XMC Dev Day 2015

Hands-on with DAVEv4 & XMClib

Alberto Squarcina IFI ATV SMD EMEA
Alessandro Manfre IFI ATV SMD EMEA

Agenda

- LLD of XMCLib: use of ACMP
- Blink an LED with PWM APP
- Use ADC APP to update PWM dutycycle

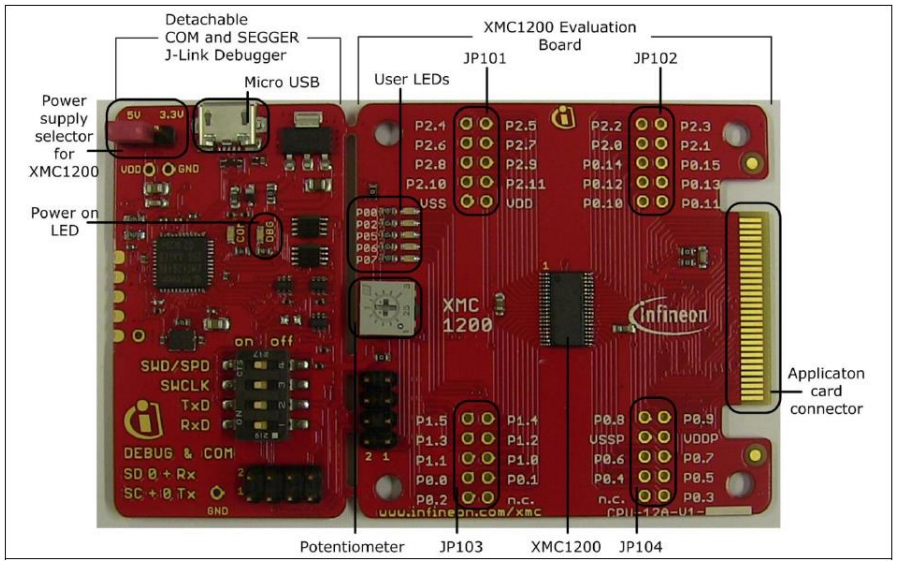
Agenda

- LLD of XMCLib: use of ACMP
- Blink an LED with PWM APP
- Use ADC APP to update PWM dutycycle

Summary

- Required XMC kit
- HOT Session: Key points
- ACMP: details and connections
- Starting DAVE
- Work with LLD: overview
- Work with LLD: details
- Digital IO configuration & Alternate Functions

Required XMC kit



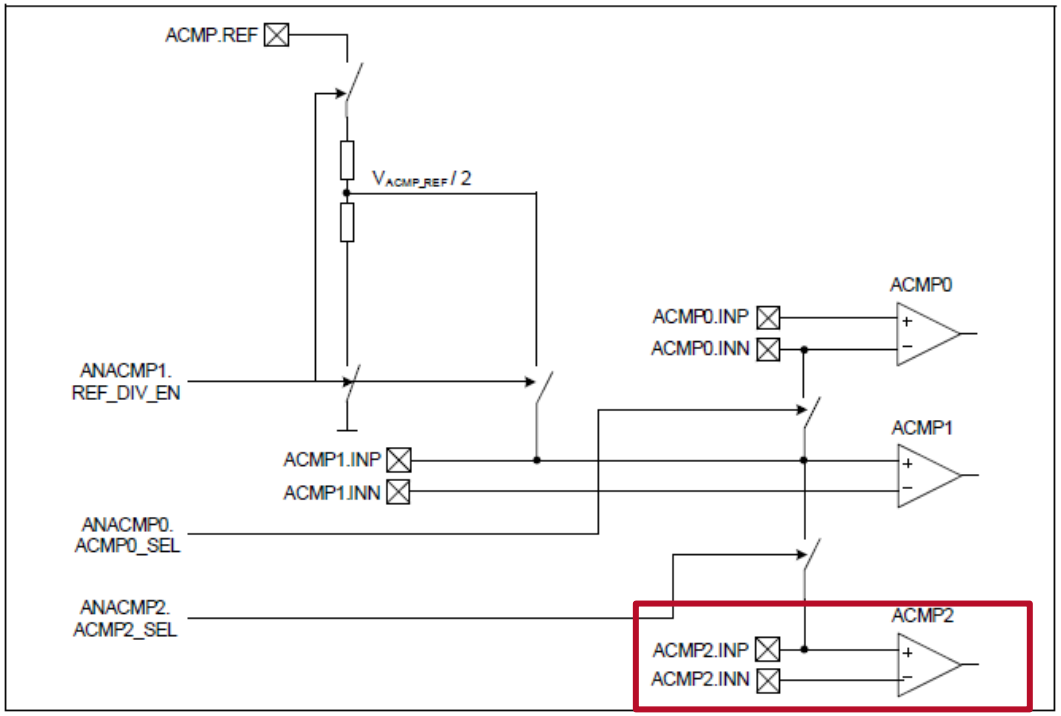
- In this tutorial is used the XMC 1200 Boot Kit equipped with an XMC1200-T038x0200 and Segger J-Link on-board debugger (OBD)
- 2 jumper cables M-F are mandatory

