

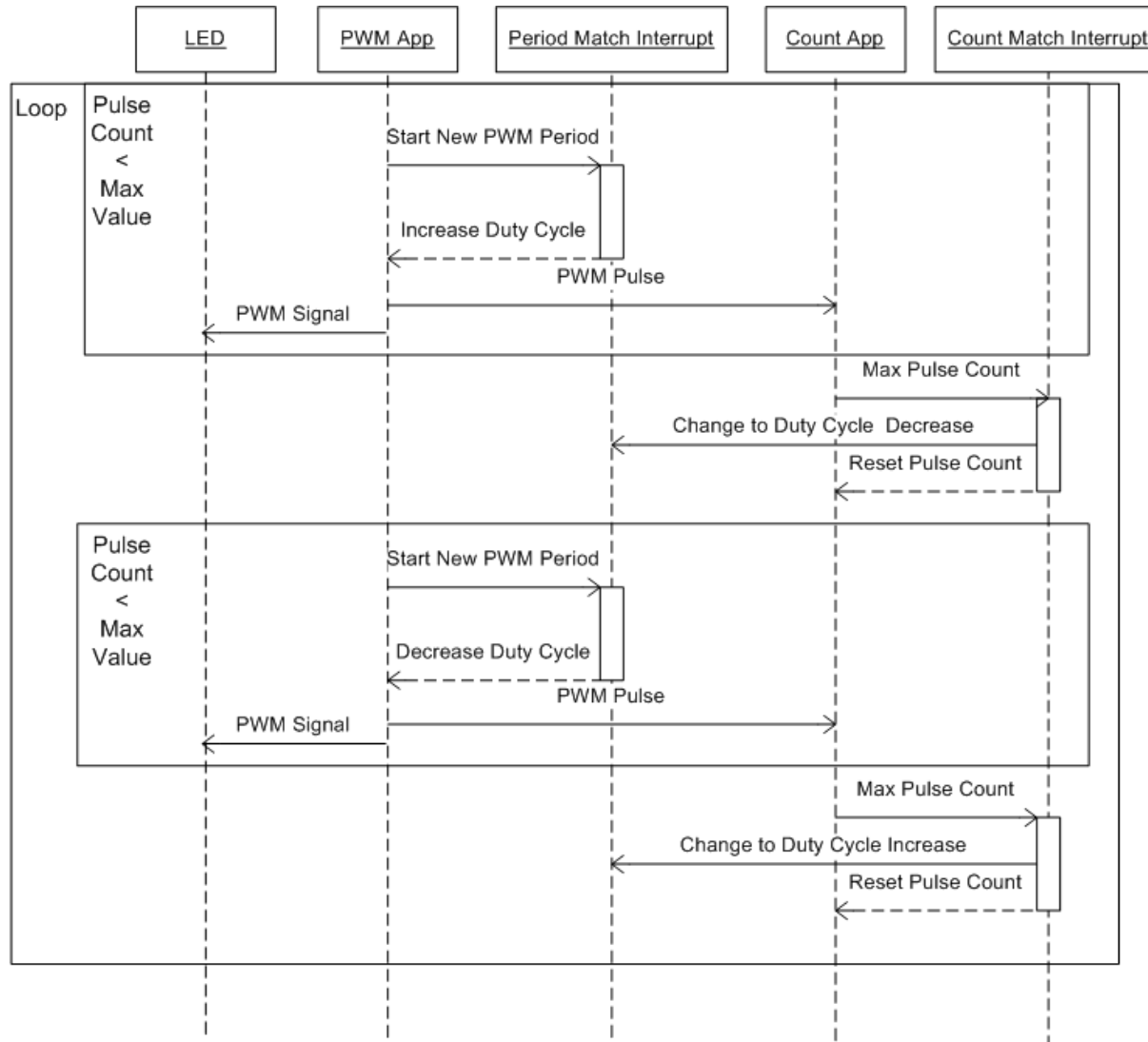
Quick Start Tutorial

Presentation Tutorial for a Quick Start Hands-on Session: Creating a simple Project using PWM and Count Apps.

Version 2.1, June, 2014



Scope of the Project for this Hands-on Tutorial



Sequence Diagram

- Changing the brightness of an LED with the PWM App PWMSP001
 - The interrupt on timer period match changes the brightness of the LED by changing the duty cycle of the PWM
- Counting of the PWM pulses with the count App CNT001
 - The interrupt on the count match toggles between increasing and decreasing of brightness

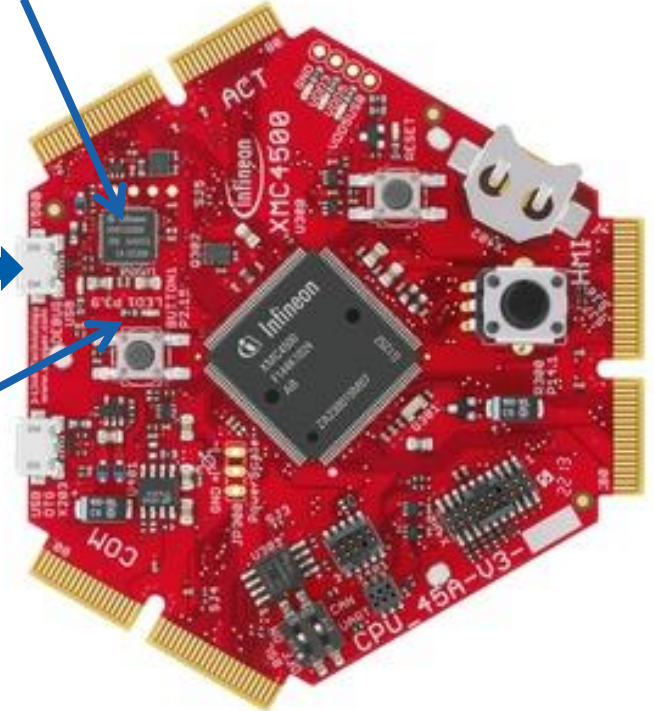
Used HW Setup for this Hands-on Tutorial

USB connection from the PC, the board is also powered via USB.

J-Link on board debugger (OBD).

XMC4500 CPU board.

The LED of the CPU board is used to demo the functionality of the PWM and Count App.

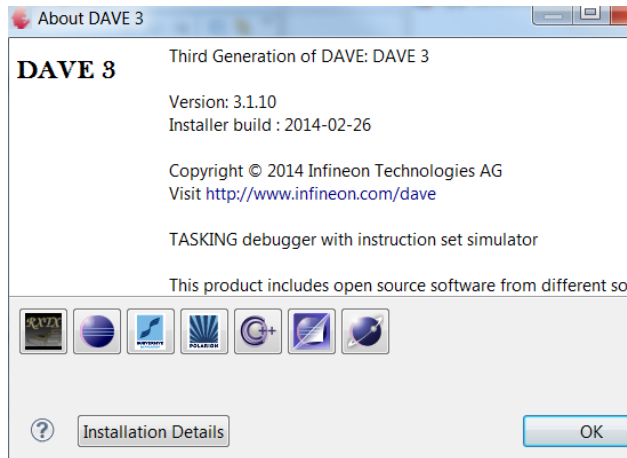


It is also possible to use a different XMC kit (e.g. Relax kit or XMC4400/XMC4200 kits or one of the XMC1000 kits). In this case a different target MCU (page 6) has to be selected and the LED might be on a different port which has then be considered in the manual pin assignment (page 29).

More details about tools and kits and how to buy can be found at:
www.infineon.com/xmc-dev

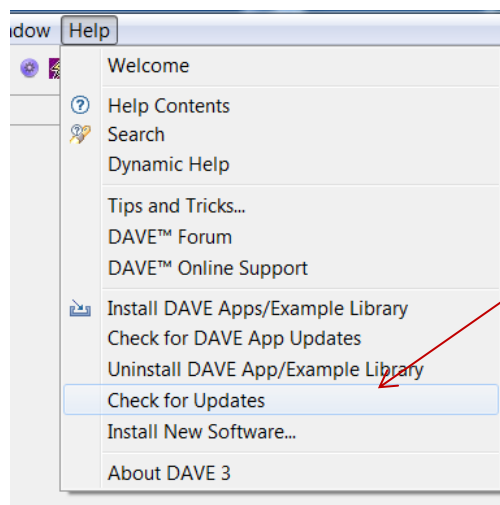
DAVE™ Development Platform

Before starting this tutorial please **download** and install DAVE. The download package contains detailed installation instructions. Video tutorials about installations can be found **here**.



- Please be sure that you always use the latest version of DAVE
This tutorial was created with DAVE v3.1.10

- Version information can be found in:
-> Help -> About DAVE 3



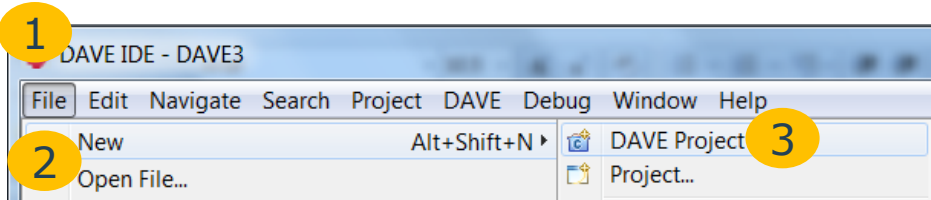
- We recommend to regularly check for plug-in updates and install new updates:
press -> Help -> Check for Updates

Steps to Accomplish this Tutorial

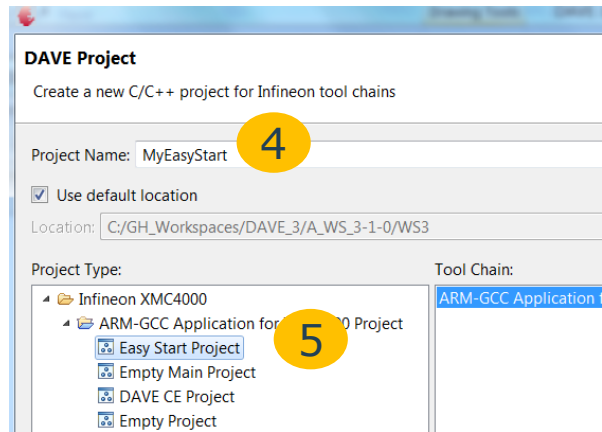
- Test of your DAVE installation with an EasyStart project
- Download DAVE Apps from the web
- Create a DAVE CE project
- Select required DAVE Apps
- Configuration of DAVE Apps
- Signal Connection
- Manual Pin Assignment
- Resource Solving
- Code Generation
- Coding

Create an EasyStart Project to check the HW Setup

Create an EasyStart Project and check that the HW and the Debugger setup is OK
The EasyStart project is a blinky project that can be used to get easily a first program running.



