• Added CCU4 or CCU8 module selection for PWM generation. • Removed timer concatenation support with use of single CCU slice of CCU for PWM generation. • Fixed point implementation supporting a duty cycle range from 0.01 to 100.00. Duty cycle value is scaled by 100. • PWM output enabled by default. • External events not supported.

Table 6 Mapping DAVE v3 to DAVE v4: PWM-related APPs

S/N	DAVE v3	Equivalent APP for DAVE v4	Comments
1	CCU4ST01	GLOBAL_CCU4	-
2	CCU8ST01	GLOBAL_CCU8	-
3	CCUST01	GLOBAL_CCU4 GLOBAL_CCU8	-
4	CCU4GLOBAL	GLOBAL_CCU4	-
5	CCU8GLOBAL	GLOBAL_CCU8	-
6	CNT001	COUNTER	-
7	PWMSP001	PWM	-
8	PWMSP002	PWM	

2.2.3 ADC APPs

Table 7 DAVE v4: ADC APPs

S/N	APP Name	Description	Available (2015-02-23)	New
1	ADC_MEASUREMENT	To incorporates analog to digital conversion for the required measurements.	✓	-
2	ADC_QUEUE	To provide configurations for queue request source of VADC.	✓	-
3	ADC_SCAN	To provide configurations for the scan request source of VADC.	✓	-
4	GLOBAL_ADC	To configure the VADC global registers. It consumes CLOCK_XMC4 for XMC4x devices and CLOCK_XMC1 for XMC1x devices.	✓	-

Table 8 Main changes in DAVE v4 from DAVE v3: ADC APPs

S/N	APP Name	DAVE v3 equivalent	Comments
1	ADC_MEASUREMENT	ADC001 ADCCH001	-
2	ADC_QUEUE	ADC002 ADCCH001	Requires use with XMC ADC LLD to achieve full functionality.
3	ADC_SCAN	ADC003 ADCCH001	
4	GLOBAL_ADC	ADCCMP001 ADCGROUP001	-

Table 9 Mapping DAVE v3 to DAVE v4: ADC APPs

S/N	DAVE v3	Equivalent APP for DAVE v4	Comments
1	ADC001	ADC_MEASUREMENT	-
2	ADC002	ADC_QUEUE	-
3	ADC003	ADC_SCAN	-
4	ADCCH001	ADC_QUEUE ADC_SCAN ADC_MEASUREMENT	-
5	ADCCMP001	GLOBAL_ADC	-
6	ADCGROUP001	GLOBAL_ADC	-

2.2.4 DAC APPs

Table 10 DAVE v4: DAC APPs

S/N	APP Name	Description	Available (2015-02-23)	New
1	DAC	To provide Pattern, Noise and Ramp waveform generation. It can be used in single value and data processing mode.	✓	-
2	DAC_LUT	This is an advance waveform generation using a Look Up Table (LUT).	✓	-

Table 11 Main changes in DAVE v4 from DAVE v3: DAC APPs

S/N	APP Name	DAVE v3 equivalent	Comments
1	DAC	DACWG001 DACWG002 DACWG003	-
3	DAC_LUT	DACWG002 DACWG003	-

Table 12 Mapping DAVE v3 to DAVE v4: DAC APPs

S/N	DAVE v3	Equivalent APP for DAVE v4	Comments
1	DACWG001	DAC	-
2	DACWG002	DAC_LUT	-
3	DACWG003	DAC DAC_LUT DAC_BCCU	-

2.2.5 GPIO APPs

In DAVE v3.1.x there are multiple IO APPs for different I/O function such as general purpose input/output, analogue input, output peripheral connection, and power save mode.

In DAVE v4 this is simplified to a single GPIO APP that can be used as a standalone APP to consume a port pin. It facilitates configuring different I/O functions such as general purpose input/output, analogue input, output peripheral connection, and power save mode.

Table 13 DAVE v4: GPIO APPs

S/N	APP Name	Description	Available (2015-02-23)	New
1	ANALOG_IO	To select port pin as an analog input/output	✓	-
2	DIGITAL_IO	To configure a port pin as a digital input/output	✓	-

Table 14 Main changes in DAVE v4 from DAVE v3: GPIO APPs

S/N	APP Name	DAVE v3 equivalent	Comments
1	ANALOG_IO	IO001	-
2	DIGITAL_IO	IO002 IO004	-

Table 15 Mapping DAVE v3 to DAVE v4: GPIO APPs

S/N	DAVE v3	Equivalent APP for DAVE v4	Comments
1	IO001	ANALOG_IO	-
2	IO002	DIGITAL_IO	-
4	IO004	DIGITAL_IO	-

2.3 System

2.3.1 Generic APPs

Table 16 DAVE v4: Generic APPs

S/N	APP Name	Description	Available (2015-02-23)	New
1	EVENT_DETECTOR	To provide the configuration for Event Request Source and Event Trigger Logic.	✓	-
2	EVENT_GENERATION	To provide the configuration for Output Gating Unit.	✓	-
3	RTC	To provide timing and alarm functions using RTC in the calendar time format.	✓	-
4	SYSTIMER	To provide software timer interface for all non-RTOS applications.	✓	-
5	WATCHDOG	To provide an interface to configure watchdog timer.	✓	-

Table 17 Main changes in DAVE v4 from DAVE v3: Generic APPs

S/N	APP Name	DAVE v3 equivalent	Comments
1	EVENT_DETECTOR	ERU001	-
2	EVENT_GENERATION	ERU002	-
3	RTC	RTC001	-
4	SYSTIMER	SYSTM001 SYSTM002	-
5	WATCHDOG	WDT001	-

Table 18 Mapping DAVE v3 to DAVE v4: Generic APPs

S/N	DAVE v3	Equivalent APP for DAVE v4	Comments
1	ERU001	EVENT_DETECTOR	-
2	ERU002	EVENT_GENERATOR	-
3	RTC001	RTC	-
4	SYSTM001	SYSTIMER	-
5	SYSTM002	SYSTIMER	-
6	WDT001	WATCHDOG	-

2.3.2 Clock System APPs

Table 19 DAVE v4: CLOCK APPs

S/N	APP Name	Description	Available (2015-02-23)	New
1	CLOCK_XMC4	To configure system and peripheral clock for XMC4000 devices	✓	-
2	CLOCK_XMC1	To configure system and peripheral clock for XMC1000 devices	✓	-

Table 20 Main changes in DAVE v4 from DAVE v3: CLOCK APPs

S/N	APP Name	DAVE v3 equivalent	Comments
1	CLOCK_XMC4	CLK001	-
2	CLOCK_XMC1	CLK002	-

Table 21 Mapping DAVE v3 to DAVE v4: CLOCK APPs

S/N	DAVE v3	Equivalent APP for DAVE v4	Comments
1	CLK001	CLOCK_XMC4	-
2	CLK002	CLOCK_XMC1	-

2.3.3 Interrupt APPs

Table 22 DAVE v4: INTERRUPT APPs

S/N	APP Name	Description	Available (2015-02-23)	New
1	INTERRUPT	To allow user to overwrite the provided interrupt service routine (ISR) in system file and to set the interrupt priority. This APP aggregates the *CPU_CTRL_XMCx* APP.	✓	-
2	GLOBAL_SCU_XMC4	To allow to register callback functions and to handle the service request Events. This APP aggregates *CPU_CTRL_XMC4* APP.	✓	-
3	GLOBAL_SCU_XMC1	To allow to register callback functions and to handle the SR0/SR1/SR2 service request events. This APP aggregates *CPU_CTRL_XMC1* APP.	✓	-
4	CPU_CTRL_XMC4	To define the number of bits assigned to pre-empt priority and sub-priority. Not called directly by user.	-	✓
5	CPU_CTRL_XMC1	To define the number of bits assigned to pre-empt priority and sub-priority. Not called directly by user.	-	✓

Table 23 Main changes in DAVE v4 from DAVE v3: INTERRUPT APPs

S/N	APP Name	DAVE v3 equivalent	Comments
1	INTERRUPT	NVIC001 NVIC002	-
2	GLOBAL_SCU_XMC4	NVIC_SCU001	-
3	GLOBAL_SCU_XMC1	NVIC_SR001 NVIC_SR101 NVIC_SR201	-

Table 24 Mapping DAVE v3 to DAVE v4: INTERRUPT APPs

S/N	DAVE v3	Equivalent APP for DAVE v4	Comments
1	NVIC001	INTERRUPT	-
2	NVIC002	INTERRUPT	-
3	NVIC_SCU001	GLOBAL_SCU_XMC4	-
4	NVIC_SR001	GLOBAL_SCU_XMC1	-
5	NVIC_SR101	GLOBAL_SCU_XMC1	-
6	NVIC_SR201	GLOBAL_SCU_XMC1	-

2.4 Motor Control APPs

Table 25 DAVE v4: MOTOR CONTROL APPs

S/N	APP Name	Description	Available (2015-02-23)	New
1	ACIM_FREQ_CTRL	To support Frequency Control- constant V/f Control for the AC Induction Motor.	✓	-
2	AUTOMATION	Provide mechanism to connect two APPs using function block processor. This supports online parameter update and error logging. This consumes System Timer APP and provides task registration feature.	-	✓
3	BLDC_SCALAR_CTRL	To support block commutation using 3Hall/2Hall feedback for Brushless DC motor.	✓	-
4	GLOBAL_POSIF	To configure the POSIF module global settings.	-	✓
5	MOTOR_LIB	This is a library APP and does not provide any UI configurations. This APP only provides common motor control algorithm APIs. This library is used by top level motor APPs.	✓	-
6	HALL_POSIF	To get the motor position and speed using hall sensors separated at 120 degrees.	✓	-
7	PMSM_SCALAR_CTRL	To support PMSM sinusoidal commutation using three or two hall sensors.	✓	-
8	PWM_BC	To generate block commutation PWM waveforms (multichannel pattern) using CCU8 module.	✓	-
9	PWM_SVM	To generate 3-phase Space Vector Pulse Width Modulated outputs using CCU8.	✓	-

Table 26 Main changes in DAVE v4 from DAVE v3: MOTOR CONTROL APPs

S/N	APP Name	DAVE v3 equivalent	Comments
1	ACIM_FREQ_CTRL	ACIMVF01	-
2	BLDC_SCALAR_CTRL	BLDCBCH02 BLDCBCH03 BLDCBCSL01	-
3	MOTOR_LIB	MOTORLIBS	-
4	HALL_POSIF	POSHL001 POSIFH01	-
5	PMSM_SCALAR_CTRL	PMSMSINH02 PMSMSINH03	-

6	PWM_BC	PWMBC01	-
7	PWM_SVM	PWMSVM01	-

Table 27 Mapping DAVE v3 to DAVE v4: MOTOR CONTROL APPs

S/N	DAVE v3	Equivalent APP for DAVE v4	Comments
1	ACIMVF01	ACIM_FREQ_CTRL	-
2	BLDCBCH02	BLDC_SCALAR_CTRL	-
3	BLDCBCH03	BLDC_SCALAR_CTRL	-
4	BLDCBCSL01	BLDC_SCALAR_CTRL	-
5	MOTORLIBS	MOTOR_LIB	-
6	PMSMSINH02	PMSM_SCALAR_CTRL	-
7	PMSMSINH03	PMSM_SCALAR_CTRL	-
8	POSHL001	HALL_POSIF	-
9	POSIFH01	HALL_POSIF	-
10	PWMBC01	PWM_BC	-
11	PWMSVM01	PWM_SVM	-

2.5 Power Conversion APPs

Table 28 DAVE v4: POWER CONVERSION APPs

S/N	APP Name	Description	Available (2015-02-23)	New
1	BUCK_VOLT_CTRL	To control different configurations of a buck converter in voltage mode.	✓	✓
2	GLOBAL_HRPWM	To initializes HRPWM global register set configuration.	✓	-
3	HRPWM	To generate complementary PWM with optional high resolution positioning.	✓	-
4	POWER_CON_LIB	Contains the PI & PID libraries for Digital Power Conversion.	✓	✓
5	PWM_CCU4	PWM generation using one timer slice of CCU4 to generate one PWM output.	✓	-
6	PWM_CCU8	PWM generation using one timer slice of CCU8 to generate up to 4 PWM outputs.	✓	-

Table 29 Main changes in DAVE v4 from DAVE v3: Power Conversion APPs

S/N	APP Name	DAVE v3 equivalent	Comments
1	GLOBAL_HRPWM	HRPWMGLOBAL	-
2	HRPWM	HRC001	-
3	PWM_CCU4	PWMSP001	Added supported for external events. Removed timer concatenation support with use of single CCU slice of CCU for PWM generation. Fixed point implementation and supports duty cycle range from 0.01 to 100.00. Duty cycle value scaled by 100.
4	PWM_CCU8	PWMSP002 PWMSP003	

Table 30 Mapping DAVE v3 to DAVE v4: Power Conversion APPs

S/N	DAVE v3	Equivalent APP for DAVE v4	Comments
1	HRC001	HRPWM	-
2	HRPWMGLOBAL	GLOBAL_HRPWM	-
3	PWMSP001	PWM_CCU4	-
4	PWMSP002	PWM_CCU8	-
5	PWMSP003	PWM_CCU4	-

2.6 Communication APPs

Table 31 DAVE v4: Communication APPs

S/N	APP Name	Description	Available (2015-02-23)	New
1	CAN_NODE	To configure a Node and MO registers of MultiCAN module. It provides a run-time APIs to change the node baud rate, to enable/disable Node and MO interrupts.	✓	-
2	GLOBAL_CAN	To configure global resources of MultiCAN module.	✓	-
3	I2C_MASTER	To implement I2C Protocol. It uses only master mode for communication. IIC uses two bidirectional open-drain lines, Serial Data Line (SDA) and Serial Clock (SCL).	✓	-
4	MANCHESTER_SW	To provide user configuration for Manchester CODEC.	✓	-
5	SPI_MASTER	To implement SPI Master Protocol. It uses only master mode for communication.	✓	-
6	UART	To configure a USIC channel to perform transfer and receive operations through UART protocol.	✓	-

Table 32 Main changes in DAVE v4 from DAVE v3: Communication APPs

S/N	APP Name	DAVE v3 equivalent	Comments
1	CAN_NODE	CAN001 CAN002	-
2	GLOBAL_CAN	CANGLOBAL	-
3	I2C_MASTER	I2C001 I2C002	-
4	MANCHESTER_SW	MANC01	-
5	SPI_MASTER	SPI001 SPI002	-
6	UART	UART001 UART002	-

Table 33 Mapping DAVE v3 to DAVE v4: Communication APPs

S/N	DAVE v3	Equivalent APP for DAVE v4	Comments
1	CAN001	CAN_NODE	-
2	CAN002	CAN_NODE	-

APP changes from DAVE v3 to v4

3	CANGLOBAL	GLOBAL_CAN	-
4	I2C001	I2C_MASTER	-
5	I2C002	I2C_MASTER	-
6	MANC01	MANCHESTER_SW	-
7	SPI001	SPI_MASTER	-
8	SPI002	SPI_MASTER	-
9	UART001	UART	-
10	UART002	UART	-

2.7 Communication

2.7.1 USB APPs

Table 34 DAVE v4: USB APPs

S/N	APP Name	Description	Available (2015-02-23)	New
1	USBD	This is for USB device protocol layer APP. This APP handles the device and endpoint requests from LLD.	✓	-
2	USBD_VCOM	This is for USB virtual COM port application. This APP implements the VCOM over USB CDC class driver.	✓	-
3	USBD_WINUSB	This is a WinUSB device APP that allows the user to interface XMC4XXX devices with windows USB driver (winusb.sys). Users do not have to implement the custom driver on the windows host side.	✓	-

Table 35 Main changes in DAVE v4 from DAVE v3: USB APPs

S/N	APP Name	DAVE v3 equivalent	Comments
1	USBD	USBD	-
2	USBD_VCOM	USBCDC001 USBD_VCOM USBVC001	-
3	USBD_WINUSB	USBD_WINUSB	-

Table 36 Mapping DAVE v3 to DAVE v4: USB APPs

S/N	DAVE v3	Equivalent APP for DAVE v4	Comments
1	USBCDC001	USBD_VCOM	-
2	USBD	USBD	-
3	USBD_VCOM	USBD_VCOM	-
4	USBD_WINUSB	USBD_WINUSB	-
5	USBVC001	USBD_VCOM	-