HOT Session: Key points

- The target for the example of HOT session is to use the ACMP peripheral, in combination with a Digital IO
- The example is made mixing the use of XMCLib by LLD and the APPs of DAVE4
- The example is made in order to see the Alternate Functions (both as Input or Output) of each pin
- Output of the example is the switch ON/OFF of a LED accordingly to the output of ACMP

ACMP: details [1/2]

- In the XMC1200 there are up to 3 analog comparators. In this session ACMP2 will be used
- Pins used
  - Inputs: P2.1,P2.2
  - Output: P0.5

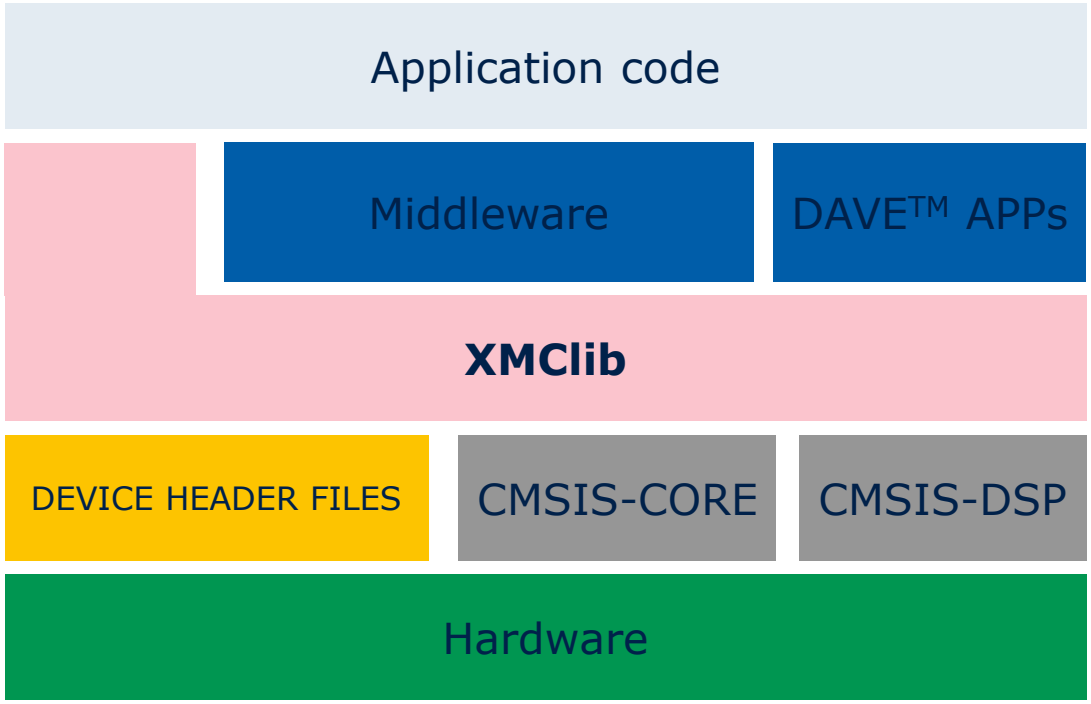


ACMP: details [2/2]