Press: >File >New >DAVE Project

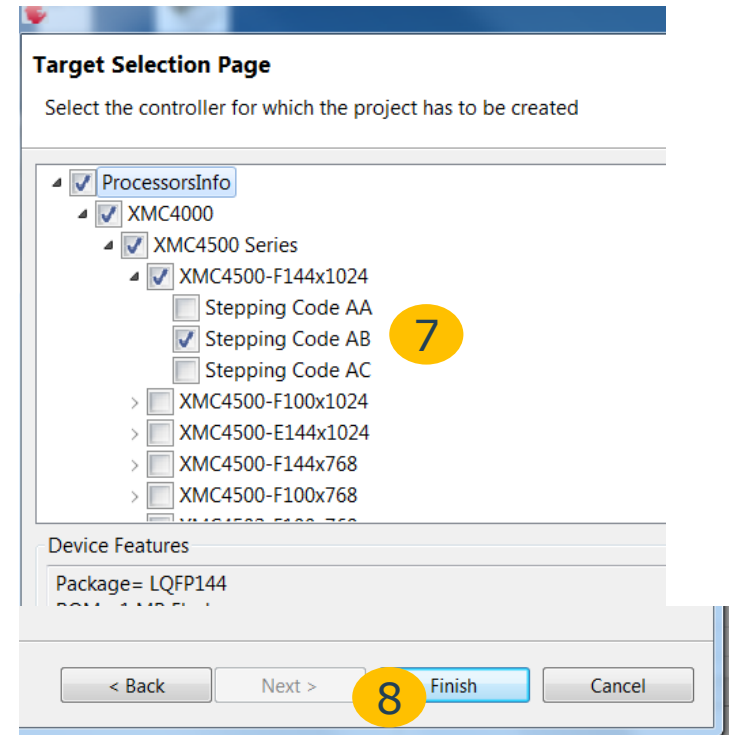


Enter project name:
MyEasyStart

Select Project Type:
Easy Start Project



Press: Next



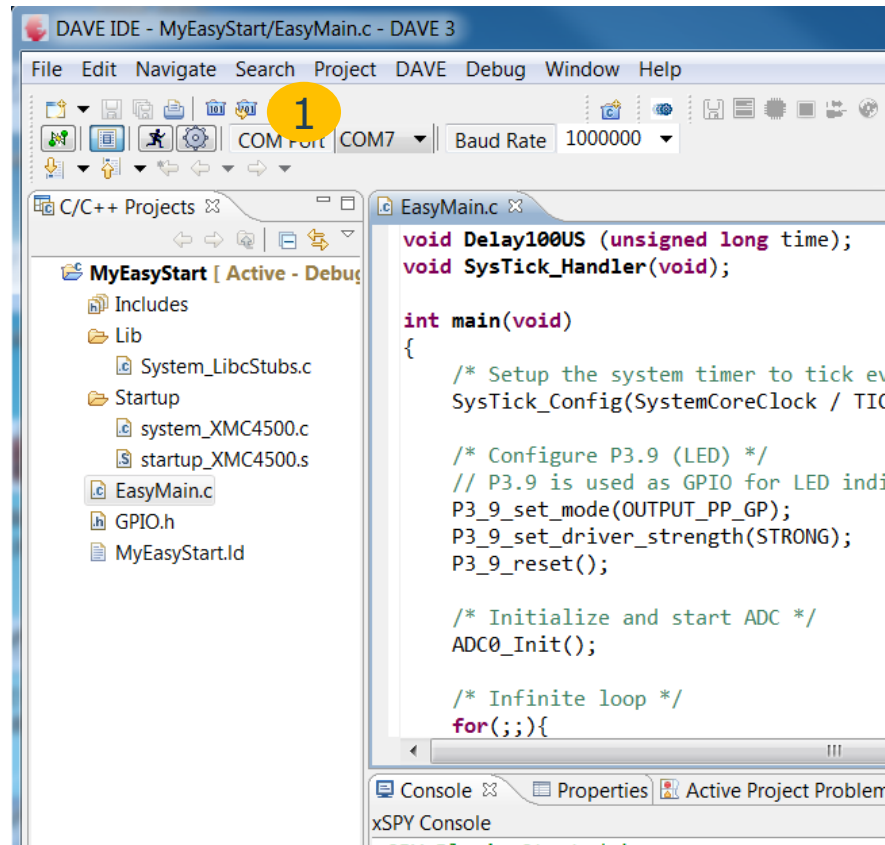
Select the target MCU
Press: Finish

The EasyStart Project is now added in the Workspace

The project contains the :

- Startup files
- NewLib stubs
- EasyMain.c and GPIO.h
- Linker script file

The new created project is the active project, most actions are applied to the active projects (build, debug, properties,...). The active project can be changed with right mouse click on the project.



The EasyMain.c can be regarded as template for the first simple programming trials with the XMC4500 without using DAVE™ Apps.

Default functionality: LED toggling on Port 3.9, toggle frequency based on the analog value on port 1.14.

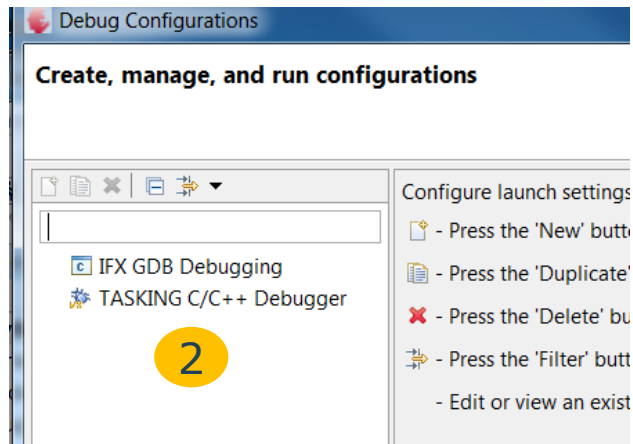
- 1 By pressing the "Build Active Project" or "Rebuild Active Project" Button the project can be build.

Starting / Launching the Debugger

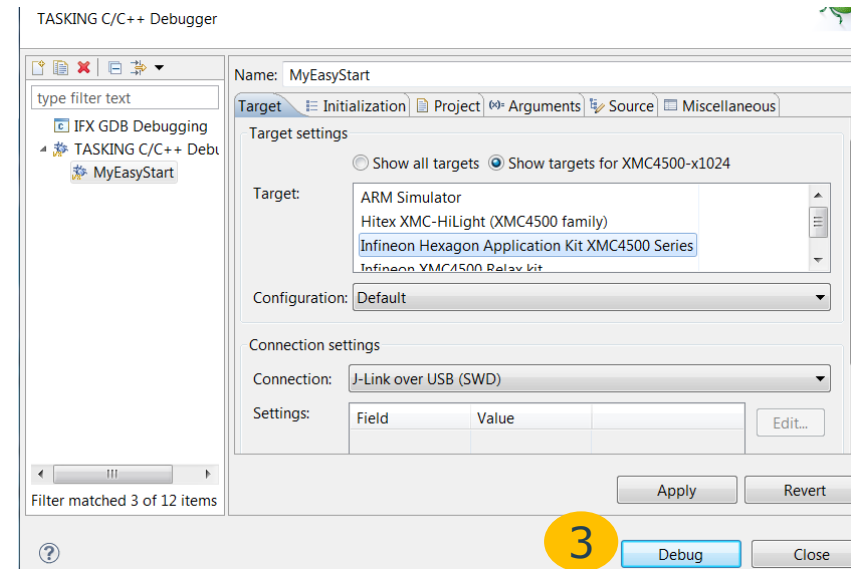
After successfully building of the project the debugger can be started (ensure that the board is connected and correctly and the J-Link debugger device is ready to use).



1 Click on the Debug button to start the debugger



2 The first time when the debugger is launched for a project the debug configuration view show up: Double click on TASKING C/C++ Debugger creates a new debug configuration. Alternatively a double click on IFX GDB Debugging creates a debug configuration for the GDB.



The default settings support this HW setup, in case a different HW setup is used it might be necessary to adjusted the settings.

3 Press Debug to start the debugger.

When pressing the Debug button the second time the debugger starts automatically with the last used debug configuration.

How to operate the TASKING debugger

After the debugger has loaded the image into the Flash the Eclipse perspective is changed to the TASKING Debugger perspective.

- 1 When pressing Resume, the program starts and the LED on the Hexagon CPU board should blink.

Debug view for program control:

- Resume (start)
- Single stepping
- Suspend
- Reset
- Terminate
- etc

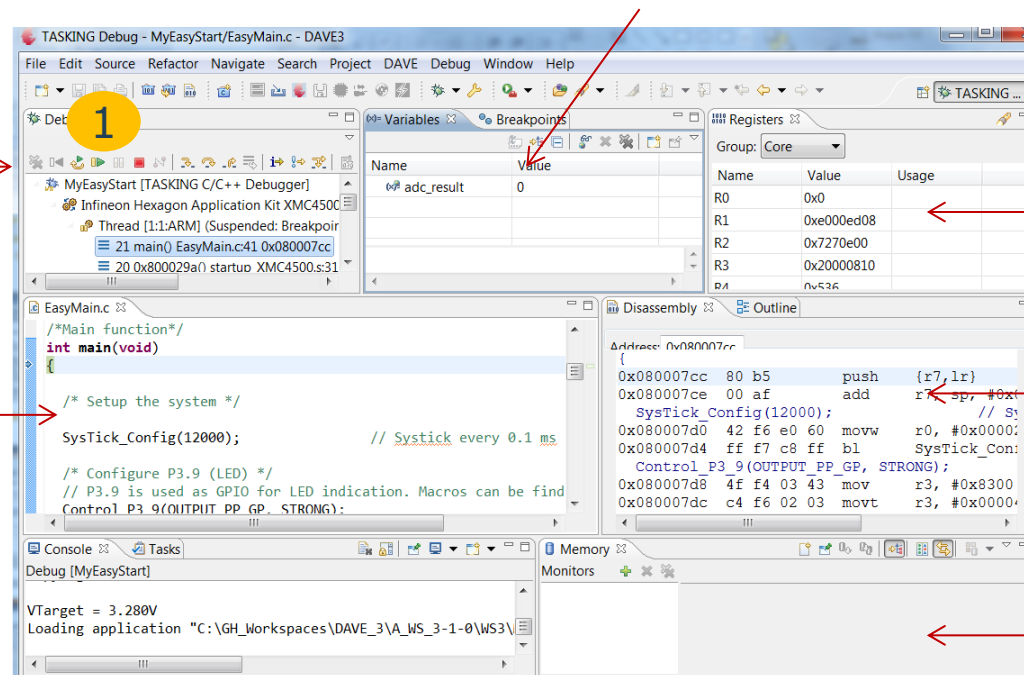
Editor window with source code

Variable View
Breakpoint view

Register view of the SFRs for the XMC

Disassembly view

Memory view

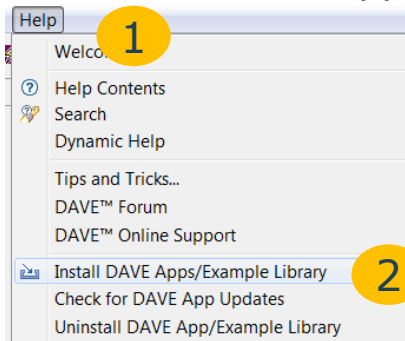


Views can be freely and flexible arranged, new views can be added: >Window >Show View

When using the GDB then the default CDT debug perspective will be opened that provides a similar functionality as the TASKING debugger.

Download of DAVE™ Apps from the Web

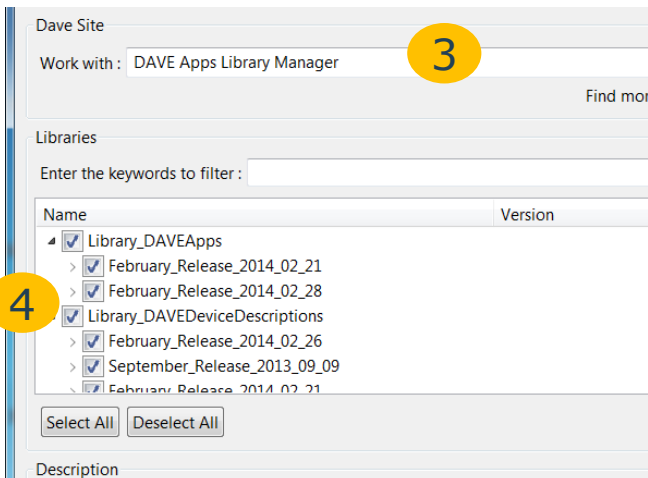
When using the DAVE installer then a complete set of DAVE Apps libraries has been already installed. In spite we recommend to do the following steps to ensure that the most current versions of DAVE Apps libraries are installed.



1 2 Press: >Help >Install Apps/Example Library.

3 Select: DAVE Apps Library Manager.

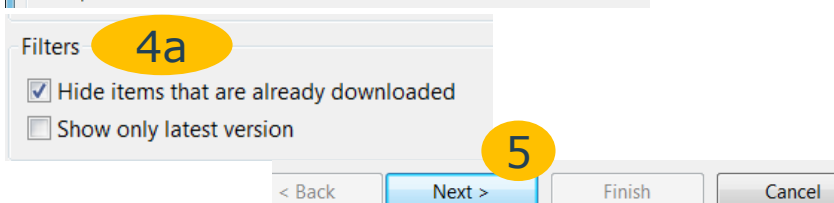
In case this option is not visible click on Library Update Site and activate both links
In case of connection failures see next page.



4 Select: Library_DAVEApps and Library_DAVEDeviceDescription (Release is continuously updated)

4a Filters may be checked.

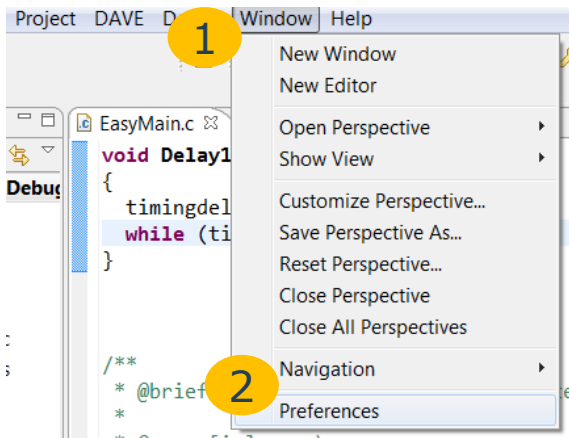
5 Press: Next and follow further instructions.