2.8 Lighting APPs

Table 37 DAVE v4: Lighting APPs

S/N	APP Name	Description	Available (2015-02-23)	New
1	DIM_BCCU	To configure Brightness and Color Control Unit (BCCU) Dimming Engine registers.	✓	-
2	DMX512_RD	To provide user configuration for DMX512 application Stack.	✓	-
3	GLOBAL_BCCU	To configure the global registers of the Brightness and Color Control Unit (BCCU).	✓	-
4	LED_LAMP	To create a virtual lamp with up to 9 channels, with up to 9 PDM_BCCU APPs.	✓	-
5	PDM_BCCU	To configure Brightness and Color Control Unit (BCCU) channel registers.	✓	-

Table 38 Main changes in DAVE v4 from DAVE v3: Lighting APPs

S/N	APP Name	DAVE v3 equivalent	Comments
1	DIM_BCCU	BCCUDIM01	-
2	DMX512_RD	DMX512RD01	-
3	GLOBAL_BCCU	BCCUGLOBAL	-
4	LED_LAMP	COLORLAMP01 LIGHTINGSYS01 WHITELAMP01	-
5	PDM_BCCU	BCCUCH01	-

Table 39 Mapping DAVE v3 to DAVE v4: Lighting APPs

S/N	DAVE v3	Equivalent APP for DAVE v4	Comments
1	BCCUCH01	PDM_BCCU	-
2	BCCUDIM01	DIM_BCCU	-
3	BCCUGLOBAL	GLOBAL_BCCU	-
4	COLORLAMP01	LED_LAMP	-
5	DMX512RD01	DMX512_RD	-
6	LIGHTINGSYS01	LED_LAMP	-
7	WHITELAMP01	LED_LAMP	-

2.9 HMI APPs

Table 40 DAVE v4: HMI APPs

S/N	APP Name	Description	Available (2015-02-23)	New
1	DISPLAY_7SEG	To configure 7-Segment character display with configurable luminance level for LEDs.	✓	-
2	GLOBAL_LEDTS	To allows LEDTS kernels are configured.	✓	-
3	LED_MATRIX	This is a top level APP that supports various application use-cases for different HMI applications.	✓	-
4	LEDTS_COL_CTRL	LED column driving application.	✓	-
5	LEDTS_COLA_CTRL	Configure the brightness and pattern for LED Column resource.	✓	-

Table 41 Main changes in DAVE v4 from DAVE v3: HMI APPs

S/N	APP Name	DAVE v3 equivalent	Comments
1	DISPLAY_7SEG	DISP_7SEG01	-
2	GLOBAL_LEDTS	LEDTSGLOBAL	-
3	LED_MATRIX	DOTMATRIX01 LEDMATRIX01	-
4	LEDTS_COL_CTRL	COL01	-
5	LEDTS_COLA_CTRL	COLA01	-

Table 42 Mapping DAVE v3 to DAVE v4: HMI APPs

S/N	DAVE v3	Equivalent APP for DAVE v4	Comments
1	COL01	LEDTS_COL_CTRL	-
2	COLA01	LEDTS_COLA_CTRL	-
3	DISP_7SEG01	DISPLAY_7SEG	-
4	DOTMATRIX01	LED_MATRIX	-
5	LEDMATRIX01	LED_MATRIX	-
6	LEDTSGLOBAL	GLOBAL_LEDTS	-

3 Moving your project to DAVE v4

There is no essential need to move your old DAVE v3 projects because DAVE v3 will still be available and maintained for a significant period of time. However, if you do want to adopt the improvements in v4, it is possible to manually move projects between the two versions.

To move a project to DAVE v4, it is important to understand the current project requirements and functions in DAVE v3.

What to do in DAVE v3: Understand the project settings

- Step 1: Find MCU selected for the project.
- Step 2: Find the DAVE3 APPs used for the project.
- Step 3: Find the project resource used: Pins.
- Step 4: Find the project resource used: Signals.
- Step 5: Find the APP settings.
- Step 6: Find the API used and program flow.

What to do in DAVE v4: Create a new project

- **Step 1: Create a new DAVE project.**
- Step 2: Add the DAVE APPs.
- Step 3: Configure the APP settings.
- Step 4: Configure the project resource used: Signals.
- Step 5: Configure the project resource used: Pins.
- Step 6: Update the code.
- Step 7: Generate the code and compile.

3.1 DAVE v3: Understand the project settings

The selected DAVE v3 project (LED_BLINKING) toggles the connected LED. The speed of the LED toggling is adjusted by the potentiometer and updated only when button1 is pressed.

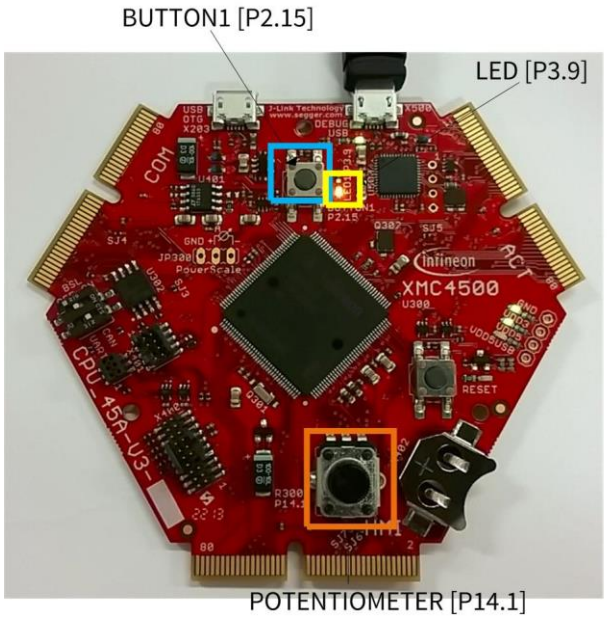


Figure 12 DAVE v3 Project: LED_BLINKING

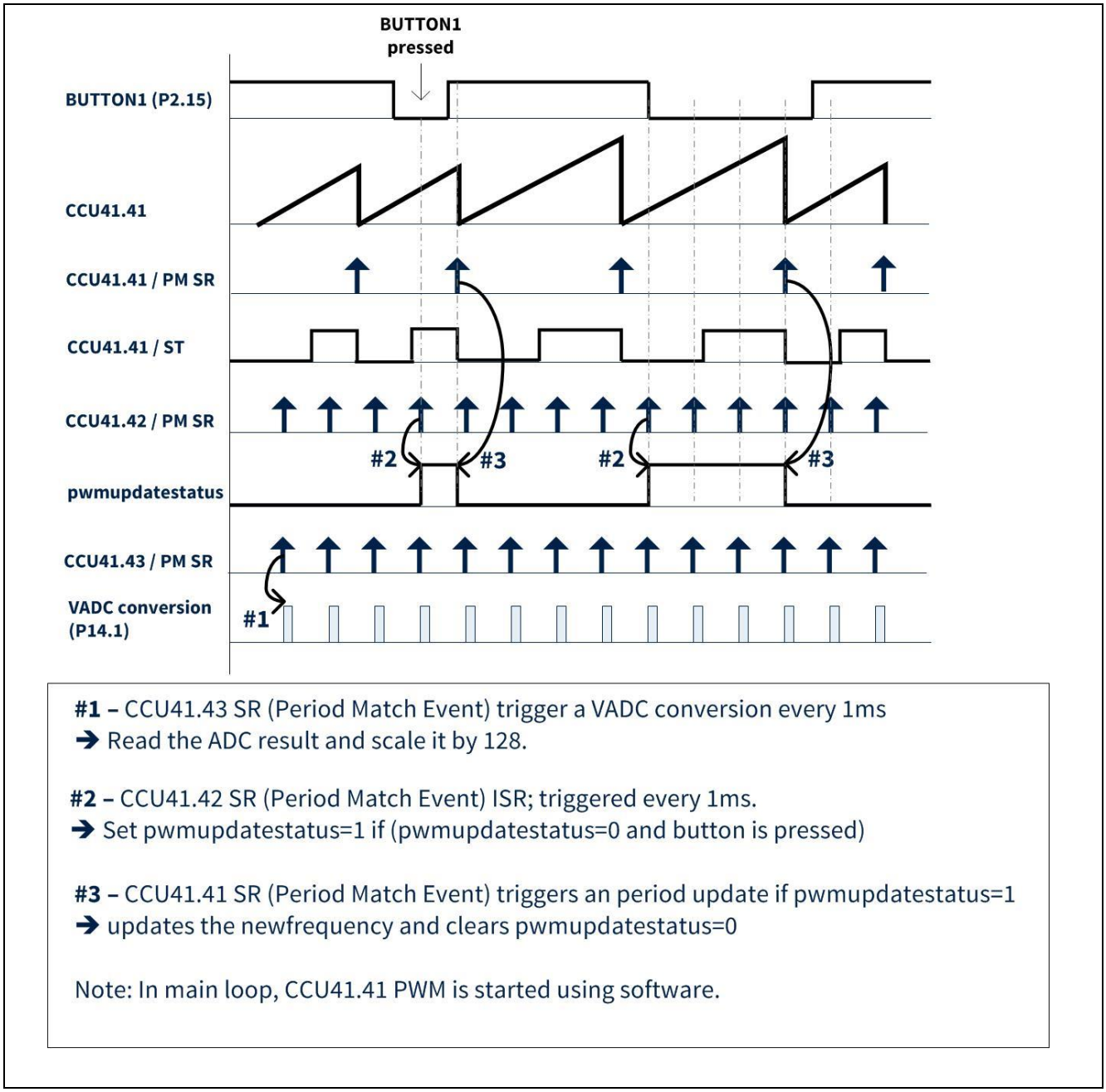


Figure 13 DAVE v3 Project: LED_BLINKING timing diagram

3.1.1 Step 1: Find MCU selected for the project

Open the DAVE v3 project to be ported to v4 and determine the device that is used, using the steps provided in the following figure:

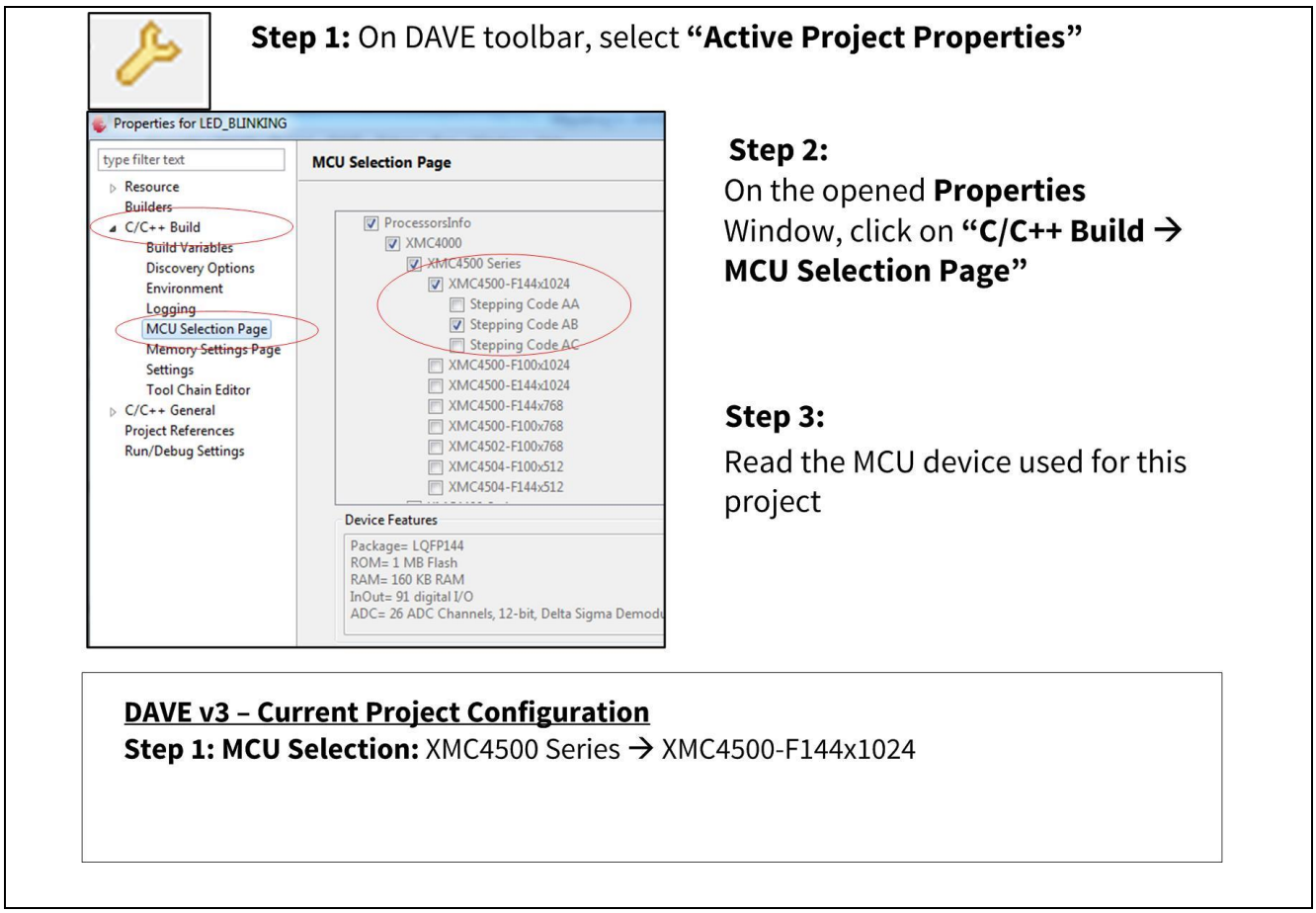


Figure 14 Determine the MCU used in the DAVE project

3.1.2 Step 2: Find the DAVE3 APPs used for the project

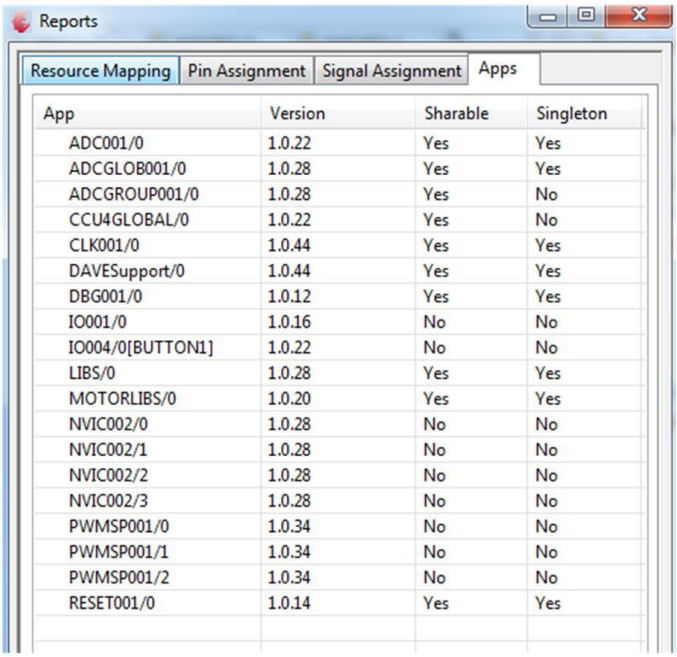
In the DAVE project, all the APPs used in the Project are listed. However, not all the APPs need to be considered when migrating to DAVE v4.

In most instances, only the top-level APPs are required (aggregated APPs do not need to be considered).

To determine the top-level APPs used in the DAVE v3 project, refer to the **S/W APP Dependency Tree View**.

Click on **"Resource Mapping Information"** on the DAVE toolbar.

This opens up a pop-up Dialog " Reports"



App	Version	Sharable	Singleton
ADC001/0	1.0.22	Yes	Yes
ADCGLOB001/0	1.0.28	Yes	Yes
ADCGROUP001/0	1.0.28	Yes	No
CCU4GLOBAL/0	1.0.22	Yes	No
CLK001/0	1.0.44	Yes	Yes
DAVESupport/0	1.0.44	Yes	Yes
DBG001/0	1.0.12	Yes	Yes
IO001/0	1.0.16	No	No
IO004/0[BUTTON1]	1.0.22	No	No
LIBS/0	1.0.28	Yes	Yes
MOTORLIBS/0	1.0.20	Yes	Yes
NVIC002/0	1.0.28	No	No
NVIC002/1	1.0.28	No	No
NVIC002/2	1.0.28	No	No
NVIC002/3	1.0.28	No	No
PWMSP001/0	1.0.34	No	No
PWMSP001/1	1.0.34	No	No
PWMSP001/2	1.0.34	No	No
RESET001/0	1.0.14	Yes	Yes

Click on tab **"Apps"** in the **"Reports"** Window

These are the list of APPs used in the Project, including aggregated APPs.