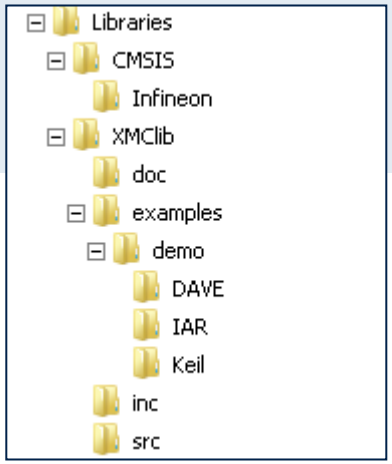
Input/Output	I/O	Connected To	Description
ACMP0.INN	I	P2.8	"-" input of ACMP0
ACMP0.INP	I	P2.9	"+" input of ACMP0
ACMP1.INN	I	P2.6	"-" input of ACMP1
ACMP1.INP	I	P2.7	"+" input of ACMP1
ACMP2.INN	I	P2.2	"-" input of ACMP2
ACMP2.INP	I	P2.1	"+" input of ACMP2
ACMP.REF	I	P2.11	Reference input of ACMP
ACMP0.OUT	O	P0.10 P2.10 ERU0.0A0 BCCU0.IN5	output of ACMP0
ACMP1.OUT	O	P1.0 ERU0.1A0 BCCU0.IN0 P2.5.HW0 pull control	output of ACMP1
ACMP2.OUT	O	P0.5	output of ACMP2
		P1.2 ERU0.2A0 BCCU0.IN3 P2.3.HW0 pull control	

Work with LLD: overview

- XMC Lib consists of low-level drivers for the XMC product family peripherals
- Collection of routines and data structures which covers all peripheral functions
- Peripheral register abstraction by a set of stateless APIs