Now the DAVE Apps and Device Descriptions are downloaded to your local library store located at: ..\<user>\Infineon\D3LibraryStore

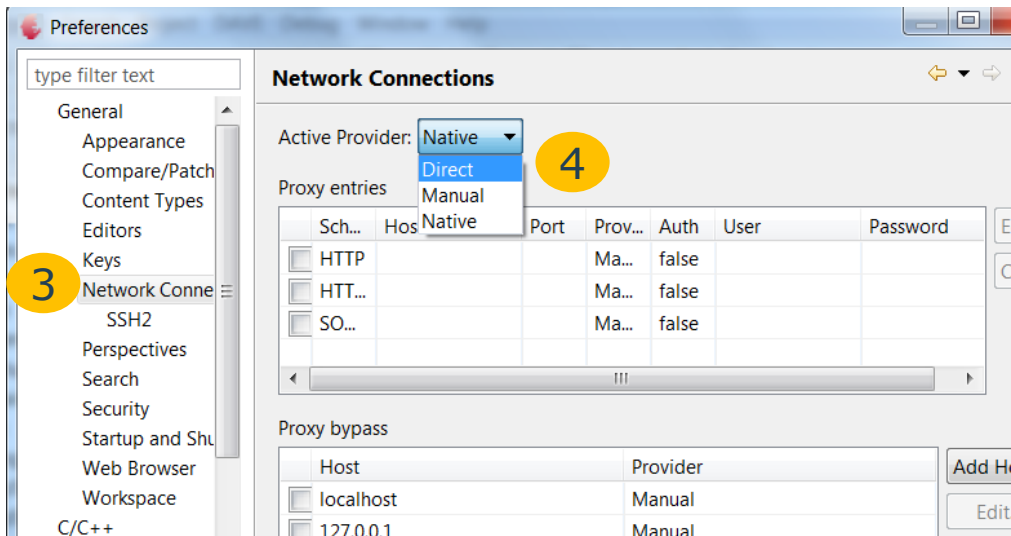
Alternative Network Connections Settings / Manual Download of DAVE™ Apps

Depending on security / firewall settings of the LAN, connection to the web may fail.
In this case changing of the Eclipse Network settings may help.



1 2 Press: >Window >Preferences.

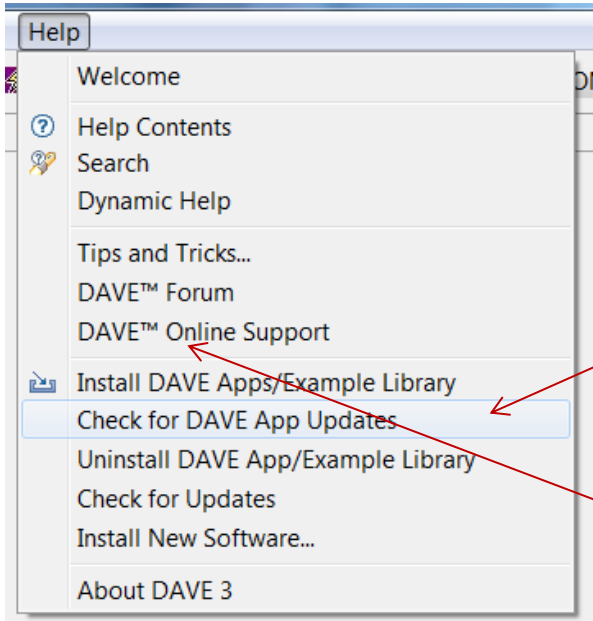
3 Press:
>General >Network
Connections.



4 Select : Direct (when outside of
a secured company LAN)
Or
Select Native (when inside of a
secured company LAN).

In case of other connection
problems please **Download** the
DAVE Apps as zip file directly
from the Infineon web and
follow the instructions described
there.

Updates of the DAVE App Library



- Infineon is continuously releasing new DAVE Apps or updating existing DAVE Apps (new device support or bug fixing).
- The policy for updates of DAVE Apps is to keep full upward compatibility to earlier versions.
- To update to the latest DAVE App library press "Check for DAVE App Updates" in Help menu.
- This will only install updates of already installed DAVE Apps libraries and Device descriptions. To ensure you don't miss new library you may use the install library feature as described two pages before and check the filter (4a).
- Information about the latest DAVE Apps libraries and Device descriptions, release notes, etc can be found on the DAVE™ Online Support portal.
- When updating an existing DAVE App the existing DAVE App version in the local library will be kept and can further on used (see page 16).

Downloading of DAVE™ Example Projects that use DAVE Apps

Press: ->Help ->Install App Library.

1 Select: DAVE Project Library Manager.

2 Click here in case there is not such an entry and activate the links.

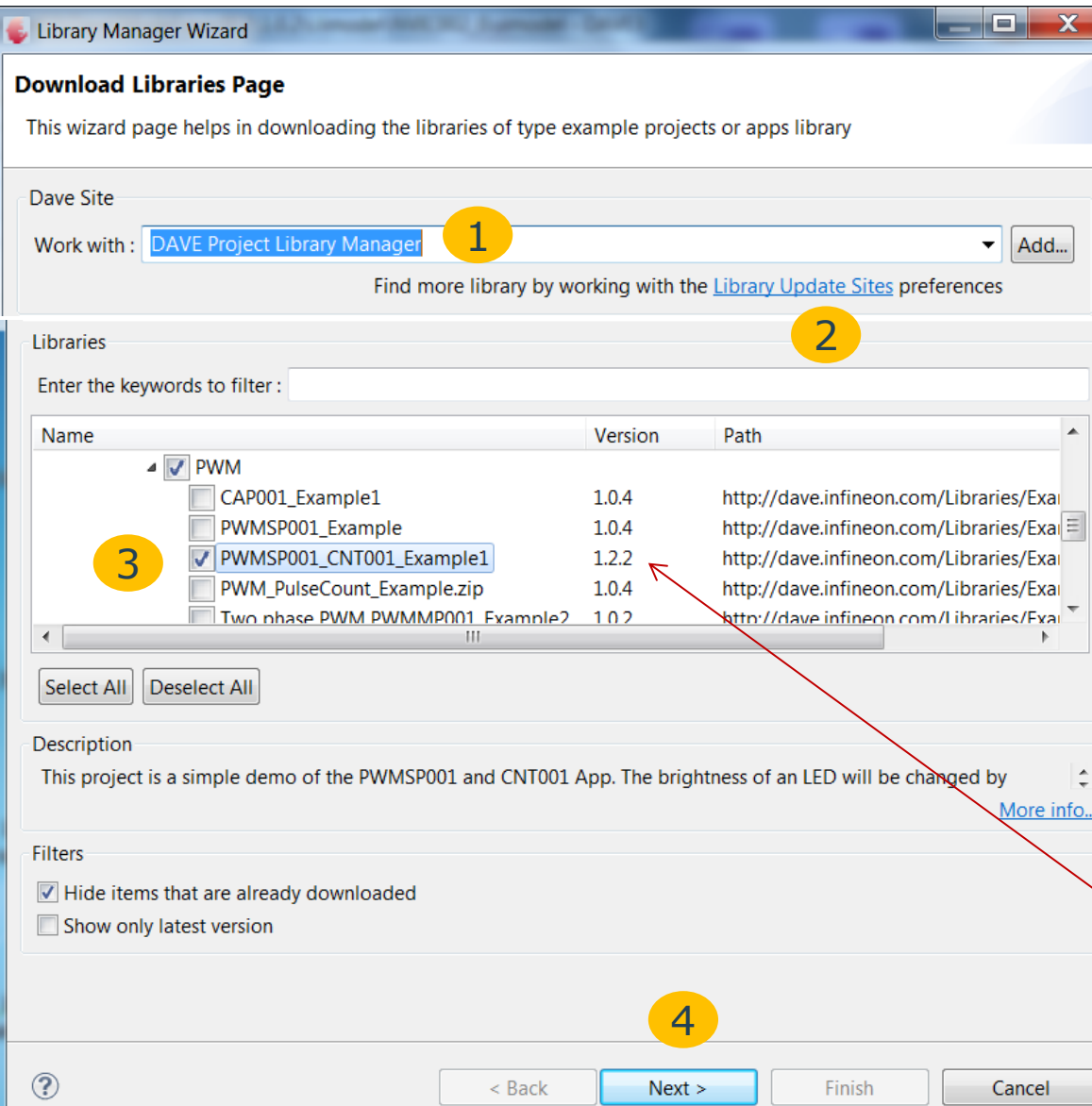
3 Select the Project PWMSP001_CNT001_Example1.

4 Press Next.

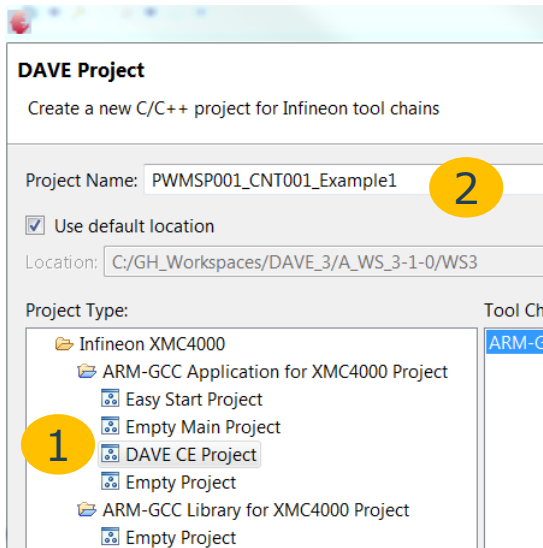
Follow the further instructions.

In case of connection fails or no projects are shown, please refer to page 11 and follow the steps described there.

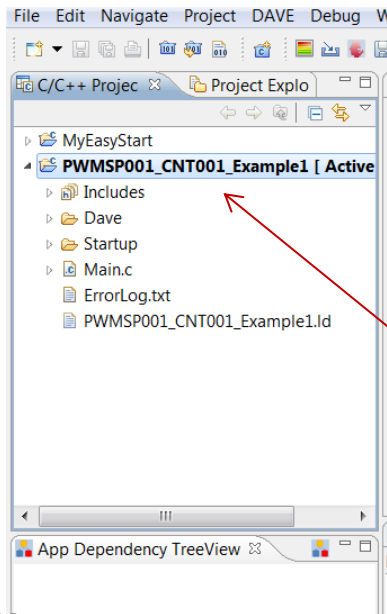
This is the example project of this tutorial. You may download it for your reference. In the meantime a newer version is available.



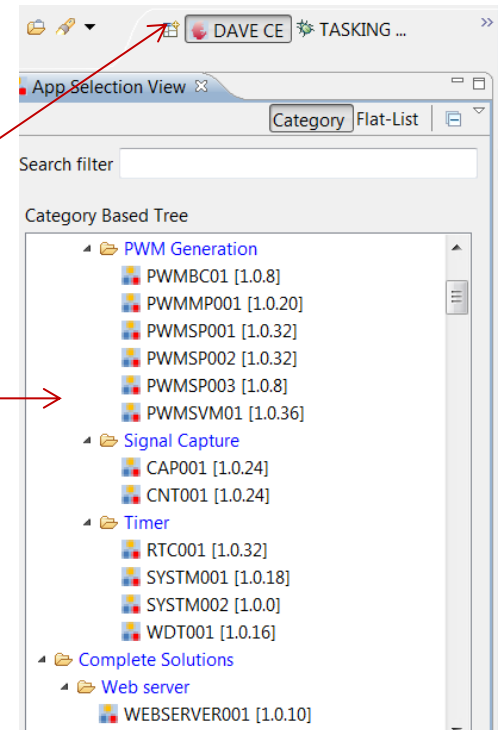
Creating a new Project; To work with DAVE™ Apps a DAVE CE Project Type is required



- 1 Create a new DAVE project as described on page 5 But select the DAVE CE Project type.
- 2 You may choose a different project name than here if you have downloaded the reference project with the same name as described on the previous page.

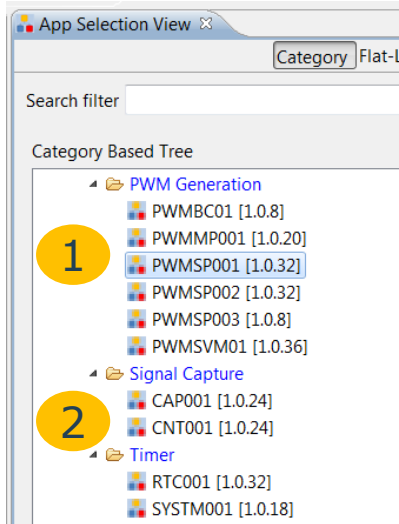


- After the project has been created:
- The Eclipse perspective is changed to the DAVE CE perspective
 - The App Selection view shows the DAVE Apps that have been downloaded before
 - The project is visible in the workspace as Active Project and includes a Main.c



Selecting the Required DAVE Apps

For the project functionality described on page two, we essentially need a PWM App a Count App and Interrupt (NVIC) Apps.