Figure 15 List of APPs used in a project, from Resource Mapping Information

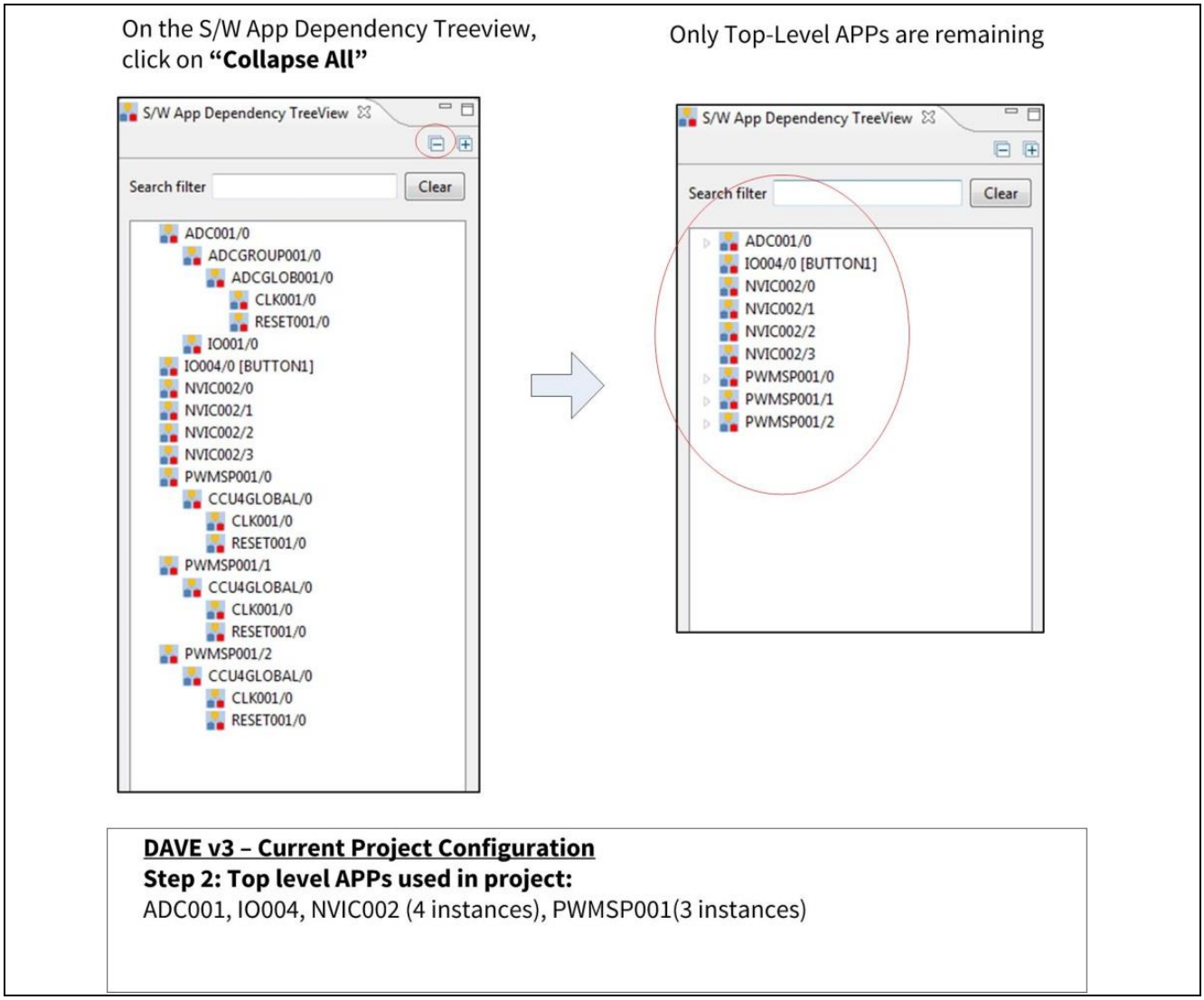


Figure 16 Determine the APPs from S/W APP Dependency Tree

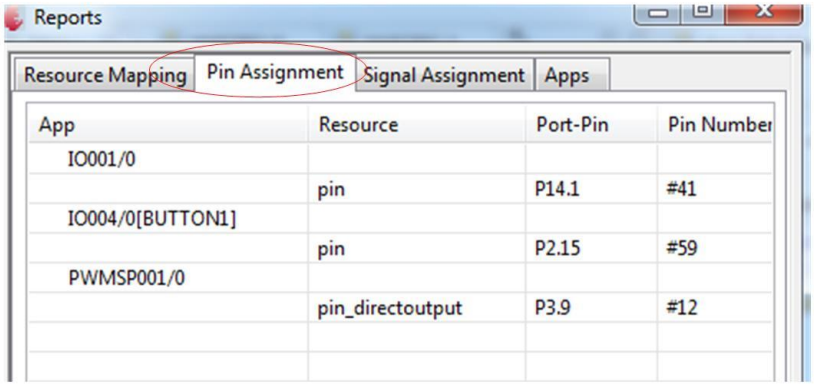
3.1.3 Step 3: Find the project resource used: Pins

Resource Mapping Information gives information on the Pin Assignment and Signal Assignment.

As the DAVE v4 APPs are new, it is important to note the signal assignments in DAVE v3 APPs. This is to understand the differences in the APPs and apply the necessary codes to the migrated project.

Click on **"Resource Mapping Information"** on the DAVE toolbar.

This opens up a pop-up Dialog " Reports"



Click on tab **"Pin Assignment"** in the **"Reports"** Window

These are the list of pins used in the project

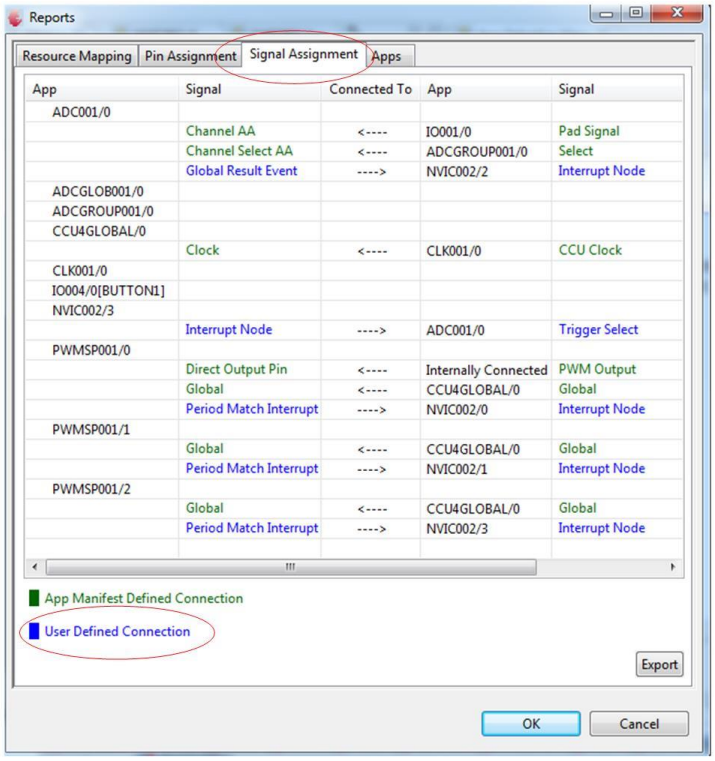
DAVE v3 – Current Project Configuration
Step 3: Pin Assignment:
IO001/0 → pin : P14.1
IO004/0 → pin : P2.15
PWMSP001/0 → pin : P3.9

Figure 17 Determine the Pin and Signal Assignment

3.1.4 Step 4: Find the project resource used: Signals

Click on **"Resource Mapping Information"** on the DAVE toolbar.

This opens up a pop-up Dialog " Reports"



Click on tab **"Signal Assignment"** in the **"Reports"** Window

These are the signal connections between APPs in the project

Record the **User Defined Connection**

DAVE v3 – Current Project Configuration

Step 4: Signal Assignment

ADC001/0 "Global Result Event" is connected to NVIC002/2 "Interrupt Node"
NVIC002/3 "Interrupt Node" is connected to ADC001/0 "Trigger Select"
PWMSP001/0 "Period Match Interrupt" is connected to NVIC002/0 "Interrupt Node"
PWMSP001/1 "Period Match Interrupt" is connected to NVIC002/1 "Interrupt Node"
PWMSP001/2 "Period Match Interrupt" is connected to NVIC002/3 "Interrupt Node"