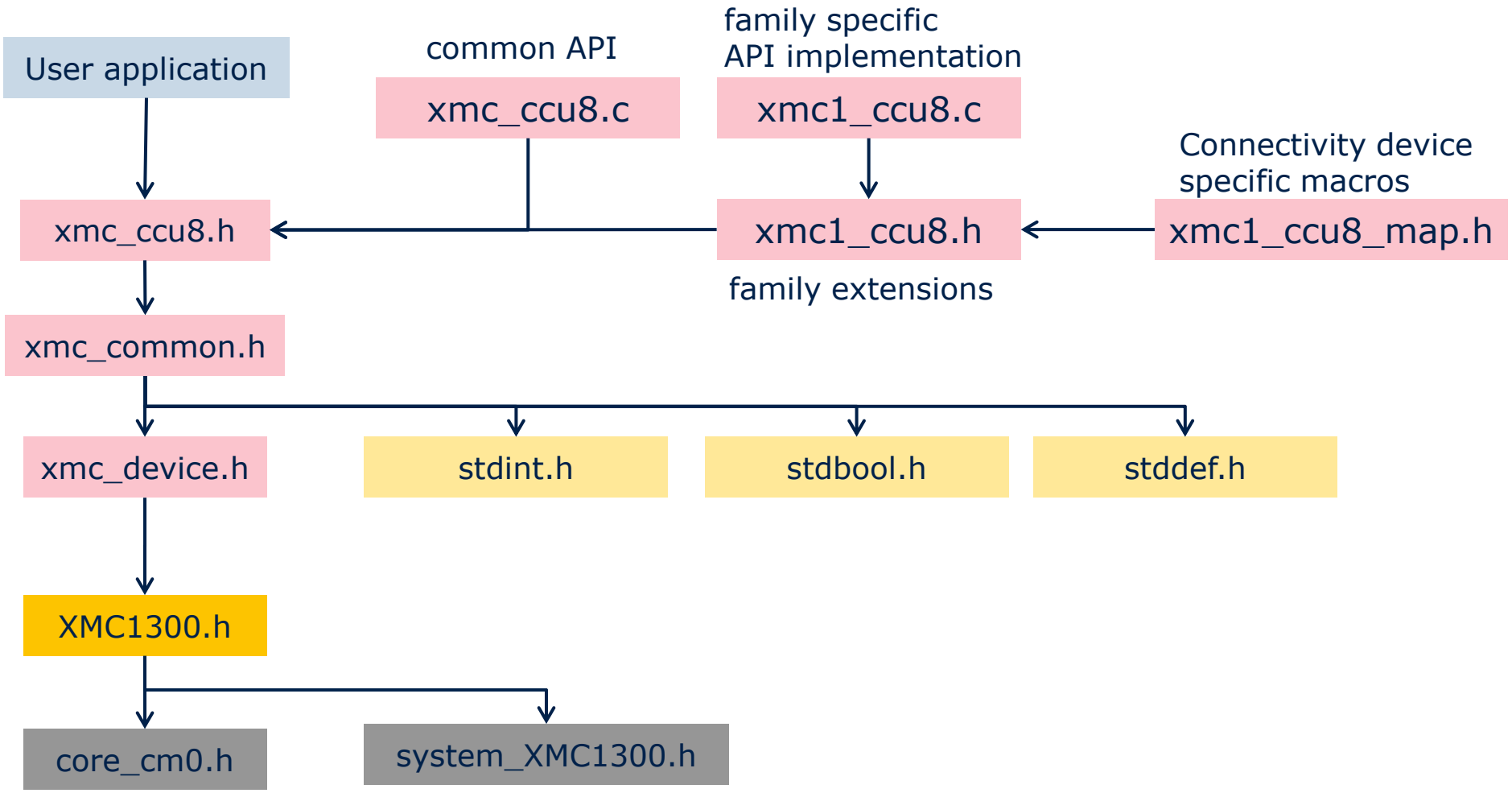


Package Description



- **inc** folder contains peripheral API header files
  - *xmc_common.h*: common functionality and definitions
  - *xmc_device.h*: device selection
  - *xmc_<peripheral>.h* (one header file per peripheral): Function prototypes, data structures and enumeration
  - *xmc<family>_<peripheral>.h* (optional): Family extensions
- **src** folder contains peripheral API implementation
  - *xmc_common.c*: common functionality and definitions
  - *xmc_<peripheral>.c:* Implementation of common peripheral functionality
  - *xmc<family>_<peripheral>.c* (optional): Family specific implementation
- **doc** folder contains API documentation generated from sources
- **examples** folder contains extensive set of examples projects for different toolchains

Package Description



Coding rules and conventions

- C99, in addtion unions and bitfields are used for more compact code and data section.
- Use only standard data types
- Use enumerations
- Naming convention:
  - Variables use only lower case,underscore separated words.
  - Functions use CamelCase convention.
- Non blocking APIs.
- Runtime error checking can be enabled by user
  - XMC_ASSERT() used to check input parameters of functions, result of calculations, …
  - XMC_DEBUG() used to monitor status of application

Coding rules and conventions

Object = attributes + methods

Attributes are the peripheral data struct.

Methods are functions that take a pointer to the peripheral data struct as the first argument.

XMC Lib uses naming conventions to bind the data struct and the functions that operate on it

XMC_PERI_DoSomething(XMC_PERI_t const *peri, …);

Each driver defines its own “namespace”, XMC_PERI.

XMC Lib APIs

- Each driver has an API to get at runtime the driver version

```
XMC_DRIVER_VERSION_t XMC_PERI_GetDriverVersion(void);
```

- Each driver has an initialization function

```
XMC_PERI_STATUS_t
XMC_PERI_Init(XMC_PERI_t const *peri,
              const XMC_PERI_CONFIG_t const *config);
```

- Most drivers have enable/disable function

```
void XMC_PERI_Enable(XMC_PERI_t const *peri);
void XMC_PERI_Disable(XMC_PERI_t const *peri);
```