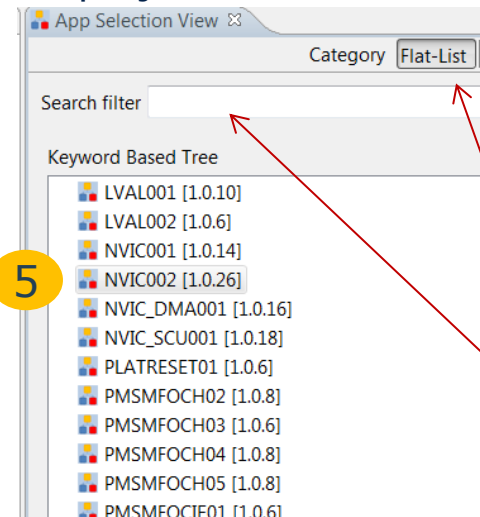
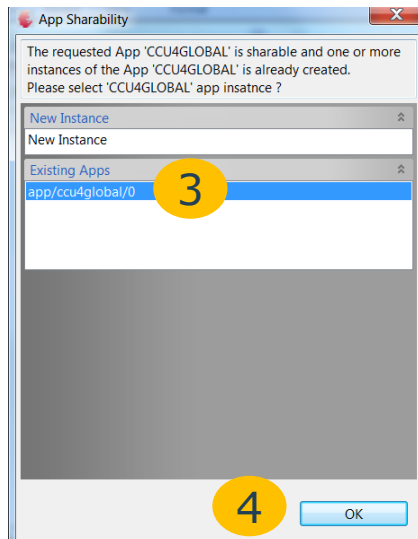
1 Double click: PWMSP001.

2 Double click: CNT001.

3 Select for the requested CCU4GLOBAL: Existing Apps.

4 Press: OK.

5 Double click: NVIC002 and then repeat this double click again, to add two instances of the NVIC002 App into the project.

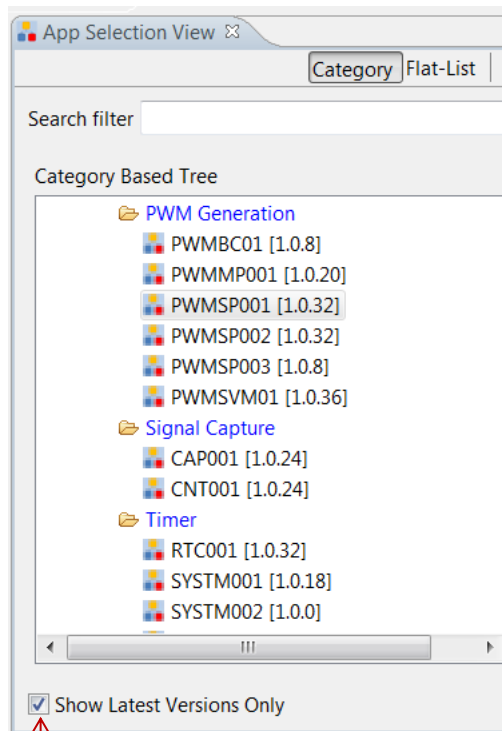


In the App Selection view the required DAVE Apps can be found in a category tree.

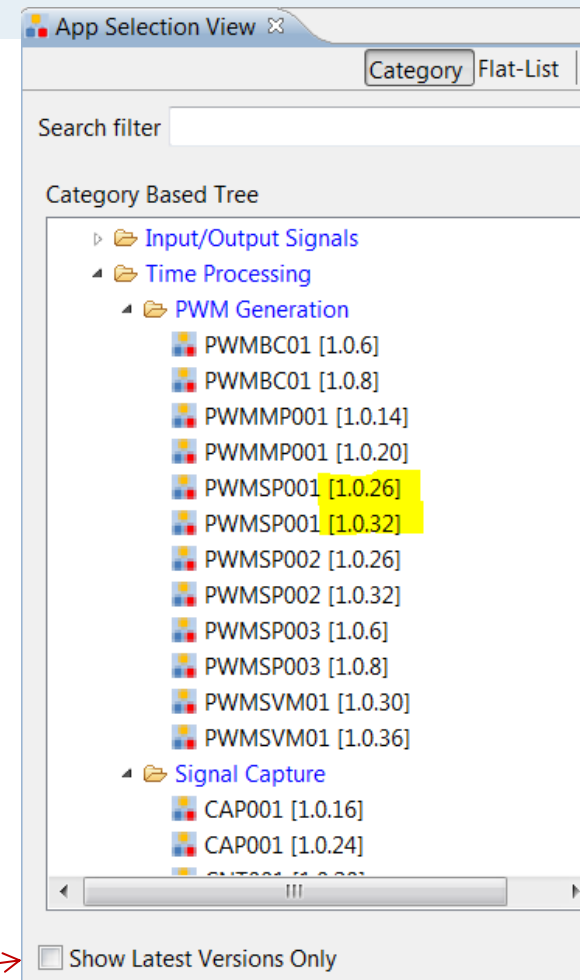
The DAVE Apps can also be listed as alphabetical flat list.

Key words can be used as search filter.

Old Versions of DAVE Apps



When the option “Show Latest Version Only” is checked only the latest installed version of DAVE Apps are shown.



If not checked all installed DAVE Apps versions are shown.

Older versions of DAVE Apps might be used if the existing project uses already an older version. It is not possible to add two different versions of the same App in one project.

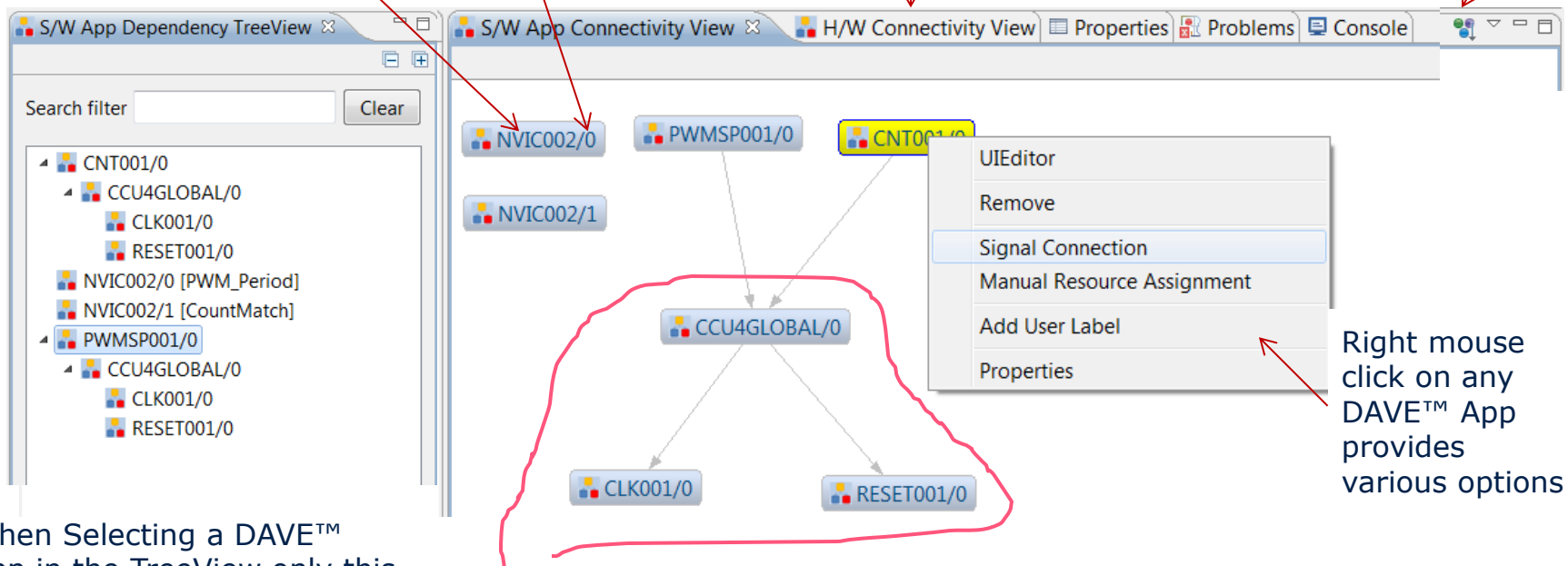
S/W App Connectivity View and App Dependency TreeView

All the DAVE™ Apps we have added to the project (double click) are visible in the S/W App Connectivity View and App Dependency TreeView, in addition all DAVE™ Apps are shown that are required by the DAVE Apps we have added.

This is the DAVE™ App name
This is the instance

We will see later the purpose of the H/W Connectivity View

Click here: then this view will be refreshed



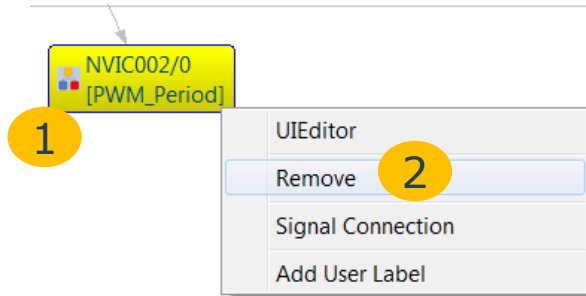
When Selecting a DAVE™ App in the TreeView only this DAVE™ App plus the next level of required DAVE™ Apps will be shown in the S/W Connectivity.

These are the DAVE™ Apps that are automatically added because they are required by the DAVE™ Apps we have added (S/W connected)

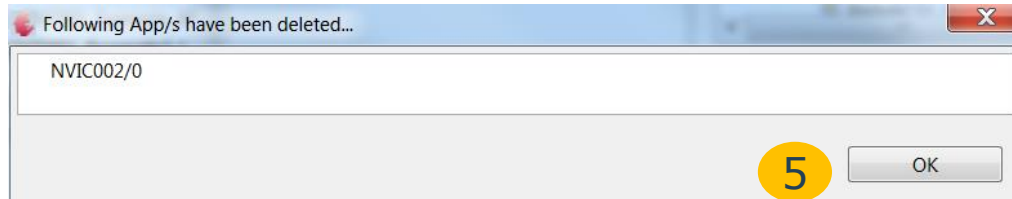
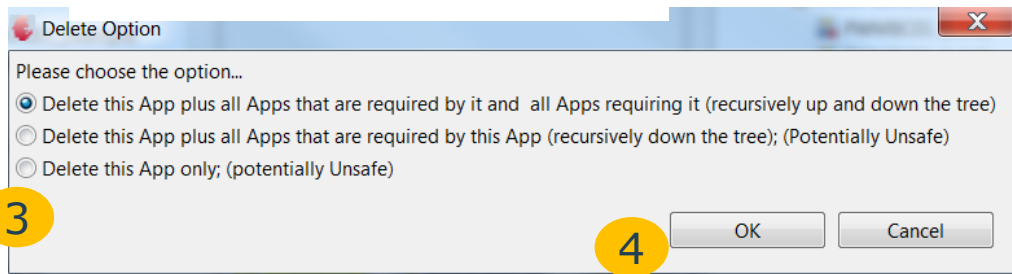
- CLK001: Provides clock information
- CCU4Global: provides some global settings of the CCU4
- RESET001: provides the API to reset peripheral modules

Deleting / Removing of DAVE™ Apps

If a DAVE™ App is not required any more or it has been added to the project erroneously or if it should be replaced, it is possible to remove a DAVE™ App from the project.



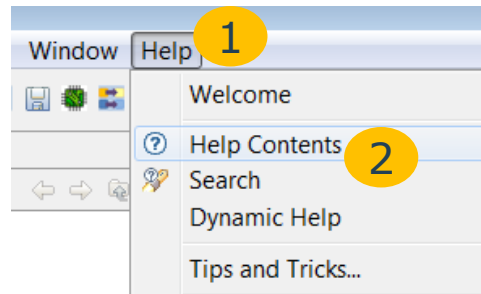
Note, it is not required to exercise the DAVE™ App remove function for successfully accomplishing this tutorial.



- 1 Right mouse click on the DAVE™ App that should be deleted.
- 2 Select: Remove.
- 3 Chose option, the default option is the safest.
- 4 Press: OK, to confirm option.
- 5 Press: OK, to confirm the deletion of the listed DAVE Apps.

In this case no other DAVE™ App will be deleted because the NVIC002/0 App is not S/W dependent on any other DAVE App. If PWMSP001/0 would be removed also only this App will removed because CCU4Global, CLK001 and RESET are still required by the CNT001 App. But if then CNT001/0 APP would be removed, all required DAVE Apps would also be removed. Just give it a try and see what happens.