Figure 18 Determine the Signal Assignment

3.1.5 Step 5: Find the APP settings

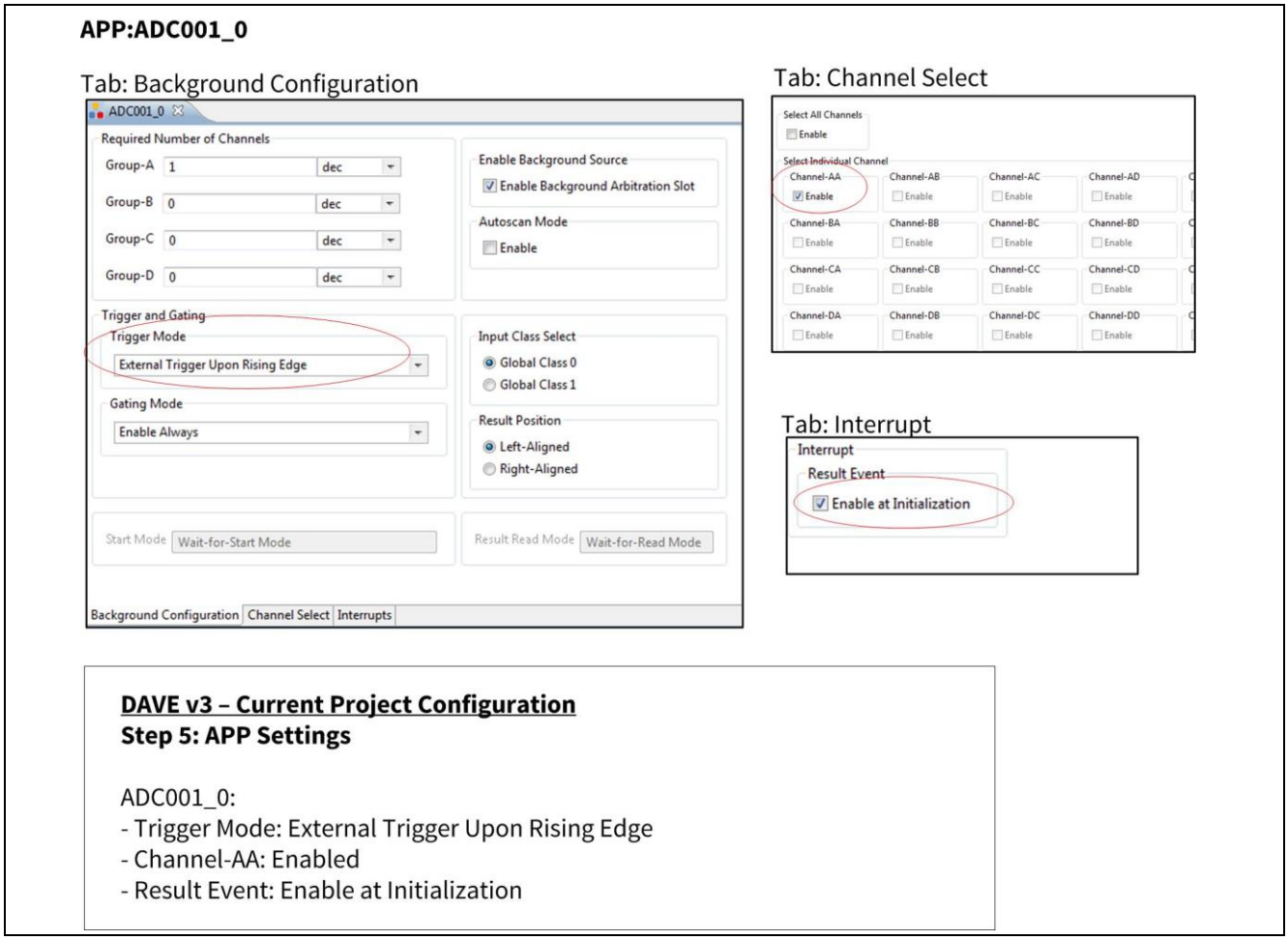


Figure 19 Determine the APP Settings for ADC001 APP

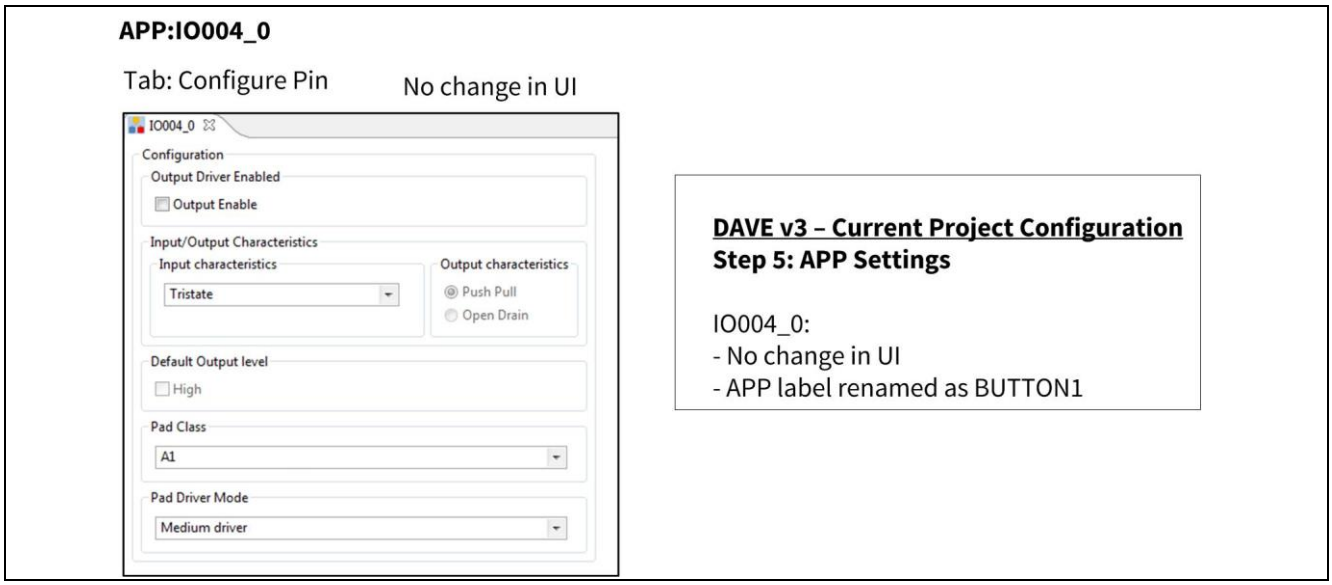


Figure 20 Determine the APP Settings for IO004 APP

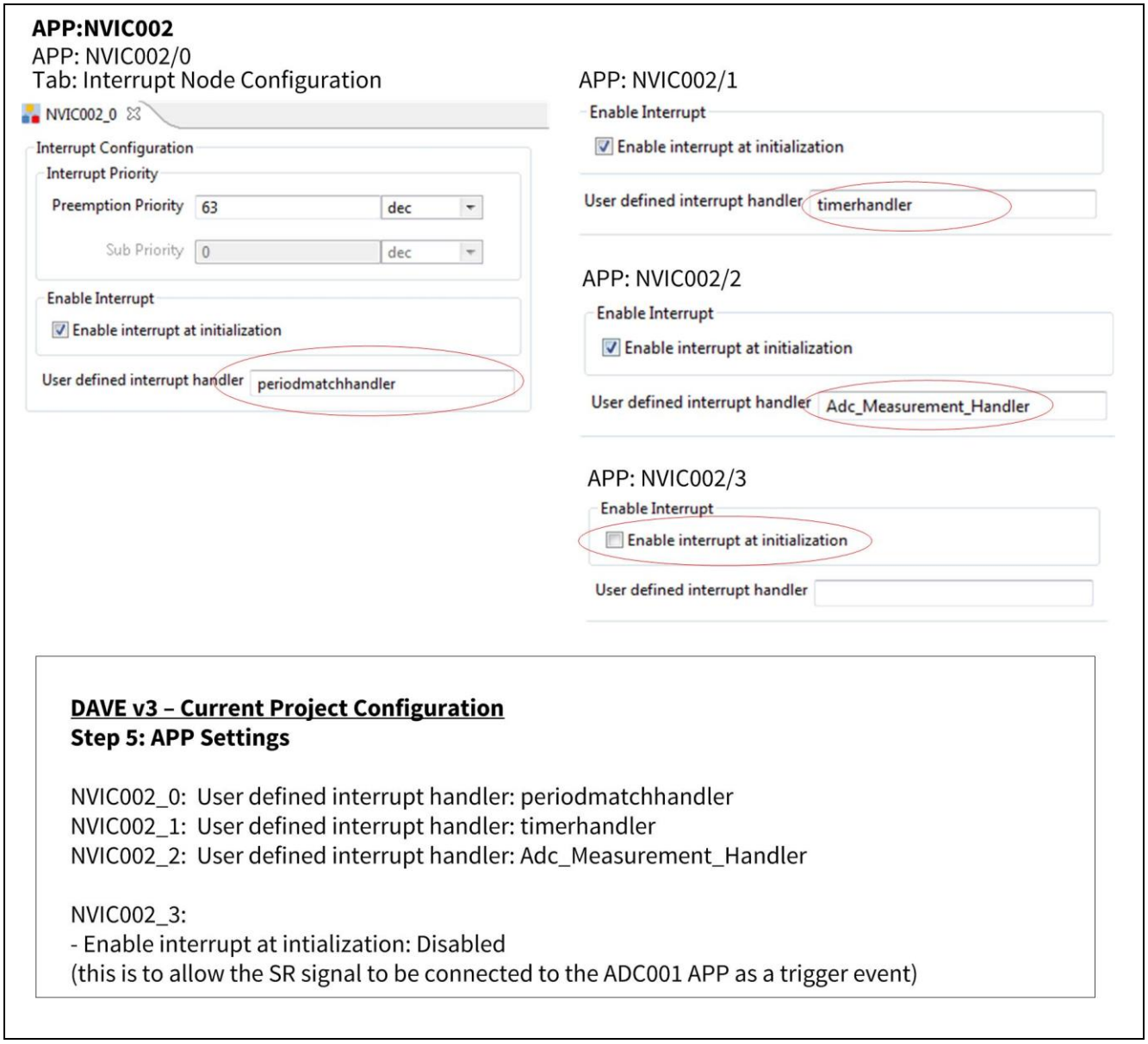


Figure 21 Determine the APP Settings for NVIC002 APP

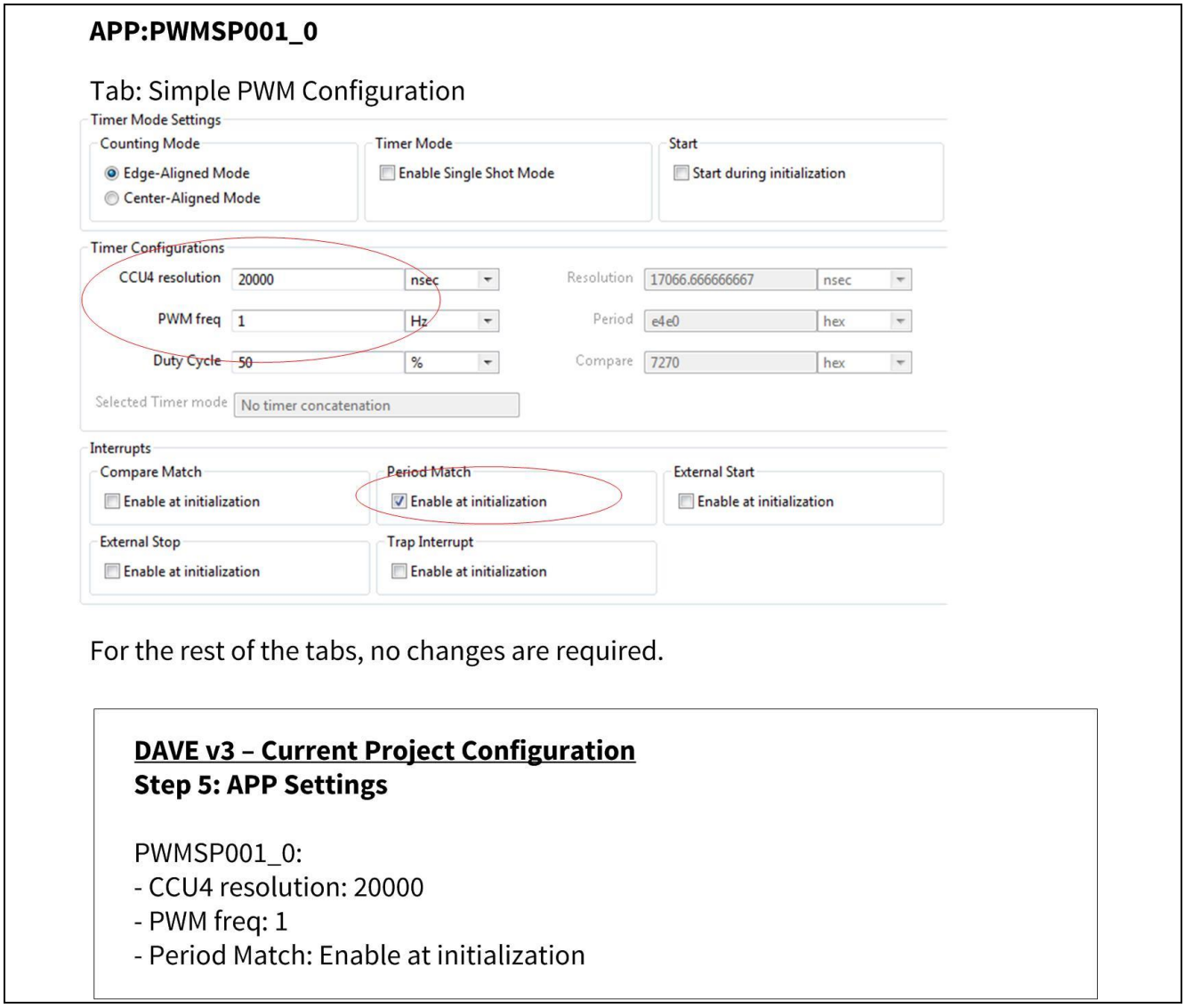


Figure 22 Determine the APP Settings for PWMSP001/0 APP

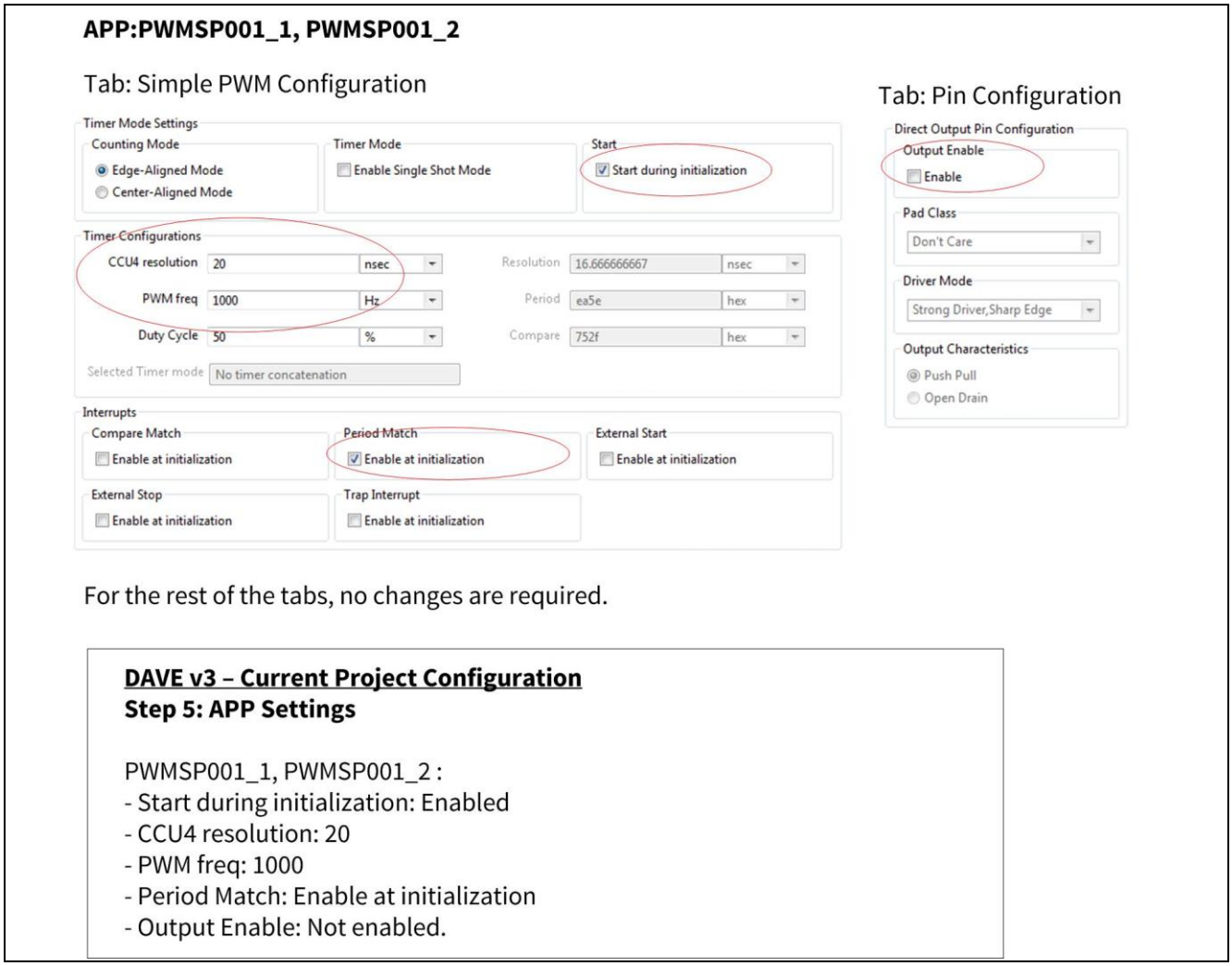


Figure 23 Determine the APP Settings for PWMSP001 APP

3.1.6 Step 6: Find the API used and program flow

```
//Declarations from DAVE3 Code Generation (includes SFR declaration)
#include <DAVE3.h>

ADC001_ResultHandleType result;
volatile float newfrequency=1;
bool pwmupdatestatus=0;
```

DAVE v3 – Current Project Configuration
Step 6: API used and program flow

Variables used:
- ADC001_ResultHandleType Result

Figure 24 Determine the APP variables used: Initialization

```
/**
 * @brief periodmatchhandler() - PWM_0 ISR handler
 *
 * <b>Details of function</b><br>
 * This routine performs a pwm frequency update when pwmupdatestatus is set
 */
void periodmatchhandler(void)
{
        if(pwmupdatestatus)
        {
                PWMSP001_ClearPendingEvent(&PWMSP001_Handle0, PWMSP001_PERIODMATCHEVENT);
                PWMSP001_SetPwmFreqAndDutyCycle(&PWMSP001_Handle0, newfrequency, 50.0f);
                pwmupdatestatus=0;      //clear the status for PWM update
        }
}
```

DAVE v3 – Current Project Configuration
Step 6: API used and program flow

APIs used in **periodmatchhandler:**
- PWMSP001_ClearPendingEvent
- PWMSP001_SetPwmFreqAndDutyCycle

Figure 25 Determine the APP APIs used interrupt handlers: periodmatchhandler(void)

```
/**
 * @brief timerhandler() - TIMER_1 ISR Handler
 *
 * <b>Details of function</b><br>
 * This routine requests for PWM frequency update only if button is pressed and no
 * on-going PWM update
 */
void timerhandler(void)
{
        /* update speed of toggling only when button is pressed*/
        if(((pwmupdatestatus==0) && (IO004_ReadPin(IO004_Handle0)) == 0))
        {
                pwmupdatestatus=1;
        }
}
```

DAVE v3 – Current Project Configuration
Step 6: API used and program flow

APIs used in **timerhandler:**
- IO004_ReadPin

Figure 26 Determine the APP APIs used interrupt handlers: timerhandler(void)

DAVE v3 – Current Project Configuration
Step 6: API used and program flow

APIs used in **ADC_Measurement_Handler:**
- ADC001_GetResult

```
/**
 * @brief Adc_Measurement_Handler()
 * - Routine entered when the ADC result is  available
 *
 * <b>Details of function</b><br>
 * This routine reads the ADC result and scale it by 128 and keeps the
 * minimum frequency at 1Hz and maximum frequency at 31Hz
 */
void Adc_Measurement_Handler(void)
{
        static float tempfreq;
        ADC001_GetResult(&ADC001_Handle0, &result);

        /* Divide the ADC result by 128; Freq(min) = 1Hz, Freq(max) = 31Hz.  */
        tempfreq = (float)(result.Result>>7);

        if(pwmupdatestatus==0)
        {
                if(tempfreq < 1.0f)
                {
                        newfrequency = 1.0f;
                }
                else
                {
                        newfrequency=tempfreq;
                }
        }
}
```

Figure 27 Determine the APP APIs used interrupt handlers: ADC_Measurement_Handler(void)

DAVE v3 – Current Project Configuration
Step 6: API used and program flow

APIs used in **main**:
- PWMSP001_Start

```
int main(void)
{//     status_t status;                    // Declaration of return variable for DAVE3 APIs (toggle comment if required)
        DAVE_Init();                        // Initialization of DAVE APPs
        PWMSP001_Start(&PWMSP001_Handle0); /*Start PWM Slice*/
        while(1)
        {
        }
        return 0;
}
```

Figure 28 Determine the APP APIs used: int main(void)

3.2 In DAVE v4

With the information on the settings of the DAVE v3 project understood, the move to DAVE v4 can begin.

3.2.1 Step 1: Create a new DAVE project

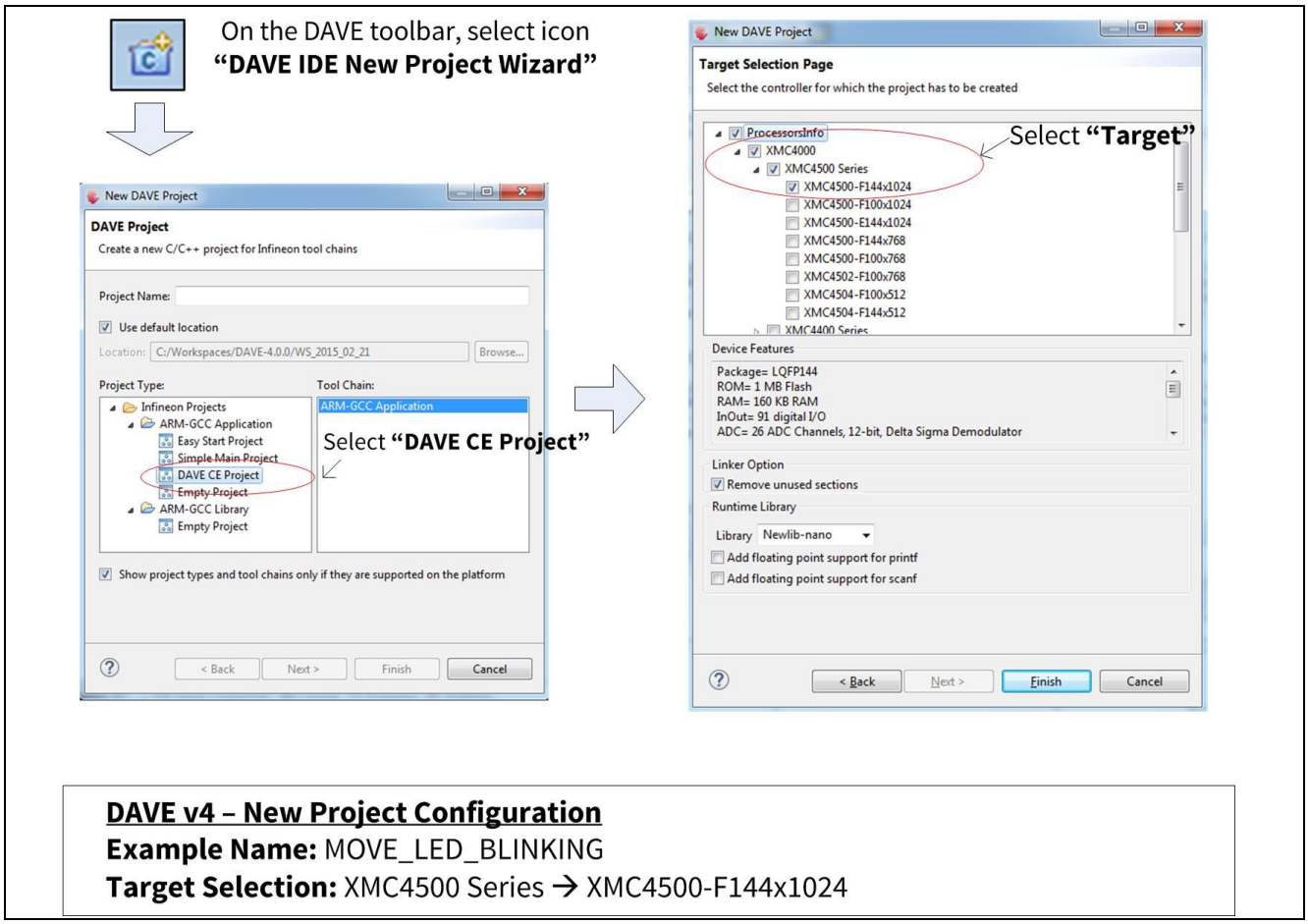


Figure 29 Create a new DAVE CE project

3.2.2 Step 2: Add the DAVE APPs

Find the compatible DAVE APPs for those used in your v3 project.