XMC Lib APIs

- Event handling

```
void XMC_PERI_Enable(XMC_PERI_t const *peri);
void XMC_PERI_Disable(XMC_PERI_t const *peri);
void XMC_PERI_EnableEvent(XMC_PERI_t const *peri);
void XMC_PERI_DisableEvent(XMC_PERI_t const *peri);
void XMC_PERI_TriggerEvent(XMC_PERI_t const *peri);
uint32_t XMC_PERI_GetEventStatus(XMC_PERI_t const *peri);
void XMC_PERI_SetInterruptNode(XMC_PERI_t const *peri);
```

- Control

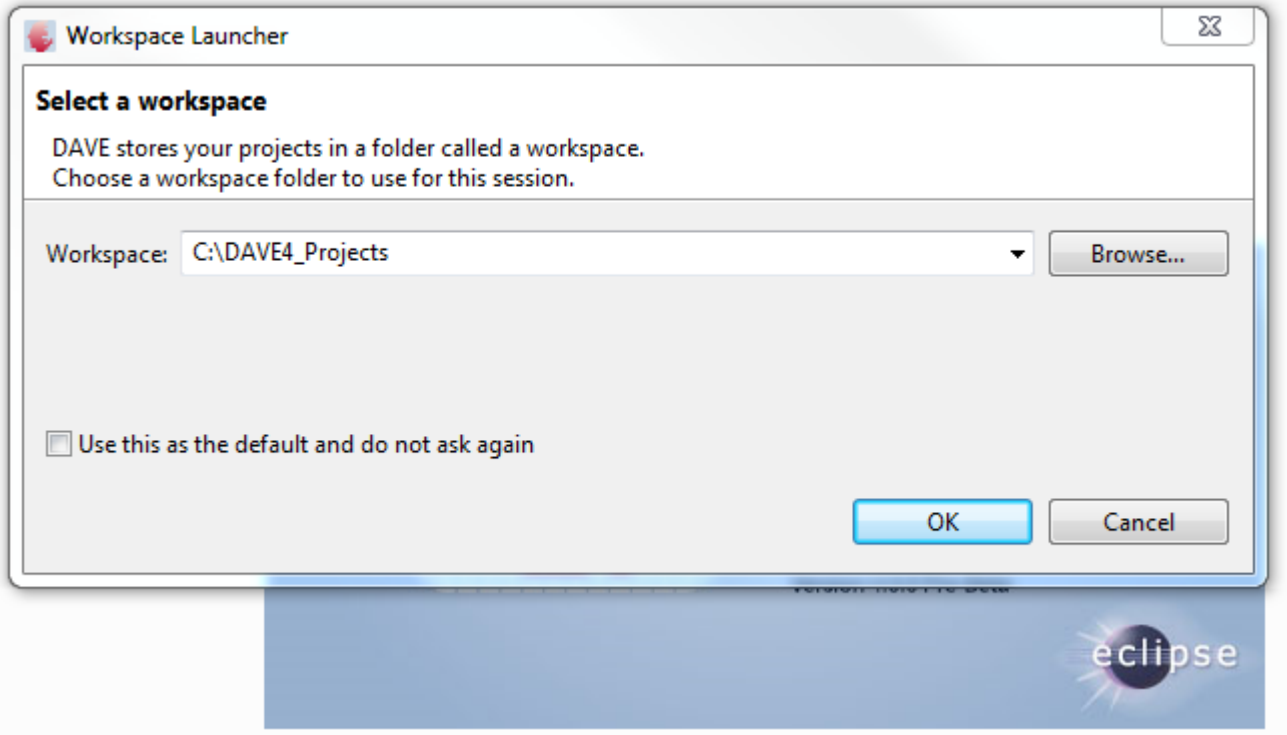
```
void XMC_PERI_Start(XMC_PERI_t const *peri);
void XMC_PERI_Stop(XMC_PERI_t const *peri);
void XMC_PERI_Suspend(XMC_PERI_t const *peri);
void XMC_PERI_Resume(XMC_PERI_t const *peri);
```

- Get/setters (run time API to get or modify state of peripheral)