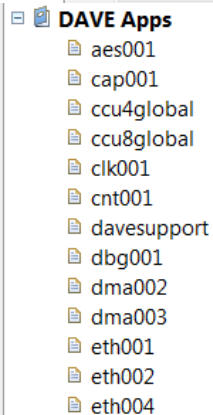
Help Information about DAVE™ Apps



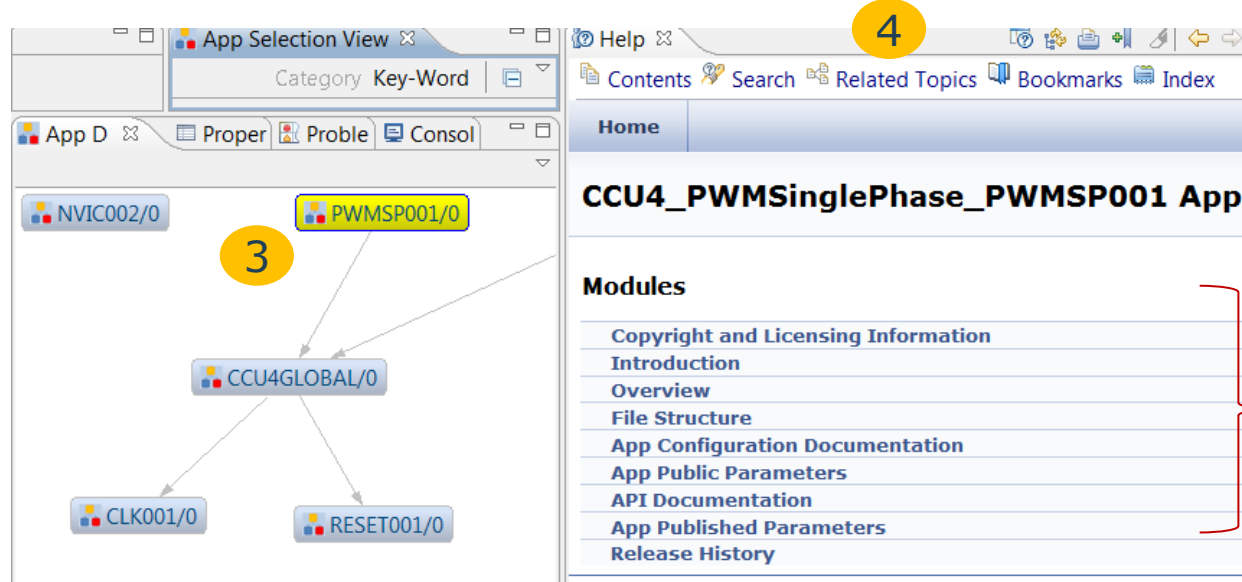
Select: >Help >Help Contents

To open the help content that contains among other topics also a chapter for DAVE™ Apps.

For Each DAVE™ App there is a dedicated documentation, that includes all information to uses the DAVE App properly.



- 3 The DAVE™ App specific document can also be opened in the DAVE CE perspective when pressing F1 and then select the respective DAVE™ App and then press "Related Topics" and "More Info" in the Help view.
- 4



Content structure of each DAVE™ App

Configuration of the PWMSP001 App

The User Interface to configure the PWMSP001 App can be opened by double click on the App or via right mouse option of the App: UIEditor

We will make the following changes to the default settings:

- 1 Change PWM resolution to about 10 μ sec.
- 2 Change PWM frequency to 100 Hz.
- 3 Enable Period Match Interrupt.

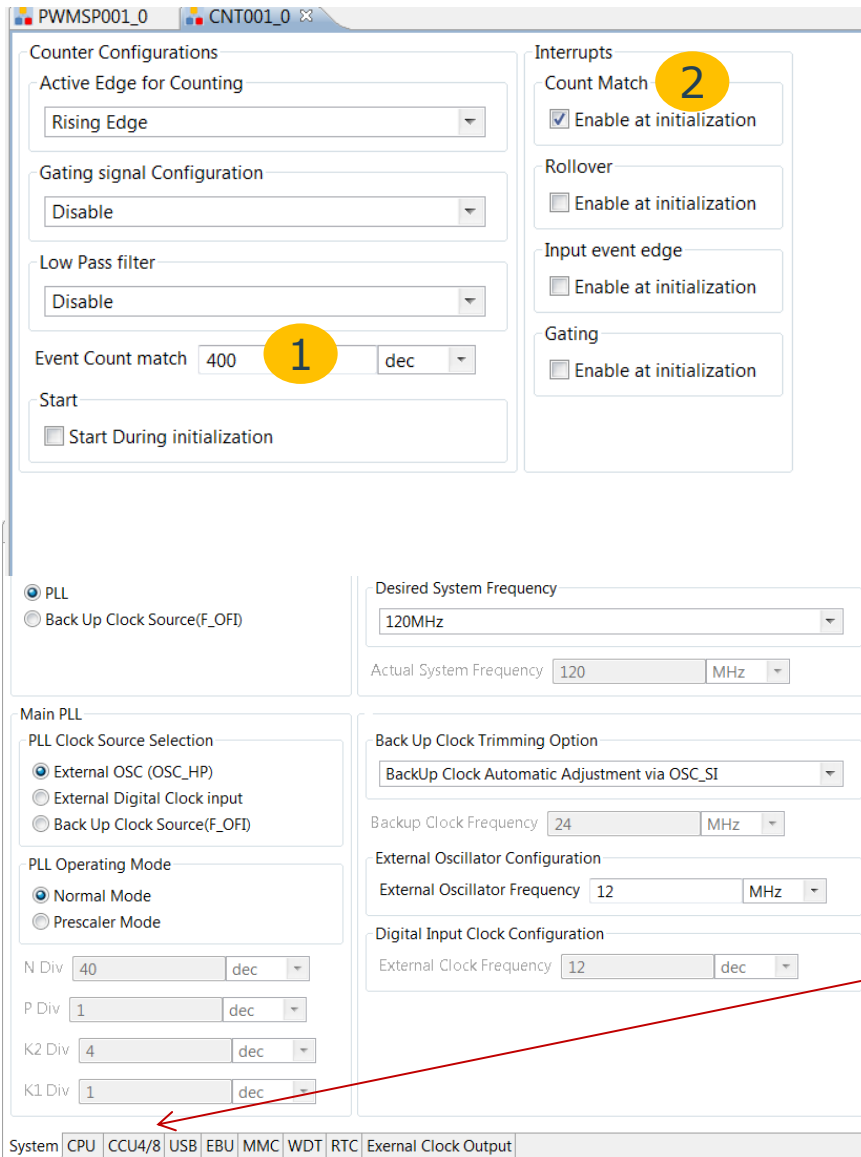
- 4 We could also check "Start during initialization" to start the PWM signal immediately after initialization, but we will do it rather with the API in our program.

Behind these taps there are other initialization options. For our example we can use the default settings, but you might check out which other options are provided by this DAVE App.



e.g. select Passive Level as "Active High" because the LED is connected to 3.3V.

Configuration of the Count App and Clock App



The screenshot shows the configuration interface for the Count App (CNT001_0) and Clock App (CLK001_0). The Count App configuration is divided into two main sections: Counter Configurations and Interrupts. The Counter Configurations section includes settings for Active Edge for Counting (Rising Edge), Gating signal Configuration (Disable), Low Pass filter (Disable), Event Count match (400), and Start (Start During initialization). The Interrupts section includes settings for Count Match (400), Rollover, Input event edge, and Gating. The Clock App configuration is divided into two main sections: Main PLL and Backup Clock Trimming Option. The Main PLL section includes settings for PLL Clock Source Selection (External OSC (OSC_HP)), PLL Operating Mode (Normal Mode), and N Div (40). The Backup Clock Trimming Option section includes settings for Backup Clock Frequency (24 MHz), External Oscillator Configuration (External Oscillator Frequency 12 MHz), and Digital Input Clock Configuration (External Clock Frequency 12 MHz).

Double click on CNT001 App.

- 1 Count match event is set to 400.
- 2 Interrupt event signal for count match is enabled .

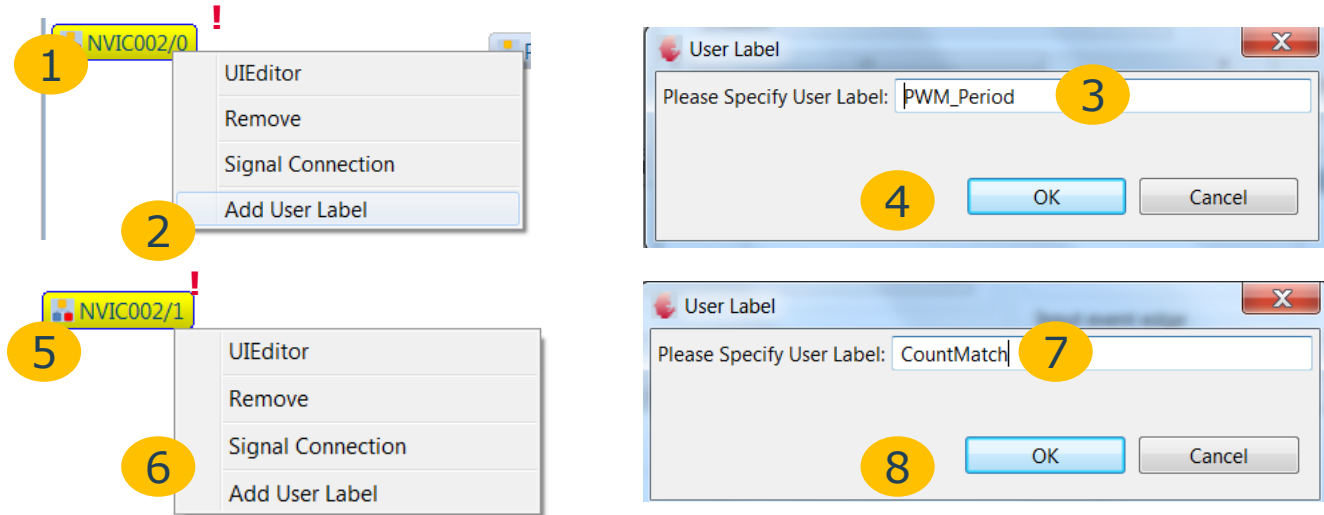
The UI of the CLK001 App allows various clock initializations, we keep the default value:

- External oscillator is 12 MHz (CPU board)
- System clock is 120 MHz
- CCU4 clock is same as system clock

Adding User Labels to the DAVE™ Apps

One of the DAVE™ App right mouse click options is to add a user label to the DAVE App. This feature helps to identify the right DAVE™ App and right instance in case of configuration, signal connection and manual pin assignment.

We will add user labels to the NVIC Apps



1

2

3

4

5

6

7

8

! Watch the instance number.

1

5

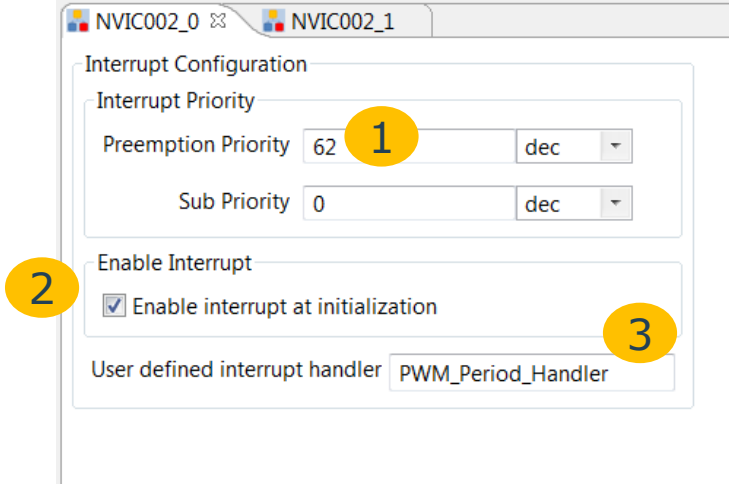
Right mouse click.

The assigned user labels are now visible in the various App views.

The user labels are not used in the user SW.