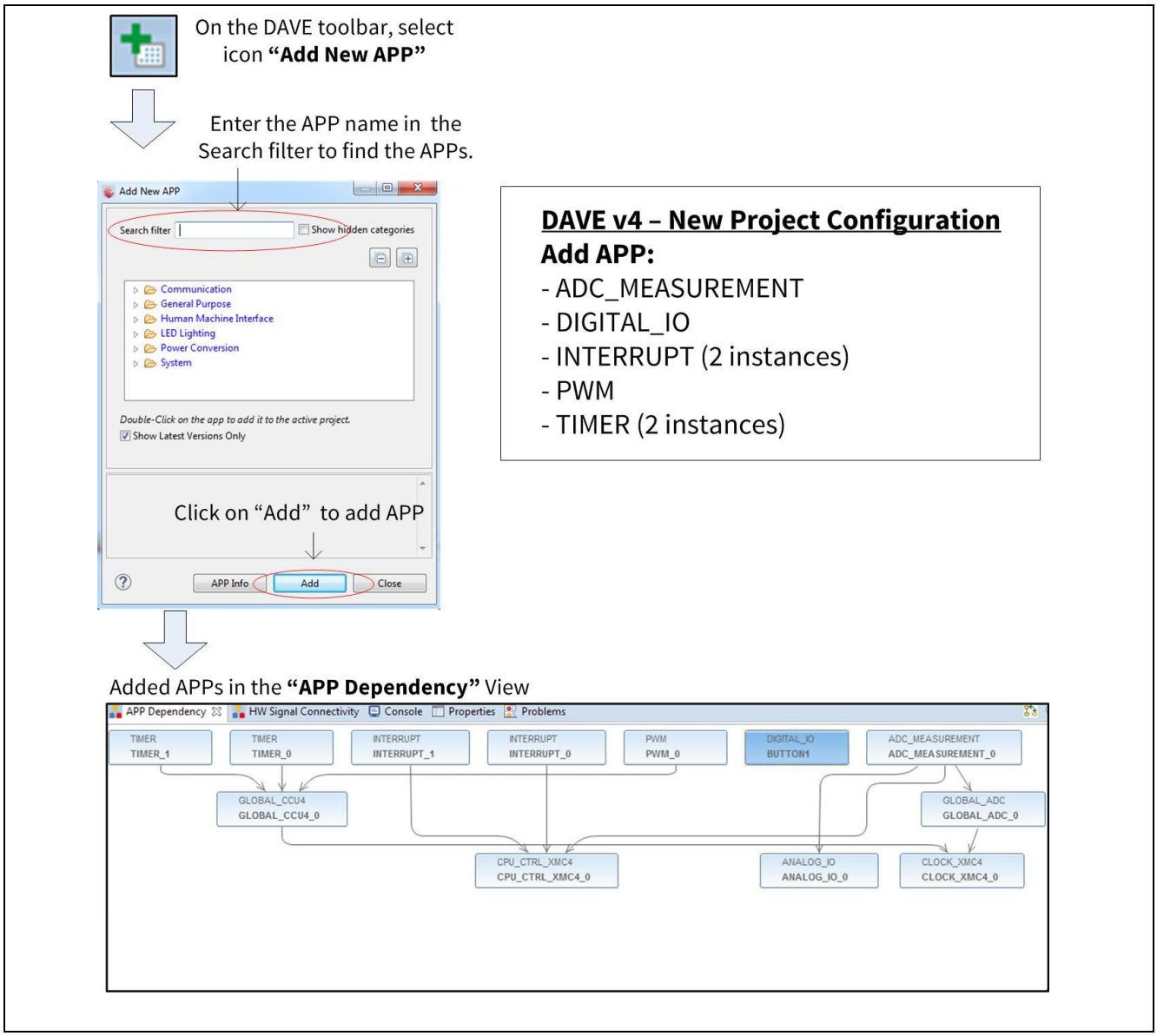


Figure 30 Add new APPs

3.2.3 Step 3: Configure the APP settings

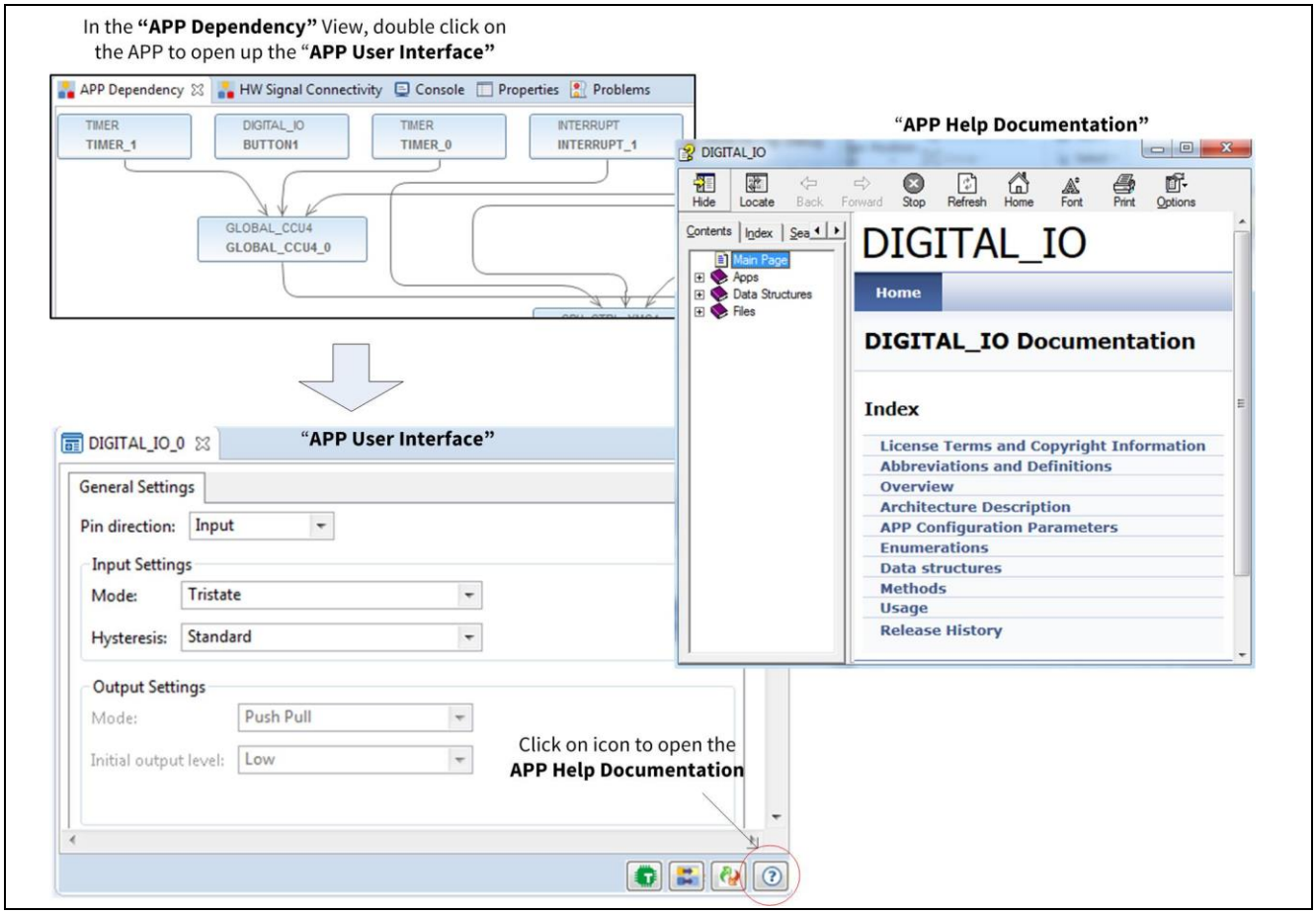


Figure 31 Get the Help Documentation

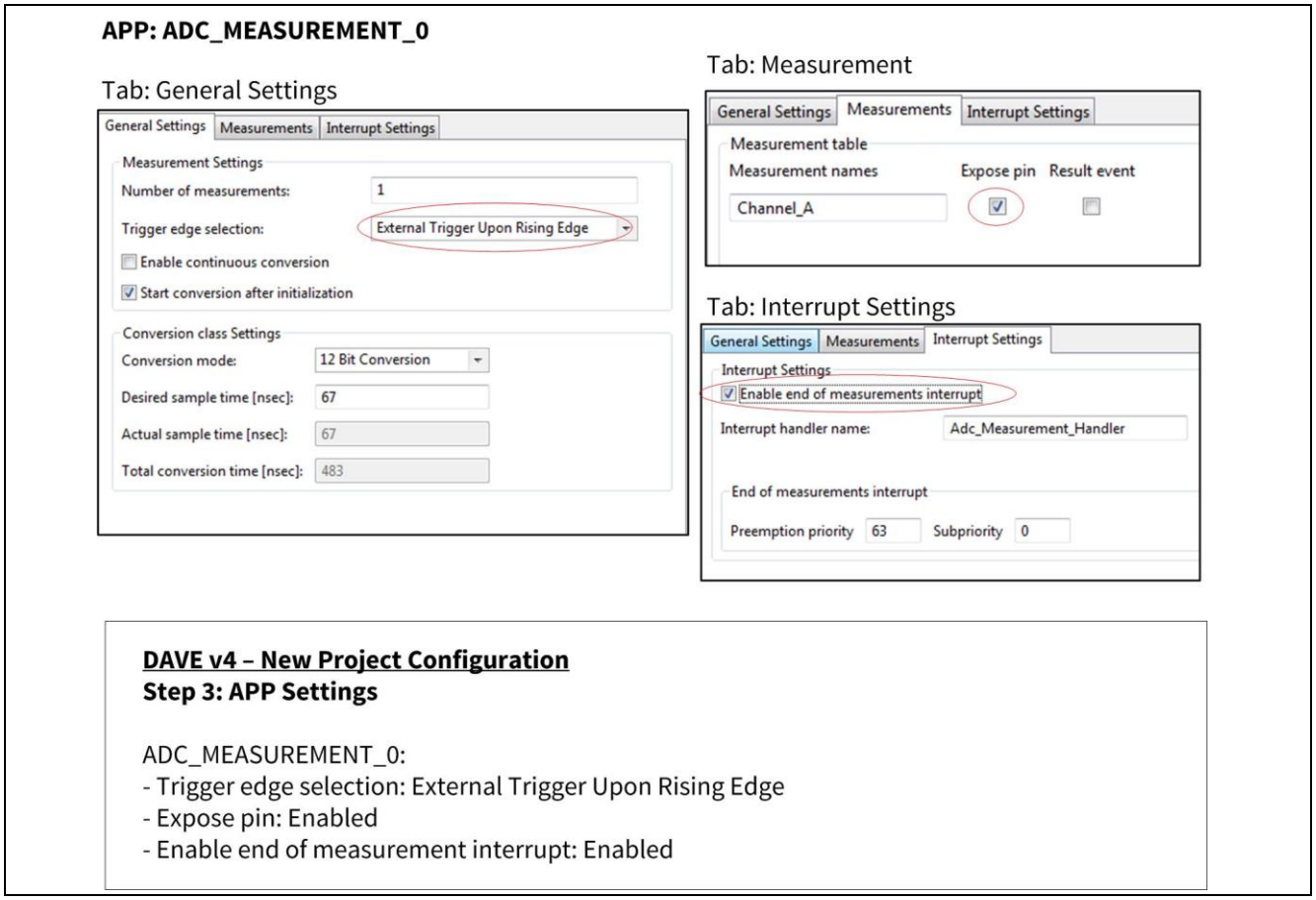


Figure 32 Configure the APP: ADC_MEASUREMENT

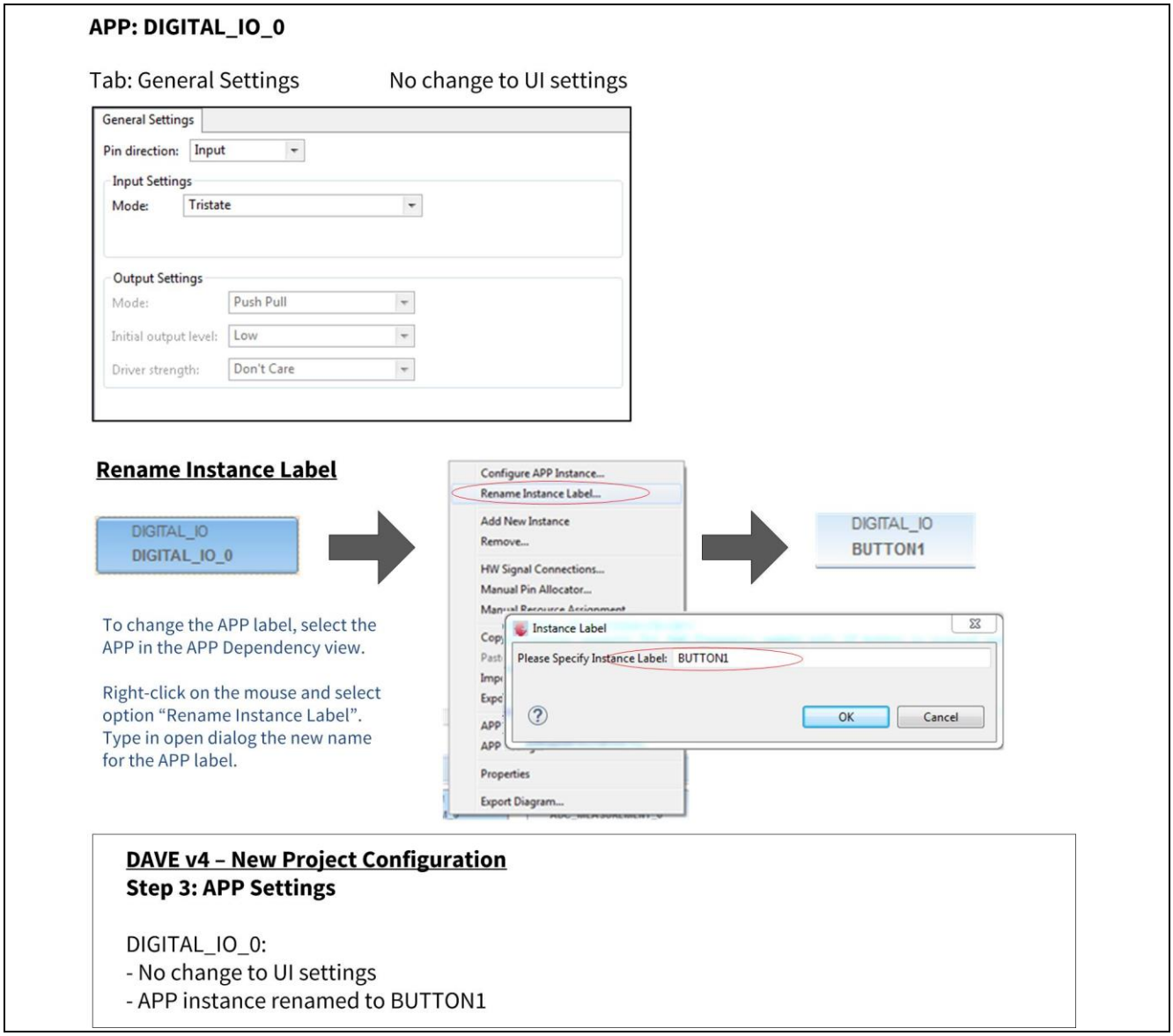


Figure 33 Configure the APP: DIGITAL_IO

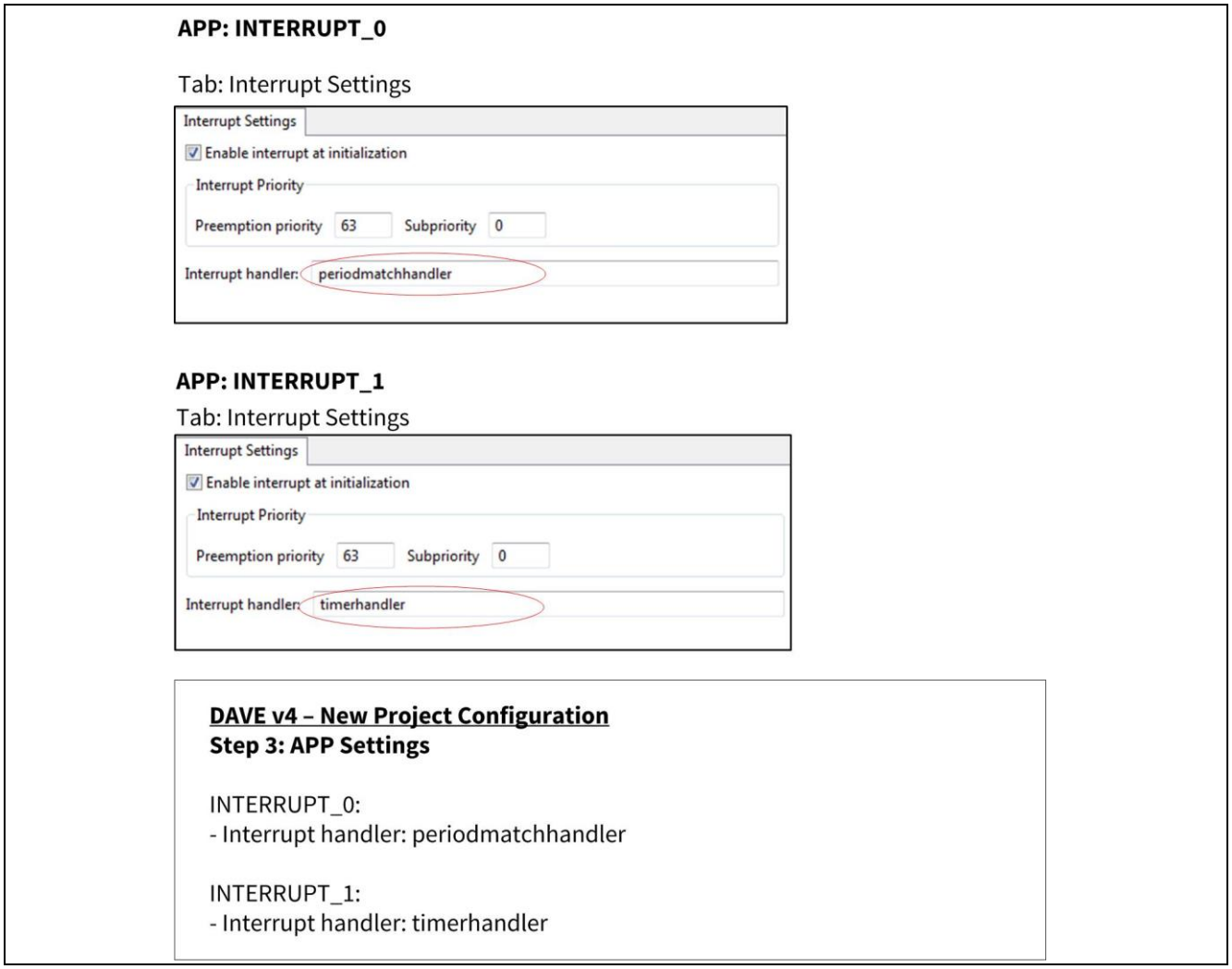


Figure 34 Configure the APP: INTERRUPT

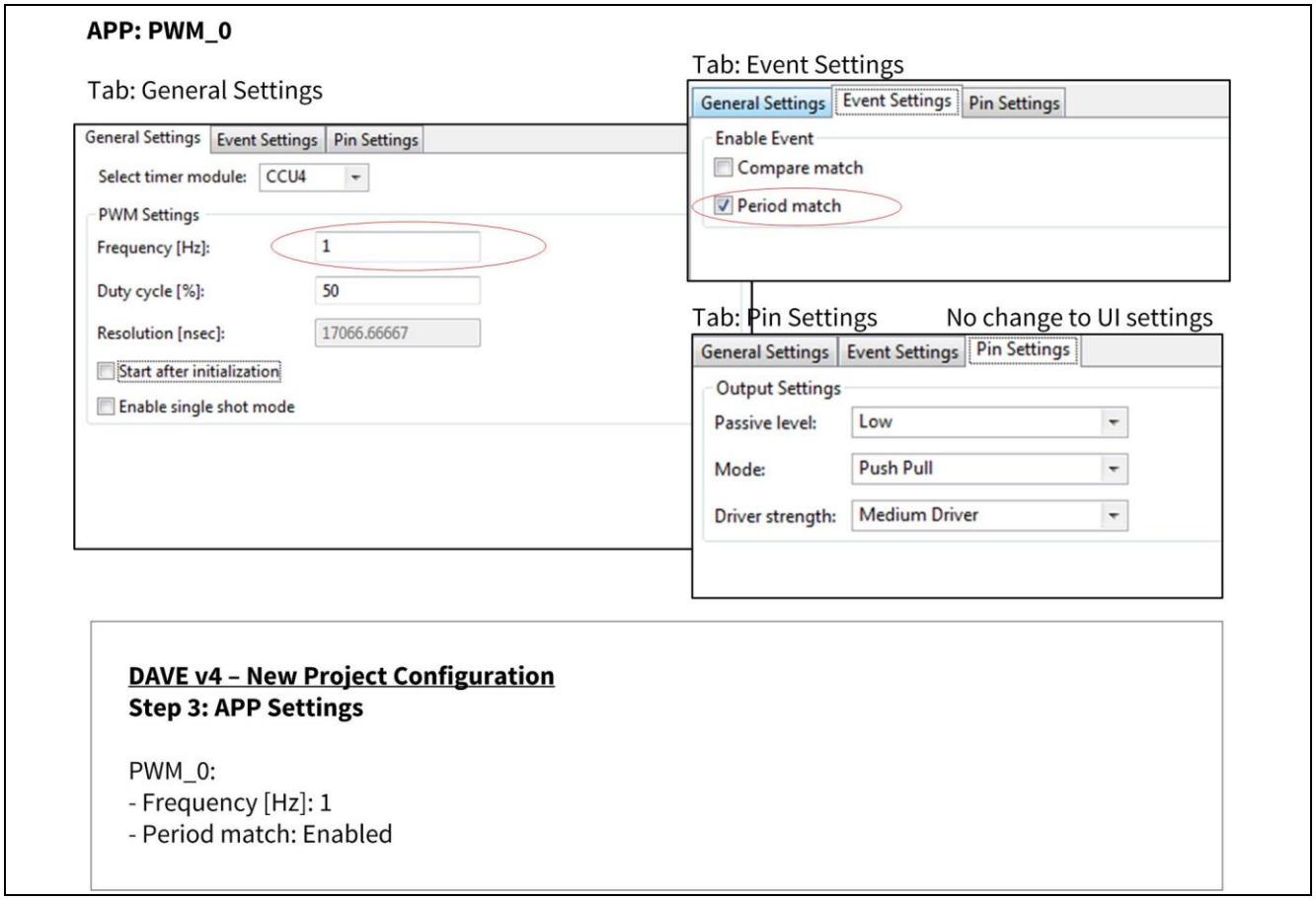


Figure 35 Configure the APP: PWM

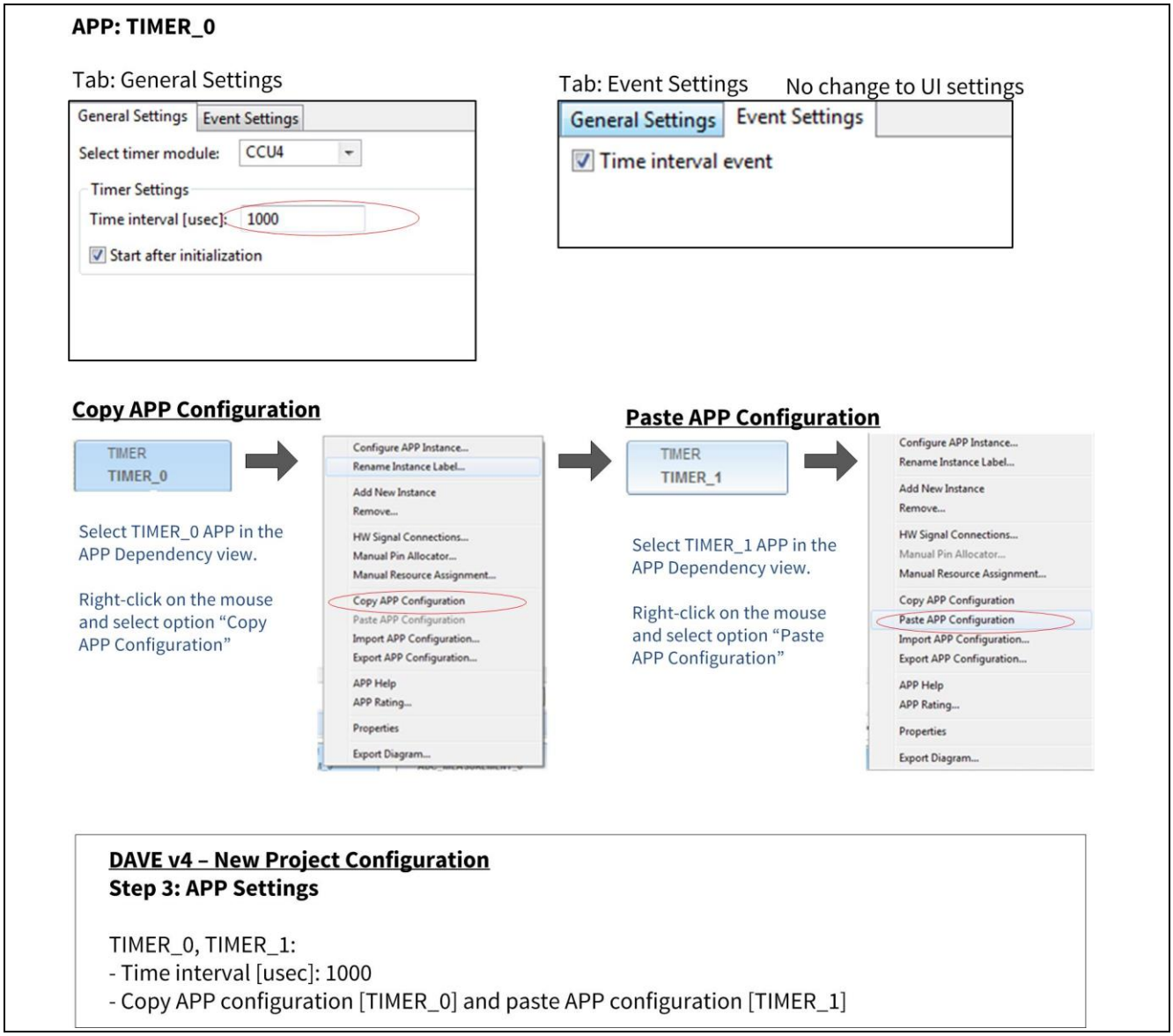


Figure 36 Configure the APP: TIMER

3.2.4 Step 4: Configure the project resource used: Signals

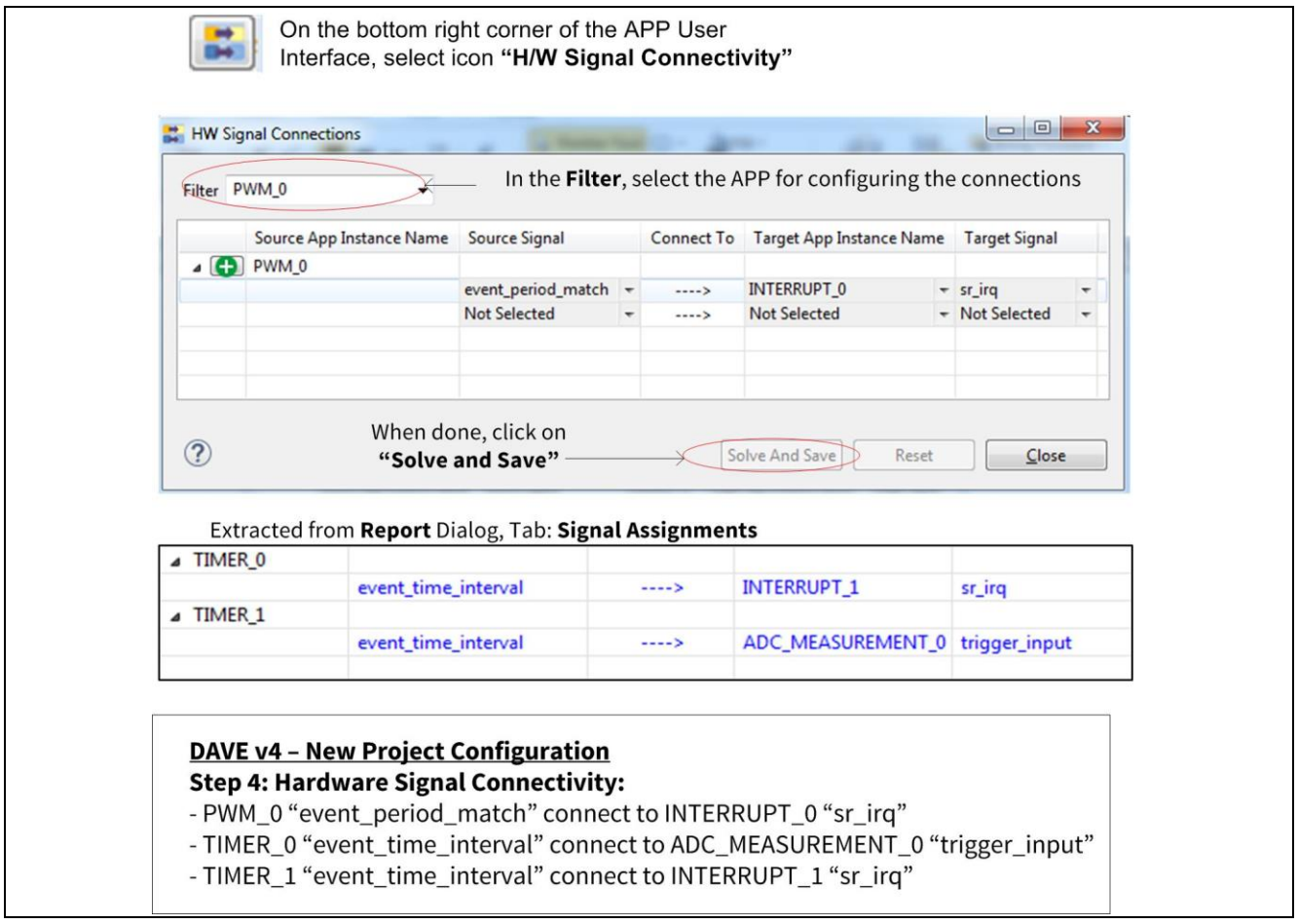


Figure 37 Signal Connection

3.2.5 Step 5: Configure the project resource used: Pins

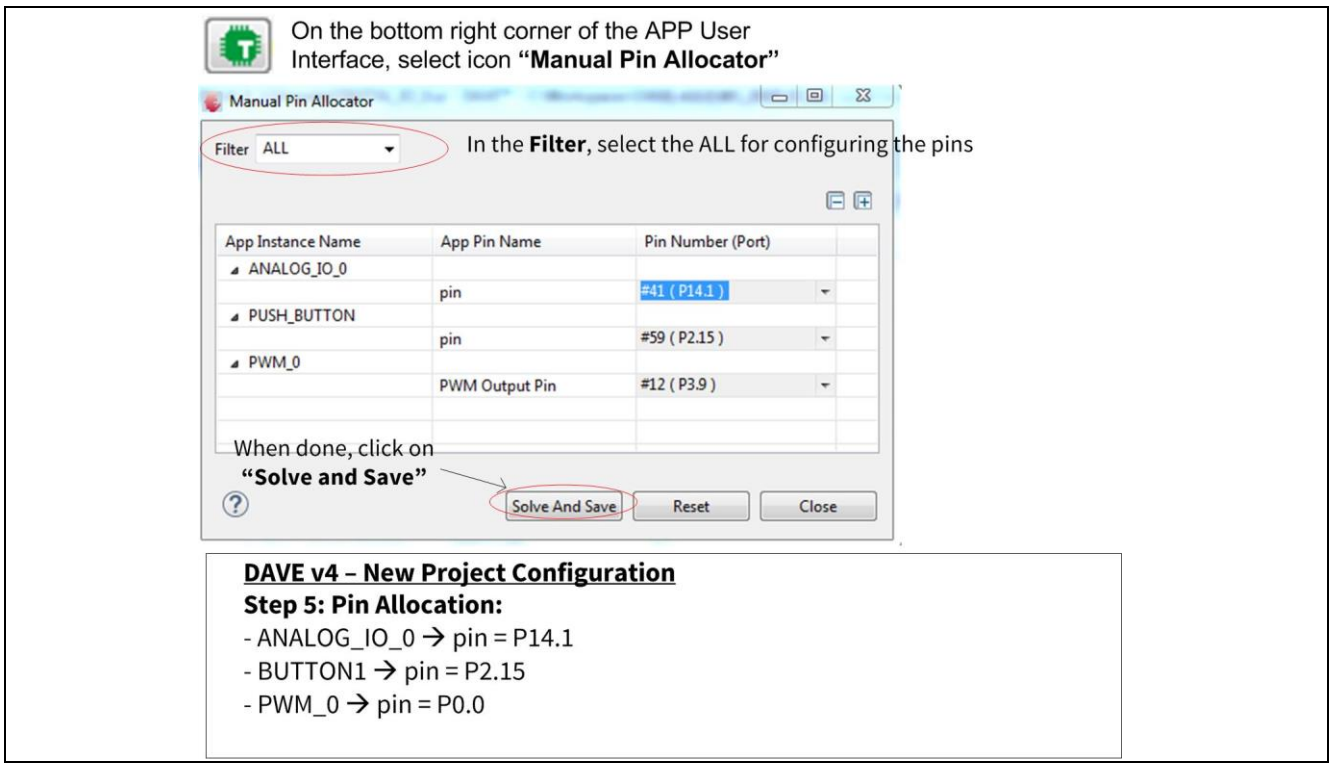


Figure 38 Manual Pin Assignment using Manual Pin Allocator

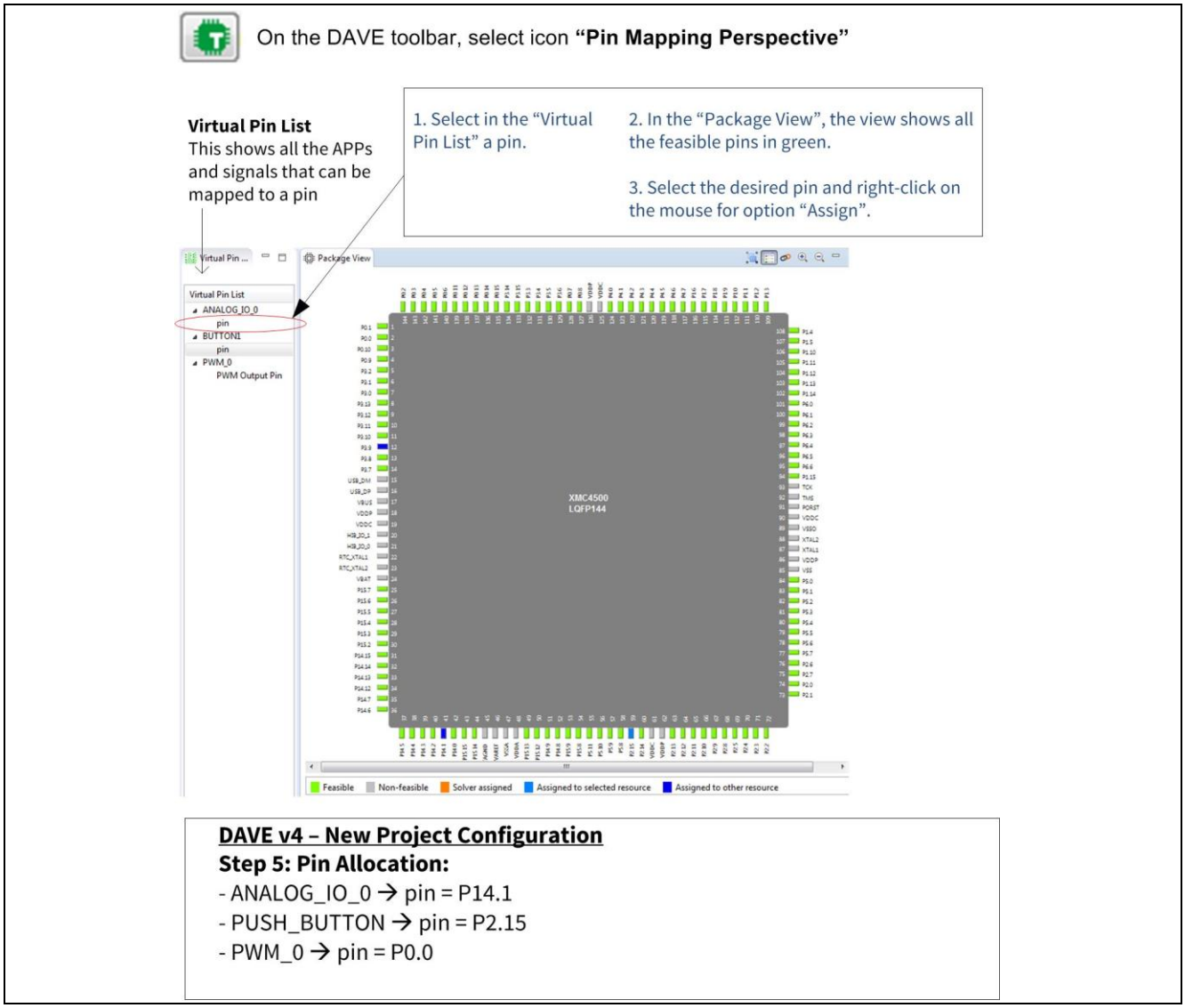


Figure 39 Manual Pin Assignment using Pin Mapping Perspective

3.2.6 Step 6: Update the code

DAVE v4 – New Project Configuration
Step 6: Update the code

Variables used:
-XMC_VADC_RESULT_SIZE_t
-uint32_t newfrequency

```
#include <XMC4500.h>
#include <DAVE.h>          //Declarations from DAVE Code Generation (includes SFR declaration)

volatile XMC_VADC_RESULT_SIZE_t result;
volatile uint32_t newfrequency=1;
bool pwmupdatestatus=0;
```

Figure 40 Update the variables: Initialization

DAVE v4 – New Project Configuration
Step 6: Update the code

APIs used:
- PWM_AcknowledgeInterrupt
- PWM_SetFreq
(Duty cycle is always maintained. Hence, it is not required for changing the duty cycle in the API)

```
/**
 * @brief periodmatchhandler() - PWM_0 ISR handler
 * <b>Details of function</b><br>
 * This routine performs a pwm frequency update when
 * pwmupdatestatus is set
 */

void periodmatchhandler(void)
{
        if(pwmupdatestatus)
        {
                PWM_AcknowledgeInterrupt(&PWM_0, PWM_PERIODMATCH_INTERRUPT);
                PWM_SetFreq(&PWM_0, newfrequency);
                pwmupdatestatus=0;      //clear the status for PWM update
        }
}
```

Figure 41 Update the API in interrupt handler: periodmatchhandler(void)

DAVE v4 – New Project Configuration
Step 6: Update the code

APIs used:
- DIGITAL_IO_GetInput

```
/**
 * @brief timerhandler() - TIMER_1 ISR Handler
 * <b>Details of function</b><br>
 * This routine requests for PWM frequency update only if button is
 * pressed and no PWM update is ongoing