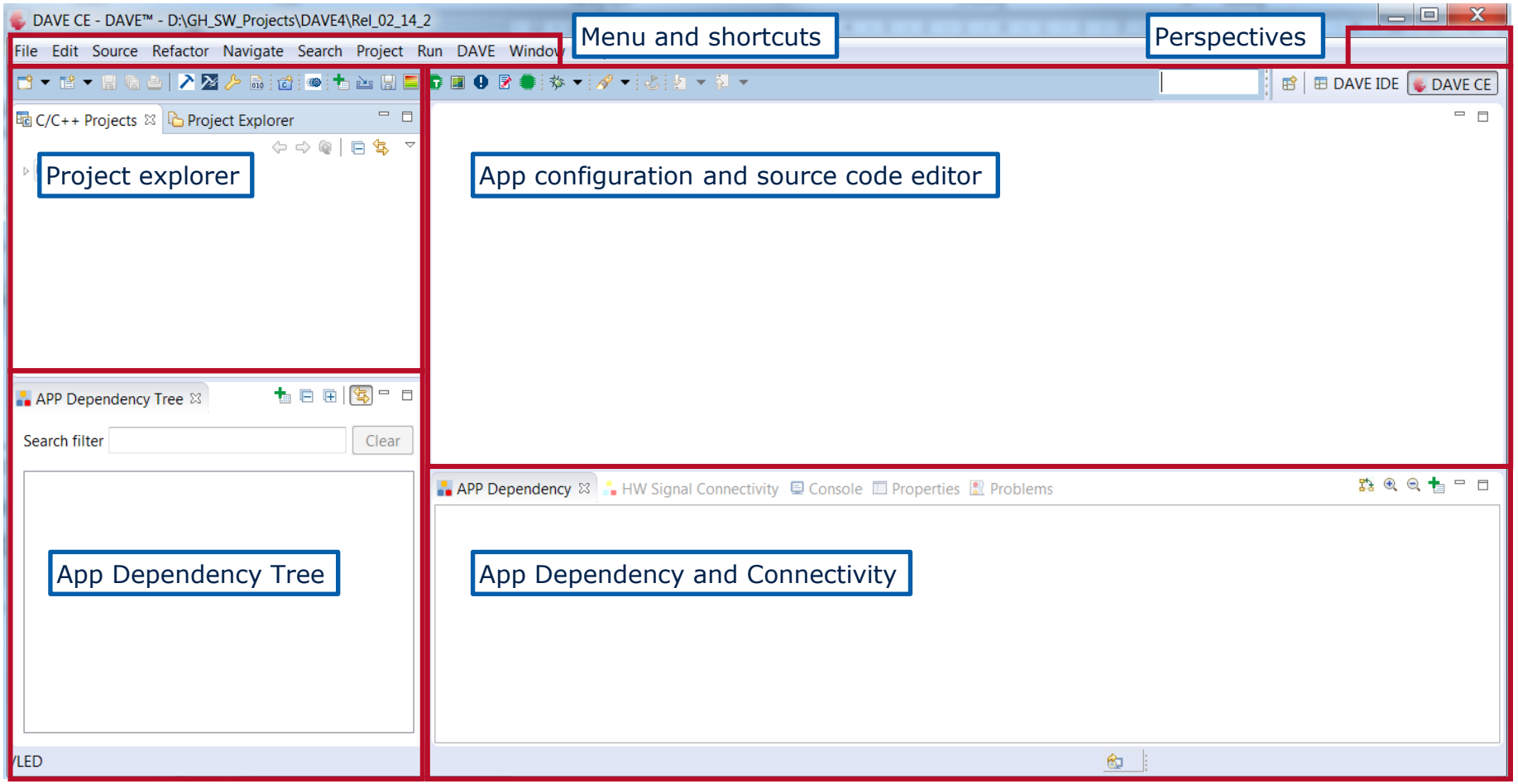
```
void XMC_PERI_SetSomething(XMC_PERI_t const *peri, …)
uint32_t XMC_PERI_GetSomething(XMC_PERI_t const *peri, …)
```