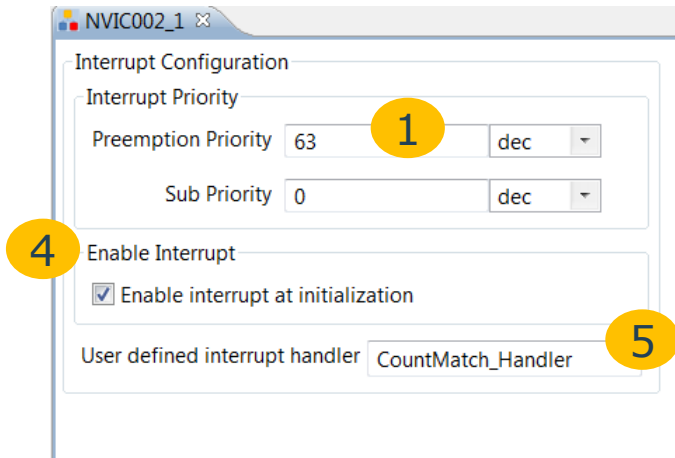
Configuration of the NVIC Apps

Double click on NVIC002, PWM_Period App



1 Set the priority of the PWM period match to a higher priority (62).
(lower number = higher priority)

Double click on NVIC002, CountMatch App

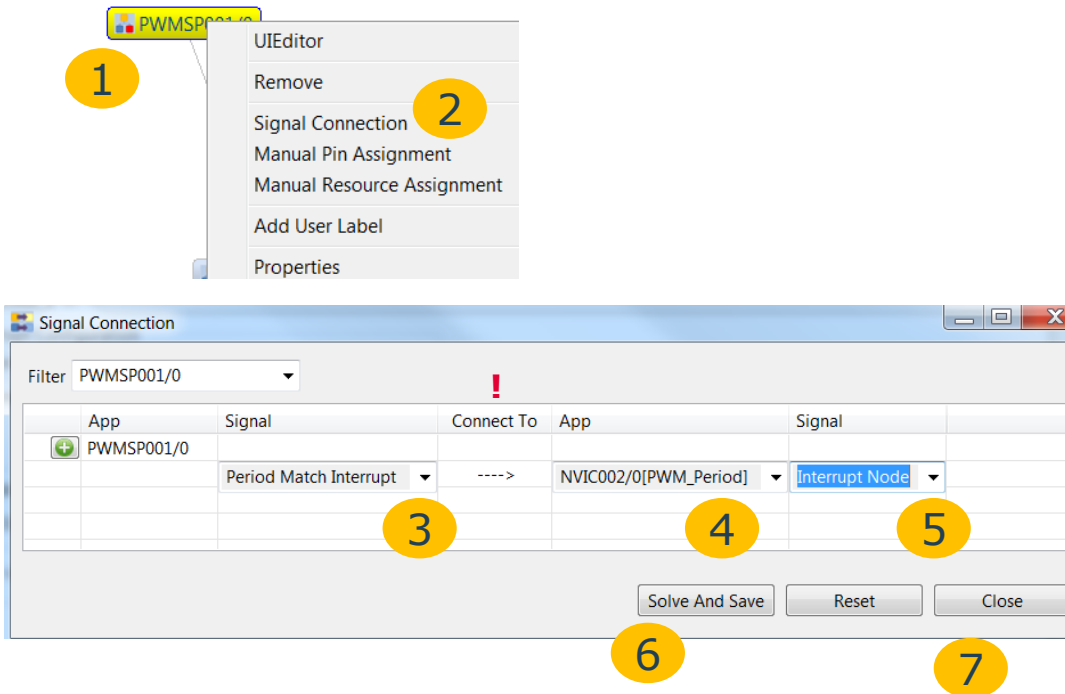


3 5 The function name for the interrupt handler has to be added.
This name has to be used to define the interrupt function in Main.c.

Signal Connection: Period Match to Interrupt

- One of the advanced features of DAVE™ 3 is the option for the user to connect HW*) signals between different DAVE™ Apps
- An important use case for this feature is the connection of an event signal with the interrupt signal of an NVIC (interrupt) App
- When doing the signal connection the direction of the signal has to be considered !

Connection of the period match event signal of the PWMSP001 App with the interrupt signal of an NVIC002 App



The screenshot shows the 'Signal Connection' dialog box. The 'Filter' is set to 'PWMSP001/0'. The table below shows the connection between the 'PWMSP001/0' app and the 'NVIC002/0[PWM_Period]' app. The 'Signal' column for the source app is 'Period Match Interrupt', and the 'Signal' column for the destination app is 'Interrupt Node'. The 'Connect To' column shows a connection from the source to the destination. The 'Solve And Save' button is highlighted, indicating the next step in the process.

App	Signal	Connect To	App	Signal
PWMSP001/0	Period Match Interrupt	----	NVIC002/0[PWM_Period]	Interrupt Node

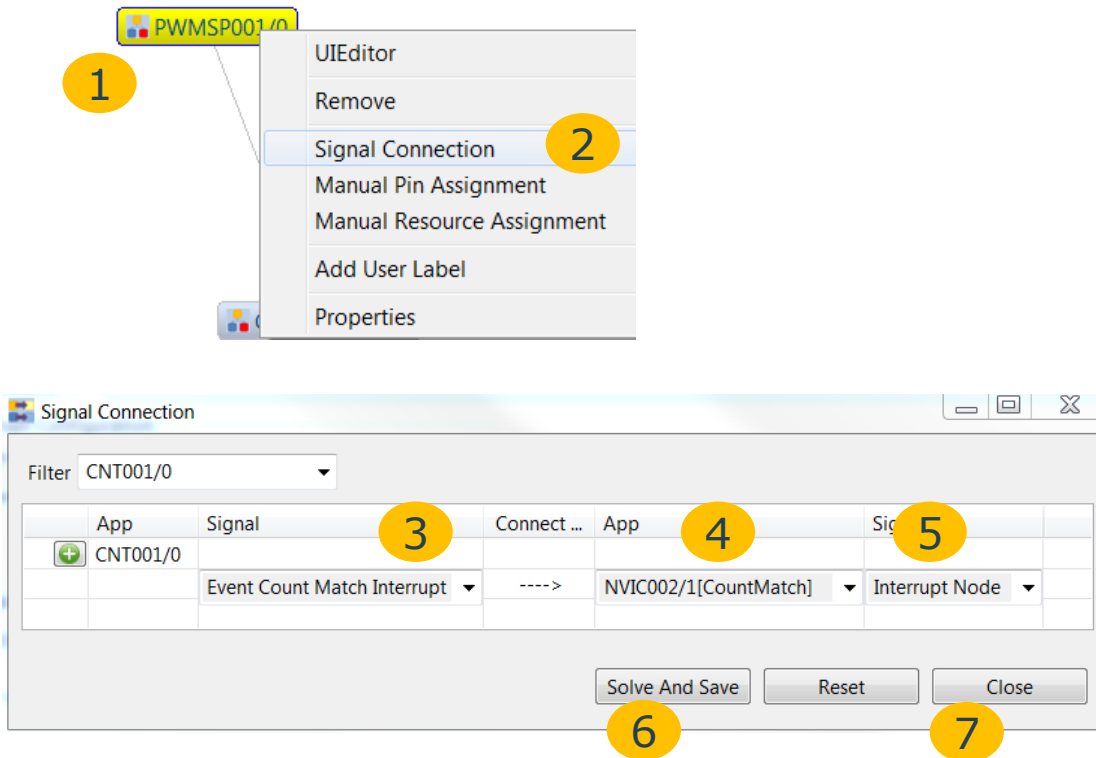
- 1 Right mouse click on the DAVE™ App that provides the source signal.
- 2 Select: Signal Connection.
- 3 Select the Period Match Interrupt signal.
- 4 Select the DAVE™ App that provides the destination signal (benefit of user label).
- 5 Select: Interrupt Node.
- 6 Click: Solve And Save then the resource solver starts.
- 7 Click: Close, after the resource solver is finished.

*) before resource solving the HW signals are logical signals, after solving it is assigned to a real HW signal.

Signal Connection: Count Event Match to Interrupt

The Event Count Match signal of the CNT001 App has to be connected to another instance of the NVIC002 App (the instance with the user label "CountMatch").

The procedure is similar as outlined on the previous page before:

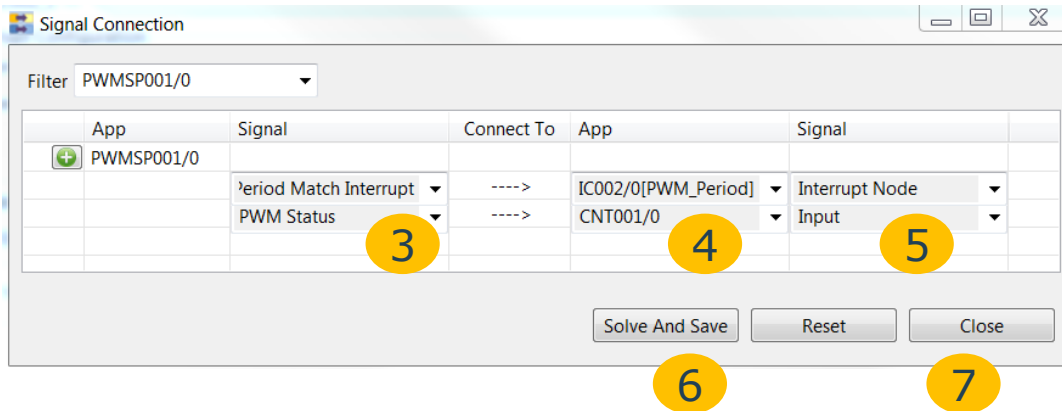
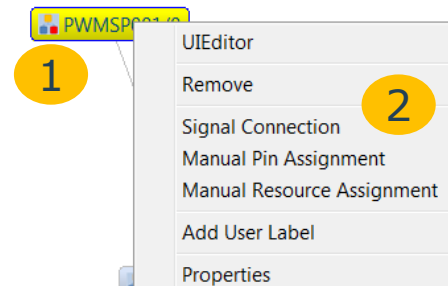


- 1 Right mouse click on the DAVE™ App that provides the source signal.
- 2 Select: Signal Connection.
- 3 Select the Count Match Interrupt signal.
- 4 Select the DAVE™ App that provides the destination signal (user label CountMatch).
- 5 Select the NVIC Interrupt.
- 6 Click: Solve And Save then the resource solver starts.
- 7 Click: Close, after the resource solver is finished.

Signal Connection: PWM Status Signal as input for the Count App

The PWM status signal of the PWMSP001 App has to be connected to the input signal of the CNT001 App.

The procedure is similar as outlined on the previous page before:



1 Right mouse click on the DAVE™ App that provides the source signal.

2 Select: Signal Connection.

3 Select the PWM Status signal.

4 Select the DAVE™ App that provides the destination signal (CNT001/0).

5 Select the Input signal.

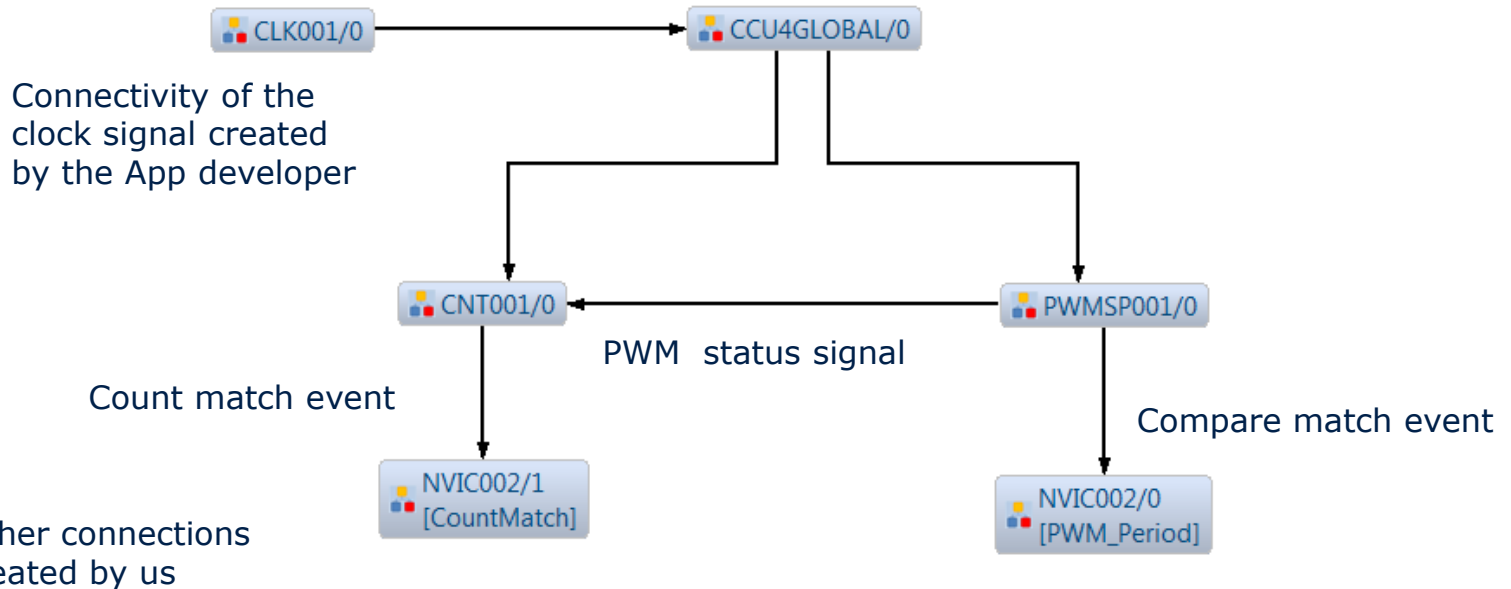
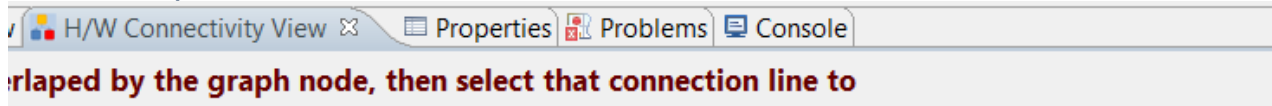
6 Click: Solve And Save then the resource solver starts.