 */
void timerhandler(void)
{
        /* update speed of toggling only when button is pressed*/
        if((pwmupdatestatus==0) && (DIGITAL_IO_GetInput(&BUTTON1) == 0))
        {
                pwmupdatestatus=1;      }
}
```

Figure 42 Update the API in interrupt handler: timerhandler(void)

DAVE v4 – New Project Configuration
Step 6: Update the code

APIs used:
- ADC_MEASUREMENT_GetResult

```
/**
 * @brief Adc_Measurement_Handler() - Routine entered when the
 * ADC result is available
 *
 * <b>Details of function</b><br>
 * This routine reads the ADC result and scale it by 128 and keeps the
 * minimum frequency at 1Hz and maximum frequency at 31Hz
 */
void Adc_Measurement_Handler(void)
{
        static uint32_t tempfreq;

        if(pwmupdatestatus==0)
        {
                result = ADC_MEASUREMENT_GetResult(&ADC_MEASUREMENT_Channel_A_handle);

                /* Divide the ADC result by 128; Freq(min) = 1Hz, Freq(max) = 31Hz.  */
                tempfreq = result >> 7;
                if(tempfreq == 0)
                {
                        newfrequency = 1;
                }
                else
                {
                        newfrequency=tempfreq;
                }
        }
}
```

Figure 43 Update the API in interrupt handler: ADC_Measurement_Handler(void)

DAVE v4 – New Project Configuration
Step 6: Update the code

APIs used:
- PWM_Start

```
/**
 * @brief main() - application entry point
 *
 * <b>Details of function</b><br>
 * This routine is the application entry point. It starts the connected
 * PWM slice.
 */
int main(void)
{
 DAVE_STATUS_t status;
 status = DAVE_Init();                  /* Initialization of DAVE APPs  */

 if(status == DAVE_STATUS_SUCCESS)
 {
  XMC_DEBUG("DAVE APPs initialization success\n");
 }
 else
 {
  /* Placeholder for error handler code. The while loop below can be replaced with an user error handler */
  XMC_DEBUG(("DAVE APPs initialization failed with status %d\n", status));
  while(1U)
```

```
  {
  }
 }

 PWM_Start(&PWM_0);  /*Start PWM Slice*/

 /* Placeholder for user application code. The while loop below can be replaced with user application code. */
 while(1U)
 {
 }
 return 1;
}
```

Figure 44 Update the API: int main (void)

3.2.7 Step 7: Generate the code and compile

On the DAVE toolbar, select icon **“Generate Code”**

Once the code generation is completed, on the DAVE toolbar, select icon **“Build Active Project”**

Figure 45 Configure the code