Starting DAVE™ for the first time

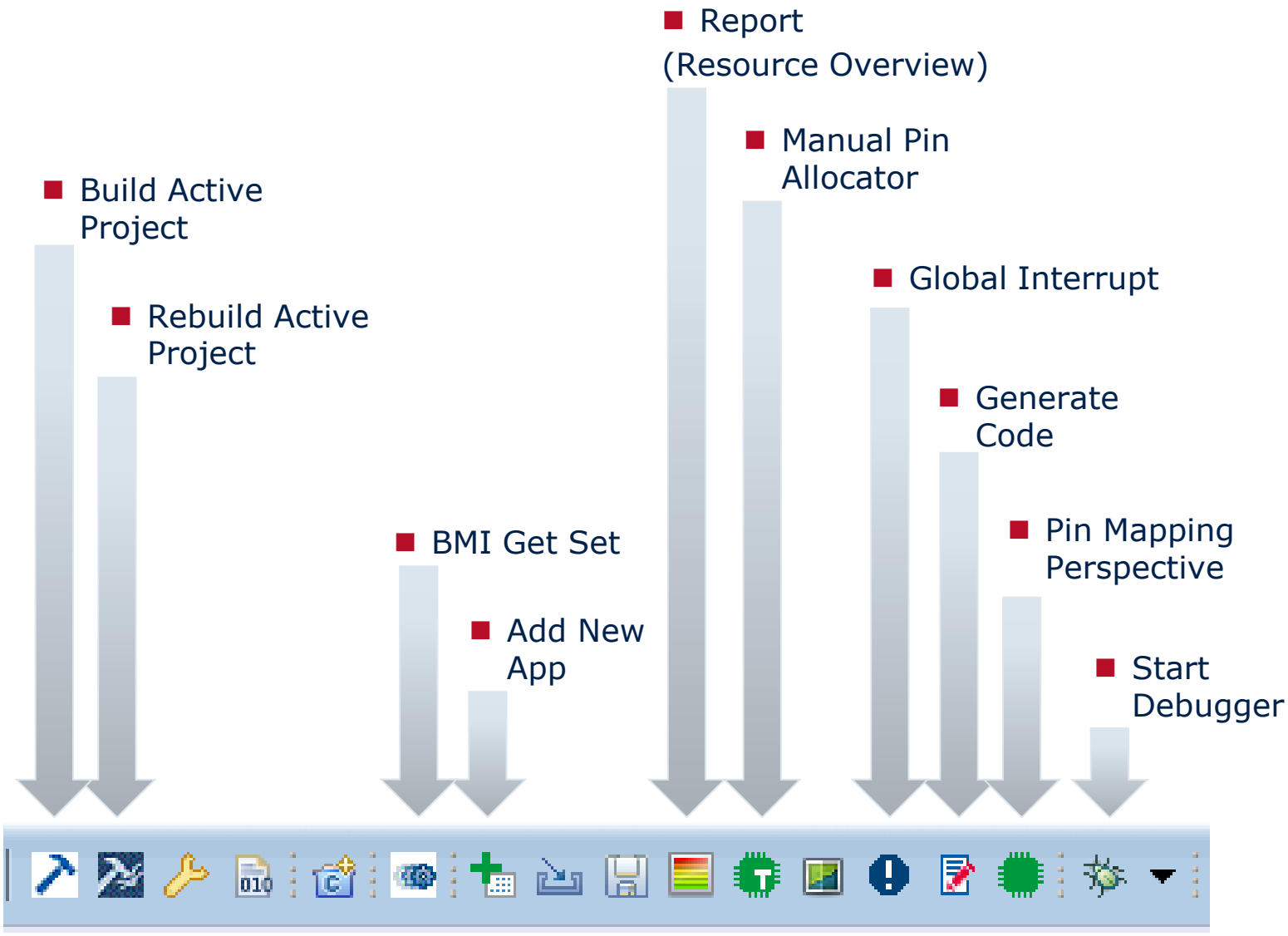
- Start DAVE
- Enter path to workspace folder
  - Please chose a new workspace folder, not an existing workspace folder form an earlier DAVE™ version