7 Click: Close, after the resource solver is finished.

H/W Connectivity View

In addition to the S/W App Connectivity View and App Dependency TreeView shown in page 17 there is another graphical view for DAVE Apps called H/W Connectivity View. The H/W connectivity view shows the connection of signals / events between DAVE Apps.

Activation of the H/W Connectivity View

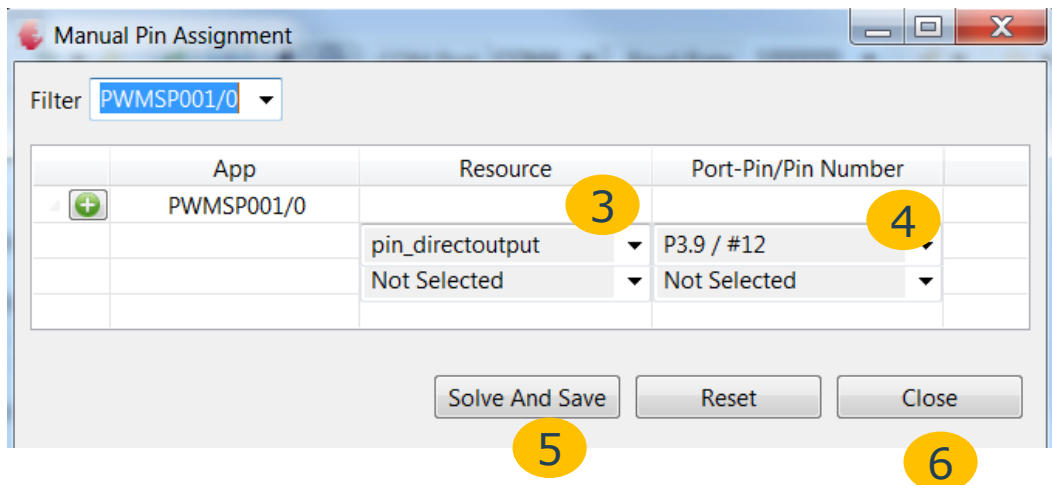
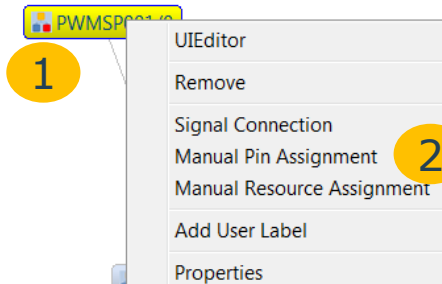


Manual Pin Assignment 1

- The resource solver integrated into DAVE™ assigns resources that are required by the DAVE™ Apps to the available chip resources
- The resources solver ensures that the resources assignments are conflict free and all connectivity requirements are fulfilled
- In particular for pad / pin assignments it is often required that specific constraints of the HW PCB design are also considered
- With the manual pin assignment functionality in DAVE™ such pin constraints for the resource solver can be defined

Manual Pin Assignment 2

- For each DAVE App that requires a pad / pin resource the manual pin assignment functionality is provided (e.g. right mouse click of the App)
- In our cases the PWMSP001 App requires the pad / pin resource to output the PWM signal
- Alternatively, if certain flexibility is required the pad / pin requirement can be disabled in the configuration UI of the PWMSP001 App (Pin Configuration tab) and the PWM Output signal can be connected to an IO002 App



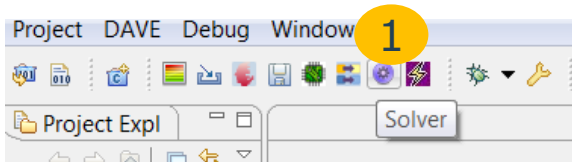
- 1 Right mouse click on the PWM App or any App the requires a pad/pin resource.
- 2 Select: Manual Pin Assignment.
- 3 Select a pin resource.
- 4 Select the port / pin where this pin resources should be assigned to *).
- 5 Click: Save.
- 6 Click: Close.

*) select a pin that is connected to LED.

Resource Solving and Code Generation

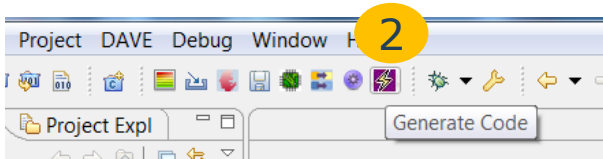
- The resource solver is a very innovative feature of DAVE™
- The resources solved ensures that the resources groups required by the DAVE™ Apps, including the signal connections and the pin constraints are assigned to the available peripherals and system resources of the XMC4500.

In complex solutions it might be possible that the resources requirements can not be solved. To understand when this point is reached it is recommended to periodically run the resource solver when making resource relevant changes in the project (adding DAVE™ Apps, connecting signals and defining pin constraints).



Press the “Solver” button to run the resource solver.

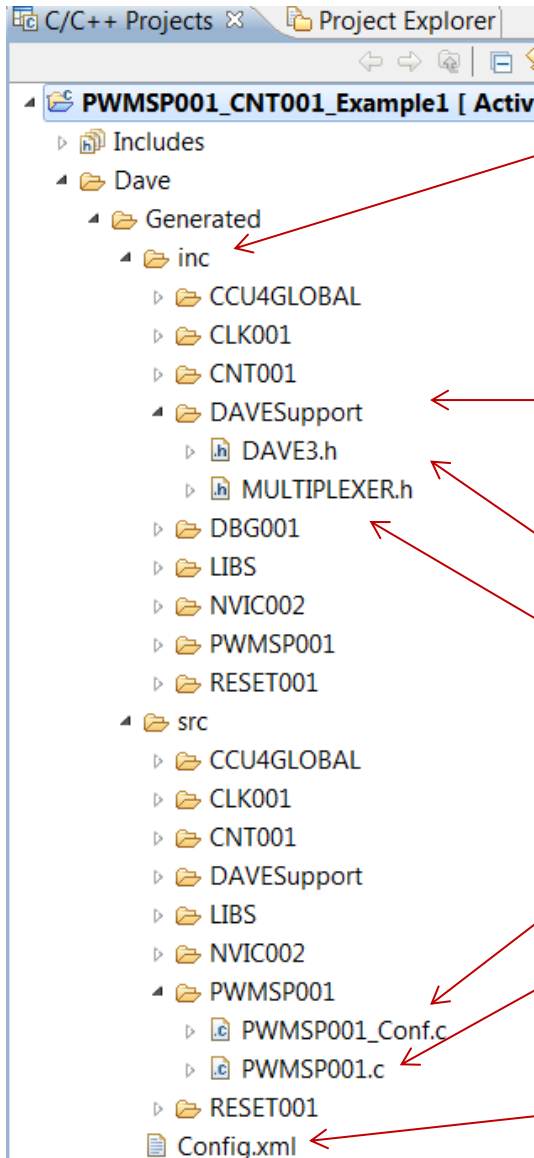
After finishing the work with DAVE™ Apps the library code that provides the initialization routines and the API to control the functionalities of the DAVE™ Apps has to be generated.



Press the “Generate Code” button to run the code generation.

To ensure that code generation is always based on a successful solver run, a solver run will be automatically issued each time before code generation is started.

The Generated Library Code



The generated library code is added to the project in the project folder: "Dave\Generated"

The include files are located under Dave\Generated\inc
The c source files are located under Dave\Generated\src

Each DAVE App has its own sub folder in the "inc" and the "src" folders (also hidden DAVE Apps are shown here).

DAVESupprt App is one of the hidden Apps that is included in each project, this DAVE App provides a consolidated header file (DAVE3.h) and specific top level initialization code.

Usually there are two header files for each DAVE™ App:

<appname>_Conf.h : contains the declaration of the global handle for the different DAVE™ App instances.

<appname>.h : contains the DAVE™ App instance independent declaration of global variables and the API and it contains macros.

Usually there are two source files for each DAVE™ App:

<appname>_Conf.c: contains the data structure (handle) for each DAVE™ App instance.

<appname>.c: contains the library functions (API.)

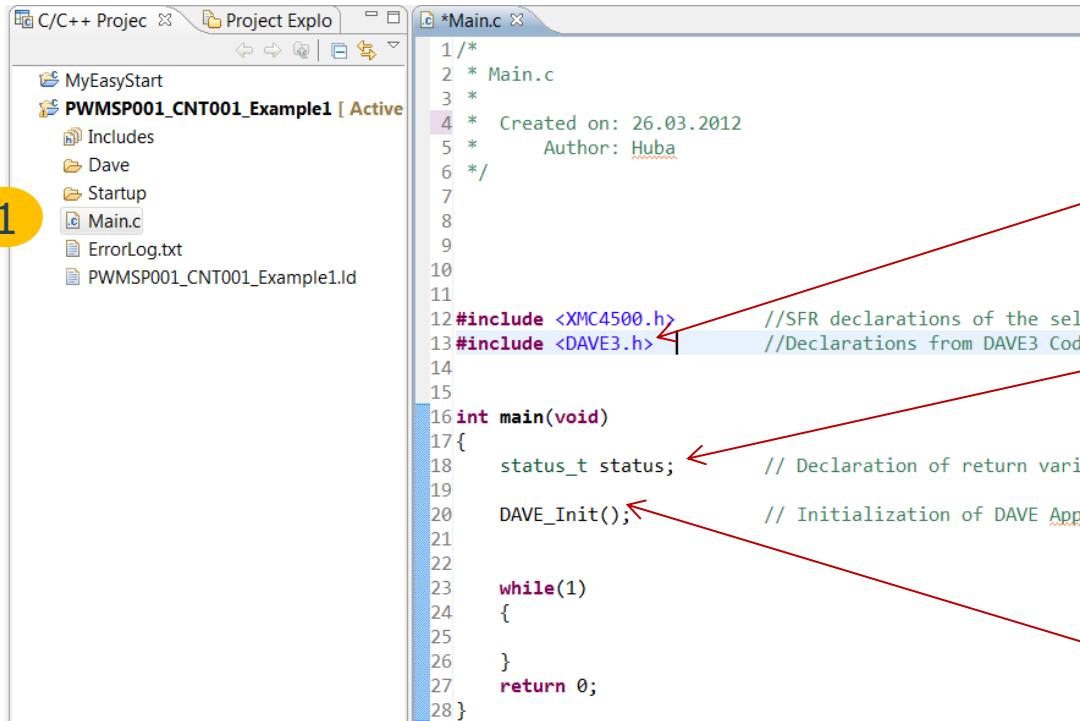
Object Orientation:

Code is independent of number of instances per DAVE™ App.

XML that contains information of the generated code that can be used by third party tools.

Using the API from the DAVE™ Apps to implement the required SW functionality

The Main.c that has been created by selecting the DAVE CE project type upon new project creation contains already includes and init functions to use the API of the used DAVE™ Apps.



```
1 /*
2  * Main.c
3  *
4  * Created on: 26.03.2012
5  * Author: Huba
6  */
7
8
9
10
11
12 #include <XMC4500.h> //SFR declarations of the sele
13 #include <DAVE3.h> //Declarations from DAVE3 Code
14
15
16 int main(void)
17 {
18     status_t status; // Declaration of return varia
19     DAVE_Init(); // Initialization of DAVE Apps
20
21     while(1)
22     {
23
24     }
25
26     return 0;
27 }
28 }
```

1 Double click to open Main.c in the editor area.

DAVE3.h contains all *.h files for the used DAVE™ Apps.

The local variable status can be used as return variable for the DAVE™ Apps APIs to indicate whether the function has been executed successful.

DAVE_Init contains all initialization functions of the used DAVE™ Apps .