Congratulations! You have successfully moved your project to DAVE v4.

4 XMC Low Level Drivers (LLDs)

A set of XMC peripheral drivers are available in v4. The libraries of the XMC Lib are peripheral IP oriented, not specific to individual devices. Device specific deviation of an IP is handled by conditional compilation.

Three possible use models exist for these peripheral drivers:

- Used by DAVE APPs to improve robustness and transparency of code generation.
- Standalone implementation (i.e. independent of DAVE APPs), to simplify the use of the XMC microcontrollers in case DAVE APPs are not sufficient.
- Mixed use of DAVE APPs and LLD.
  - For example, if there is no APP for the required application use-case, the LLD can be used. With the APIs of the LLDs the full flexibility of peripherals and connectivity can be used instead of accessing the registers directly.

4.1 Using the LLDs with DAVE APPs

It is possible to extend the project functionality by adding in the LLDs when there is no exact fit of the APP application use-case. This is done by including the header file of the required peripheral.

The following steps are required to use the peripheral:

1. Peripheral configuration and initialization.
2. Connectivity configuration.
3. Event/interrupt configuration.
4. Start operation.
5. IO configuration.
6. Manage peripheral.

The XMC Lib package can be downloaded from **www.infineon.com/DAVE**.
The help documentation for XMC Lib is included in the XMC Lib package.

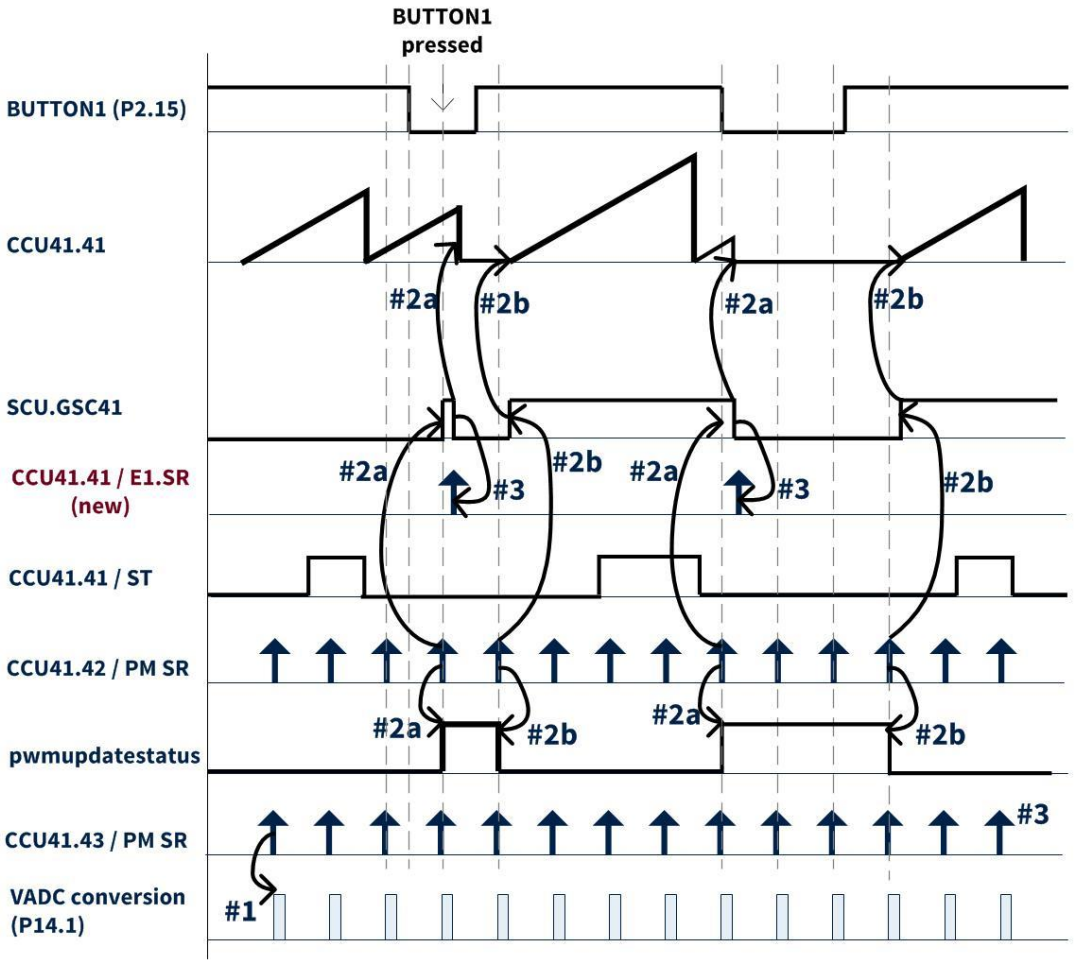
4.2 Extending the Project functionality with LLD

So far our example updates the PWM frequency based on the scaled ADC value read from the potentiometer whenever the button is pressed. The frequency is only read on each period match event.

With the LLD, the CCU4 timer is stopped when the button is pressed (falling edge / event 1) and started again when the button is released (rising edge / event 0). At event 1, which the external stop event is tied to, the CCU4 timer period is updated.

The functions are:

- Event 0 (External start event); triggered on falling edge of SCU.GSC40 signal.
- Event 1 (External stop event); triggered on rising edge of SCU.GSC40 signal.
- Interrupt event on external stop; to update the PWM frequency.
- LED does not toggle (i.e. PWM timer is stopped) as long as button is pressed.



#1 – CCU41.43 SR (Period Match Event) trigger a VADC conversion every 1ms
➔ Read the ADC result and scale it by 128.