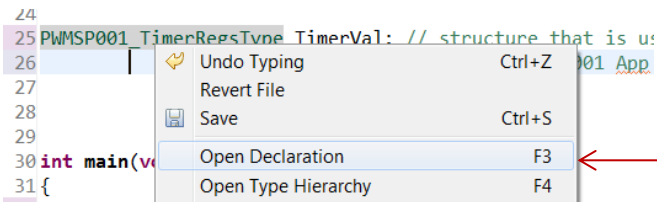
Adding User Code in Main.c: Global variables

This first part of Main .c shows all global variable definitions used in this project.

```
12 #include <XMC4500.h>           //SFR declarations of the selected device
13 #include <DAVE3.h>             //Declarations from DAVE3 Code Generation
14
15 float LED_duty = 0;            //duty cycle of the PWM output in percent
16
17 enum bright {up, down};        //enum declaration for direction to change the brightness of the LED
18 enum bright LED_bright = up;   //definition of LED-bright as either up or down
19
20 uint32_t PWMErrorcount = 0;    // accumulation of number of not successful executed API for the PWMSP001 App
21 uint32_t CNTerrorcount = 0;    // accumulation of number of not successful executed API for the PWMSP001 App
22
23 uint32_t CountVal;             // variable for value of the counter used by the CNT001 App
24
25 PWMSP001_TimerRegsType TimerVal; // structure that is used to read register values of the CCU4 slice used
26                                 // by the PWMSP001 App
27
```

Type unit32-t is declared in `stdint.h`.

PWMSP001_TimerRegsType is declared in `PWMSP001.h`.



With the right mouse option “Open Declaration” the respective header file can be automatically opened.

Adding User Code in Main.c: Function main

3

API Documentation

CCU4_PWMSinglePhase_PWMSP001 App

void	PWMSP001_Init (void) This function will initialize CCU4x_CCy global and slice registers w
status_t	PWMSP001_Deinit (const PWMSP001_HandleType *HandlePtr) This function will reset CCU4x_CCy slice registers with default val
status_t	PWMSP001_Start (const PWMSP001_HandleType *HandlePtr) This function will start the Single Phase PWM APP which will in t also enables the interrupt and clears the IDLE mode of the CCU. This function needs to be called to start the App even if Ext
status_t	PWMSP001_Stop (const PWMSP001_HandleType *HandlePtr) This function will stop the Single Phase PWM APP which will stop and set the IDLE mode of the CCU.
status_t	PWMSP001_SetCompare (const PWMSP001_HandleType *H This function will update the duty cycle of the output waveform Duty cycle is given in terms of the compare register value a
status_t	PWMSP001_SetDutyCycle (const PWMSP001_HandleType * This function will update the duty cycle of the output waveform Duty cycle is given in terms of the percentage.
status_t	PWMSP001_SetPeriod (const PWMSP001_HandleType *Han This function will modify the PWM frequency. PWM frequency is given in terms of the PWM period. In case of timer concatenation, given value is split into two Second and First slice. e.g. 0x80000010 value is written as 0x8000 as Period regis register of first slice. Total PWM period is ((0x8000 + 1) * 0
status_t	PWMSP001_SetPwmFreq (const PWMSP001_HandleType *H This function will modify the PWM frequency. PWM frequency is given in Hertz.

The function main in Main.c

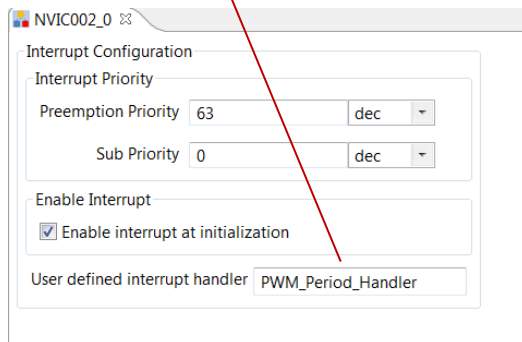
```
30 int main(void)
31 {
32     status_t status;           //declaration of return variable for DAVE3 APIs
33
34     DAVE_Init();               //initialization of DAVE Apps
35
36
37     status = PWMSP001_Start(&PWMSP001_Handle0); // start of the PWMSP001 App, instance 0
38     if (status != DAVEApp_SUCCESS) PWMErrorcount++; // if start was not successful PWM error counter incremented
39     status = CNT001_Start(&CNT001_Handle0); // start counting with the CNT001 App instance 0
40     if (status != DAVEApp_SUCCESS) CNTErrorcount++; // if start was not successful Count error counter incremented
41
42
43
44
45
46     while(1)
47     {
48         // Example to read out HW registers of the hW that is used by the respective DAVE App
49         // the content of timerVal and CountVal might be checked with the debugger
50
51         status = PWMSP001_GetTimerRegsVal((PWMSP001_HandleType*)&PWMSP001_Handle0, &TimerVal);
52         if (status != DAVEApp_SUCCESS) PWMErrorcount++;
53         status = CNT001_GetEvtCountValue((CNT001_HandleType*)&CNT001_Handle0, &CountVal);
54         if (status != DAVEApp_SUCCESS) CNTErrorcount++;
55
56     }
57     return 0;
58 }
59 }
```

- 1 We have to start the PWM and count App because we didn't check the option "start after initialization" in the UI to configure the DAVE App. The first parameter of the function is in most of the cases the pointer to the data structure instance of the used DAVE App instance.
- 2 Checking the return variable of the DAVE Apps API is just for demo purposes it has no impact in the program flow, The same is the case of the instructions in the while(1) loop.
- 3 The API definition and syntax can be found in the help document of each DAVE™ App, before adding your own code, please check page 37: code completion.

Adding User Code in Main.c: Interrupt Handlers for PWM Timer Period Match

```
62 /**
63  * @brief Interrupt Handler for the Period Interrupt of the PWMSP001 App \n
64  *
65  * @param[in] none\n
66  * @return none\n
67  *
68  * <b>Reentrancy: no</b><BR>
69  *
70  */
71
72 void PWM_Period_Handler(void) {
73
74
75     status_t status;          //declaration of return variable for DAVE3 APIs
76
77     switch (LED_bright) {      //check whether the LED should increase or decrease its brightness
78     case up: { if (LED_duty > 99)
79                 LED_duty = 0;
80                 LED_duty++;
81             }
82             break;
83     case down: { if (LED_duty < 1)
84                  LED_duty = 100;
85                  LED_duty--;
86             }
87     }
88
89     status = PWMSP001_SetDutyCycle(&PWMSP001_Handle0, LED_duty); //change brightness of LED
90     if (status != DAVEApp_SUCCESS) PWMErrorcount++; //if set of new duty cycle was not successful
91                                           //PWM error counter incremented
92 }
```

Logic to change the brightness.

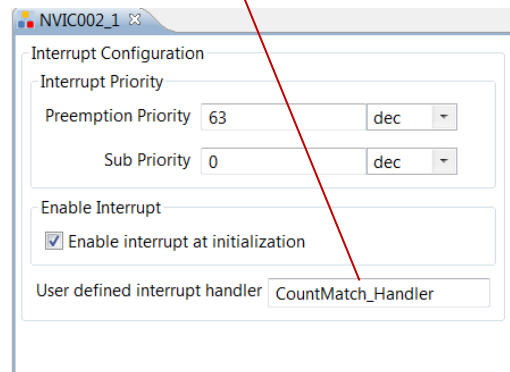


Here we have to define a function with the same name as defined in the configuration UI of the NVIC002/0 App.

Adding User Code in Main.c: Interrupt Handlers for Count Event Match

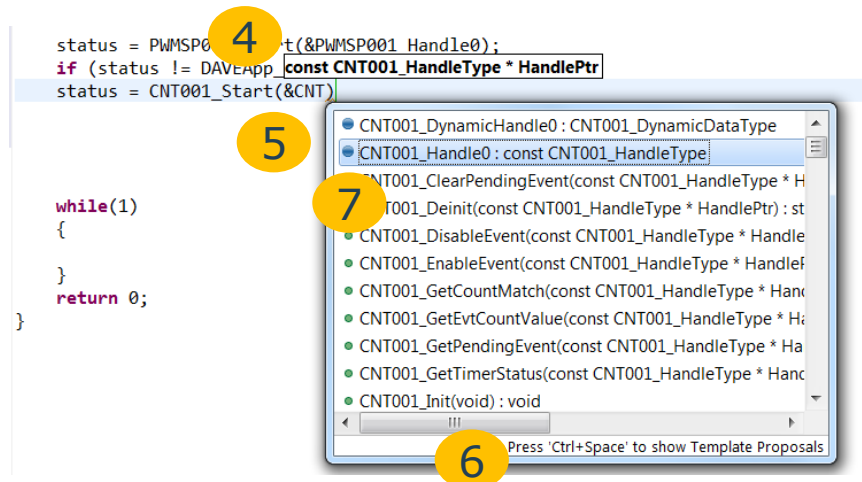
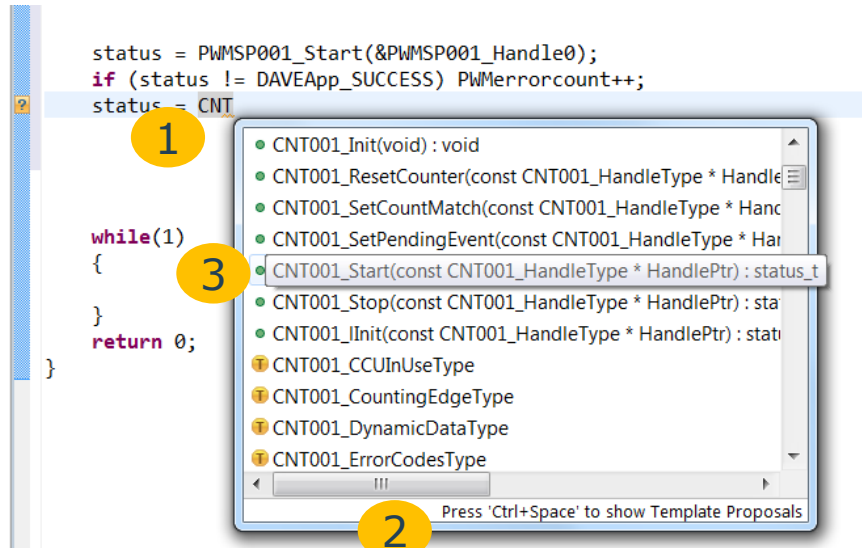
```
95 /**
96  * @brief Interrupt Handler for the count match event of the CNT001 App \n
97  *
98  * @param[in] none\n
99  * @return none\n
100  *
101  * <b>Reentrancy: no</b><BR>
102  *
103  */
104
105 void CountMatch_Handler(void) {
106     status_t status; // Declaration of return variable for DAVE3 APIs
107
108     switch (LED_bright) { // change direction of LED brightness change
109         case up: LED_bright = down;
110                 LED_duty = 100;
111                 break;
112         case down: LED_bright = up;
113                   LED_duty = 0;
114     }
115
116     status = CNT001_Stop(&CNT001_Handle0); //CNT001 App instance 0 stops counting
117     if (status != DAVEApp_SUCCESS) CNTErrorcount++; //if stop was not successful error counter will be incremented
118     status = CNT001_ResetCounter(&CNT001_Handle0); //counter of the CNT001 App instance 0 is set to 0
119     if (status != DAVEApp_SUCCESS) CNTErrorcount++; //a problem in version 1.0.0 of the CNMT001 App lead to an increment of the error counter
120     status = CNT001_Start(&CNT001_Handle0); //CNT001 App instance 0 starts counting
121     if (status != DAVEApp_SUCCESS) CNTErrorcount++; //if start was not successful error counter will be incremented
122 }
```

Logic to change direction of the brightness change.



Here we have to define a function with the same name as defined in the configuration UI of the NVIC002/1 App.

Code Completion Feature in the Eclipse Editor to support using the APIs of the DAVE™ Apps correctly



- 1 Type in the first letters of the DAVE™ App from which an API should be used (all API start with the name of the respective DAVE™ App).
- 2 Press Ctrl plus Space, then a window with all options to complete this instruction shows up.
- 3 Select the required API (function) and press return,
- 4 then the function name will be completed and the required parameter types will be shown.
- 5 Type the first letter of the DAVE™ App name after the address operator(&).
- 6 Press Ctrl plus Space, then a window with all options to complete the first parameter shows up.
- 7 Select the appropriate parameter and press return, then the first parameter is completed.

Building,...

The complete DAVE project can also be [Downloaded](#) as zip file .
Uses the keyword “Tutorial” to select this projects from the list of all available example projects.

Instructions to import the downloaded zip file to your workspace:

-> File ->Import ->Infineon ->DAVE Project ->Next

Then check “Select Archive File” and browse to the downloaded zip file.

To test this project on the real HW the following steps have to be performed:

- Building the project (page 7)
- Downloading and debugging the project (page 8)
The debugger allows to watch the different variables and assessing the functionality of this small project

Please note, that the optimization settings in the projects are set to –O0 (no optimization) therefore the code sizes is fairly large. To create productive code we recommend –O2 or –Os and to check the option to remove unused sections in the linker settings.

In case you need technical support please check out the [DAVE Forum](#).



ENERGY EFFICIENCY MOBILITY SECURITY

Innovative semiconductor solutions for energy efficiency, mobility and security.