#2 – CCU41.42 SR (Period Match Event) ISR; triggered every 1ms.
Checks if (pwmupdatestatus=0 and button is pressed)
#2a ➔Set pwmupdatestatus=1 and create a low to high transition on SCU.GSC41
Check if (pwmupdatestatus=1 and button is not pressed)
#2b ➔Set pwmupdatestatus=0 and create a high to low transition on SCU.GSC41

#3 – CCU41.41 SR (Event1: External Stop Event) triggers an period update
➔ updates the newfrequency

Note:
- In main loop, events are configured and CCU41.41 PWM is started using software.
- CCU41.41 SR (Period Match Event) ISR is removed. Frequency update is completed by Event 1(External Stop) event.

Figure 46 Extending the use-case: Event timing diagram

4.2.1 Adding the XMC Header file

We need to add the header file of the required peripheral (xmc_ccu4.h). As the GLOBAL_CCU4 APP in our example project uses “xmc_ccu4.h”, we do not need to include this manually.

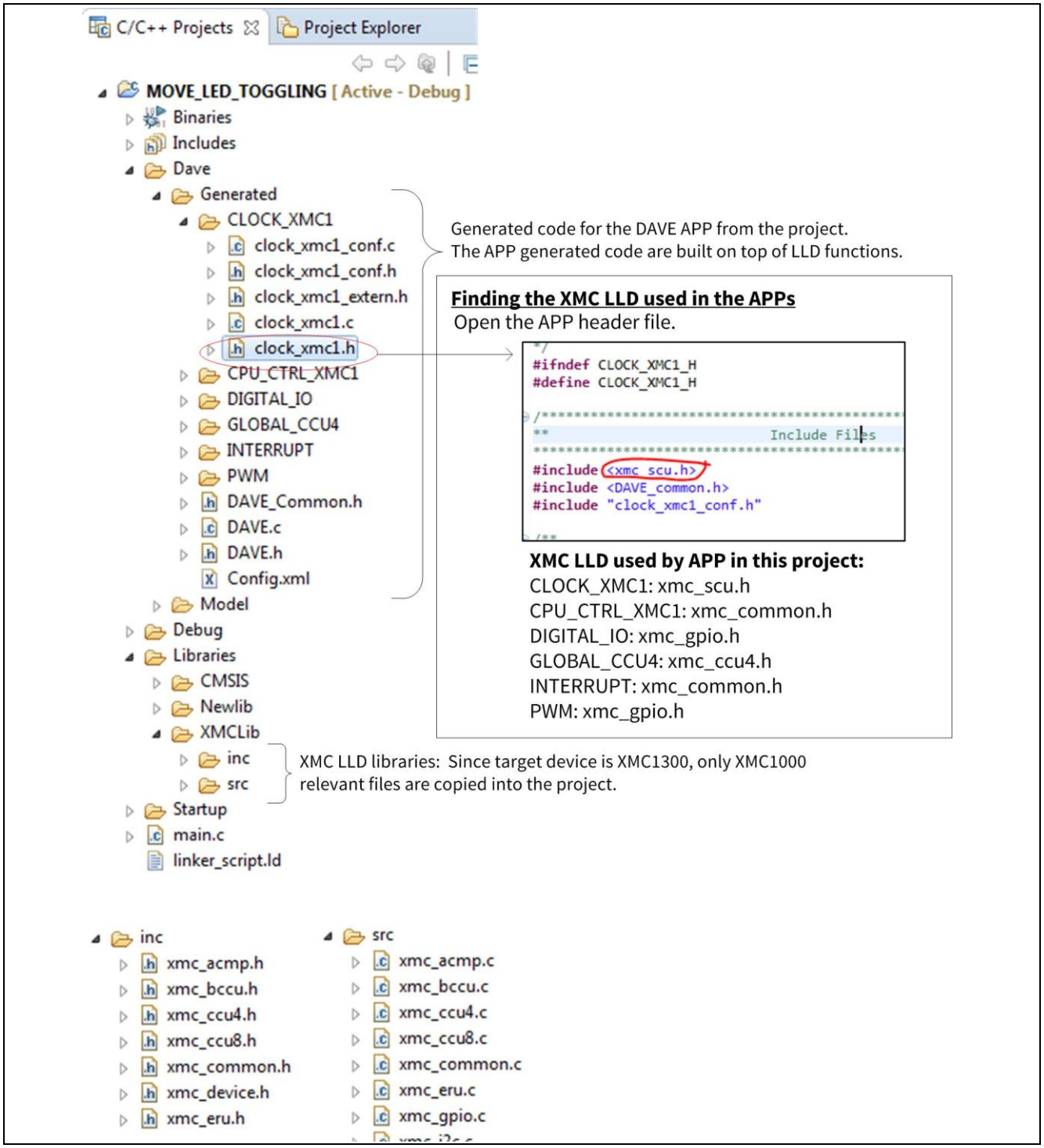


Figure 47 Finding the XMC LLD used by APP in the project

4.2.2 Updating the APP settings and related code

The application now updates the frequency of toggling on each detected external stop event. Therefore the APP settings and code supporting the period match event and interrupt settings are removed.

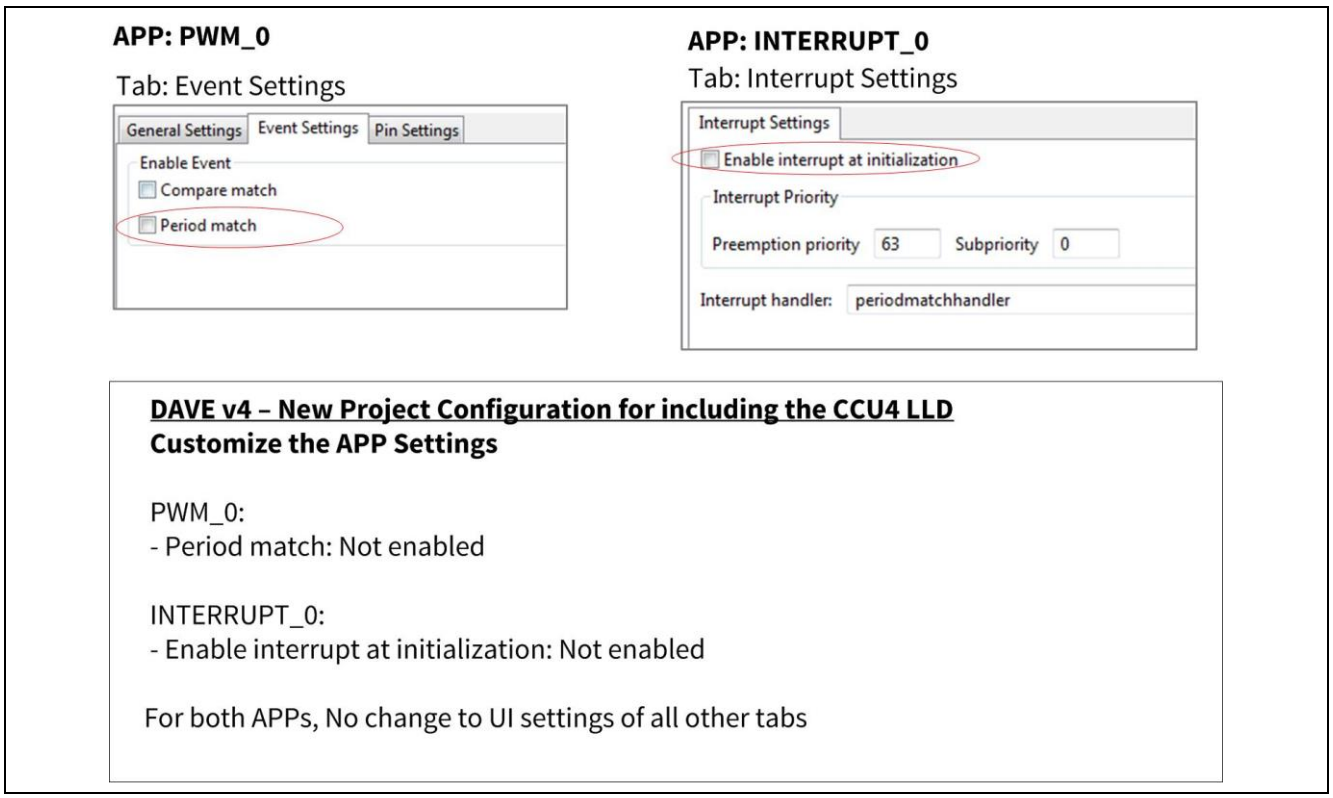


Figure 48 APP settings update

```
/**
* @brief periodmatchhandler() - PWM_0 ISR handler [commented out]
*
* <b>Details of function</b><br>
* This routine performs a pwm frequency update when pwmupdatestatus is set
*/
//void periodmatchhandler(void)
//{
//              if(pwmupdatestatus)
//              {
//                            PWM_AcknowledgeInterrupt(&PWM_0, PWM_PERIODMATCH_INTERRUPT);
//                            PWM_SetFreq(&PWM_0, newfrequency);
//
//                            pwmupdatestatus=0;      //clear the status for PWM update
//              }
//}
```

Figure 49 Remove code for periodmatchhandler(void)

4.2.3 Adding the XMC CCU4 Code

First, define the structure for the external start and stop event configuration. This determines the input and edge that the configured events are being detected. For Event 0, an event is generated on a rising edge on SCU.GSC41 signal. On Event 1, an event is generated on a falling edge of SCU.GSC41 signal.

```
#include <XMC4500.h>
#include <DAVE.h>          //Declarations from DAVE Code Generation (includes SFR declaration)

volatile XMC_VADC_RESULT_SIZE_t result;
volatile uint32_t newfrequency=1;
bool pwmupdatestatus=0;

/* Added XMC CCU4 LLD: External Start Event Configuration*/
XMC_CCU4_SLICE_EVENT_CONFIG_t event0_config = {
 .mAPPed_input = XMC_CCU4_SLICE_INPUT_I,  /* input signal - SCU.GSC41 */
 .edge = XMC_CCU4_SLICE_EVENT_EDGE_SENSITIVITY_RISING_EDGE, /* event edge - rising edge */
 .level = XMC_CCU4_SLICE_EVENT_LEVEL_SENSITIVITY_ACTIVE_HIGH, /* event level - high */
 .duration = XMC_CCU4_SLICE_EVENT_FILTER_7_CYCLES,  /* Low Pass filter duration - 7 clock cycles */
};

/* Added XMC CCU4 LLD: External Stop Event Configuration*/
XMC_CCU4_SLICE_EVENT_CONFIG_t event1_config = {
 .mAPPed_input = XMC_CCU4_SLICE_INPUT_I,  /* input signal - SCU.GSC41 */
 .edge= XMC_CCU4_SLICE_EVENT_EDGE_SENSITIVITY_FALLING_EDGE,  /* event edge - falling edge */
 .level = XMC_CCU4_SLICE_EVENT_LEVEL_SENSITIVITY_ACTIVE_HIGH,  /* event level - high */
 .duration = XMC_CCU4_SLICE_EVENT_FILTER_7_CYCLES,  /* Low Pass filter duration - 7 clock cycles */
};
```

Figure 50 Update configuration with XMC CCU4 LLD: Initialization with CCU4 event configuration

The button is checked periodically if it is pressed. In this handler the specific signal transitions are created on SCU.GSC41 signal.

- If the button is pressed, a high to low transition is generated on SCU.GSC41 for an external stop event.
- If the button is released, a low to high transition is generated for an external start event.

```
/**
 * @brief timerhandler() - TIMER_1 ISR Handler
 * <b>Details of function</b><br>
 * This routine creates low to high and high to low transition on the SCU.GSC41signals  */

void timerhandler(void)
{               /* start the timer only when timer is not pressed */
                if((pwmupdatestatus==1) && (DIGITAL_IO_GetInput(&BUTTON1) == 1))
                {
                  /* Create a rising edge on SCU.GSC41 to trigger an external start event*/
                  XMC_SCU_SetCcuTriggerLow(XMC_SCU_CCU_TRIGGER_CCU41);
                  XMC_SCU_SetCcuTriggerHigh(XMC_SCU_CCU_TRIGGER_CCU41);
                  pwmupdatestatus=0;    /* clear the status for PWM update */
                }
                /* update speed of toggling only when button is pressed*/
```

```
                if((pwmupdatestatus==0) && (DIGITAL_IO_GetInput(&BUTTON1) == 0))
                {
                  /* Create a falling edge on SCU.GSC41 to trigger an external stop event*/
                  XMC_SCU_SetCcuTriggerHigh(XMC_SCU_CCU_TRIGGER_CCU41);
                  XMC_SCU_SetCcuTriggerLow(XMC_SCU_CCU_TRIGGER_CCU41);
                  pwmupdatestatus=1; /* clear the status for PWM update */
                }
}
```

Figure 51 Add XMC SCU LLD code for interrupt handler: timerhandler(void)

Most external events for start and stop timer are configured. For the stop event at Event 1, the interrupt is enabled.

```
/**
* @brief main() - Application entry point
 * <b>Details of function</b><br>
* This routine is the application entry point. It configures the CCU4 events and starts the PWM connected slice
*/
int main(void)
{
 DAVE_STATUS_t status;
 status = DAVE_Init();  /* Initialization of DAVE APPs  */

 if(status == DAVE_STATUS_SUCCESS)
{    XMC_DEBUG("DAVE APPs initialization success\n");
 }
 else
 {
  /* Placeholder for error handler code. The while loop below can be replaced with an user error handler */
  XMC_DEBUG(("DAVE APPs initialization failed with status %d\n", status));
  while(1U)
  {
  }
 }
 /* Configure external start event using XMC_CCU4 LLD*/
 XMC_CCU4_SLICE_ConfigureEvent(PWM_0.config_ptr->ccu4_slice_ptr, XMC_CCU4_SLICE_EVENT_0, &event0_config);
 XMC_CCU4_SLICE_StartConfig(PWM_0.config_ptr->ccu4_slice_ptr, XMC_CCU4_SLICE_EVENT_0, XMC_CCU4_SLICE_START_MODE_TIMER_START);

 /* Configure external stop event using XMC_CCU4 LLD*/
 XMC_CCU4_SLICE_ConfigureEvent(PWM_0.config_ptr->ccu4_slice_ptr, XMC_CCU4_SLICE_EVENT_1, &event1_config);
XMC_CCU4_SLICE_StopConfig(PWM_0.config_ptr->ccu4_slice_ptr, XMC_CCU4_SLICE_EVENT_1, XMC_CCU4_SLICE_END_MODE_TIMER_STOP_CLEAR);
 XMC_CCU4_SLICE_EnableEvent(PWM_0.config_ptr->ccu4_slice_ptr, XMC_CCU4_SLICE_IRQ_ID_EVENT1);
 XMC_CCU4_SLICE_SetInterruptNode(PWM_0.config_ptr->ccu4_slice_ptr, XMC_CCU4_SLICE_IRQ_ID_EVENT1, XMC_CCU4_SLICE_SR_ID_1);
 NVIC_EnableIRQ(CCU41_1_IRQn);

 PWM_Start(&PWM_0); /*Start PWM Slice*/

 /* Placeholder for user application code. The while loop below can be replaced with user application code. */
 while(1U)
 {
 }
 return 1;
```

```
}
```

Figure 52 Add XMC CCU4 LLD code for event configurations: int main(void)

Since the periodmatchhandler was commented out, the update of the frequency is done on detection of each Event 1 detected for external stop event. In this interrupt handler, the event is cleared and the requested frequency is updated.

```
void CCU41_1_IRQHandler(void)
{
        /* Clears event 1 flag and set the new PWM frequency*/
        XMC_CCU4_SLICE_ClearEvent(PWM_0.config_ptr->ccu4_slice_ptr,XMC_CCU4_SLICE_IRQ_ID_EVENT1);
        PWM_SetFreq(&PWM_0, newfrequency);
}
```

Figure 53 Configure the code with XMC CCU4 LLD: Add an IRQ handler for Event 1

5 Revision History

Current Version is V1.0, 2015-05

Page or Reference	Description of change
V1.0	
	Initial Version

www.infineon.com

Edition 2015-05

Published by

Infineon Technologies AG

81726 Munich, Germany

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

Email: erratum@infineon.com

Document reference

AP32295

Legal Disclaimer

THE INFORMATION GIVEN IN THIS APPLICATION NOTE (INCLUDING BUT NOT LIMITED TO CONTENTS OF REFERENCED WEBSITES) IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED 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.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office. Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.