DAVE™ CE Workspace

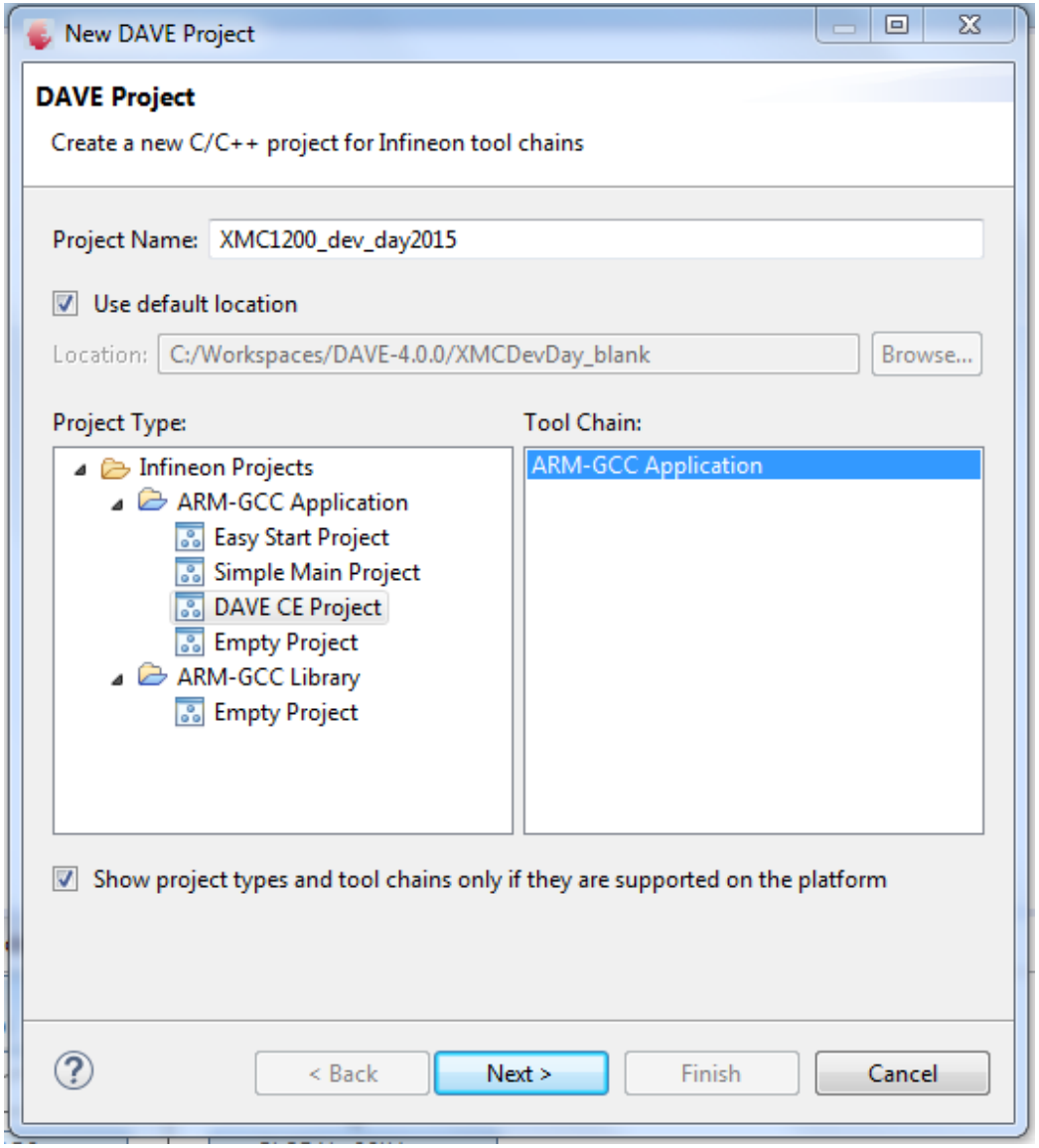


Tool Panel



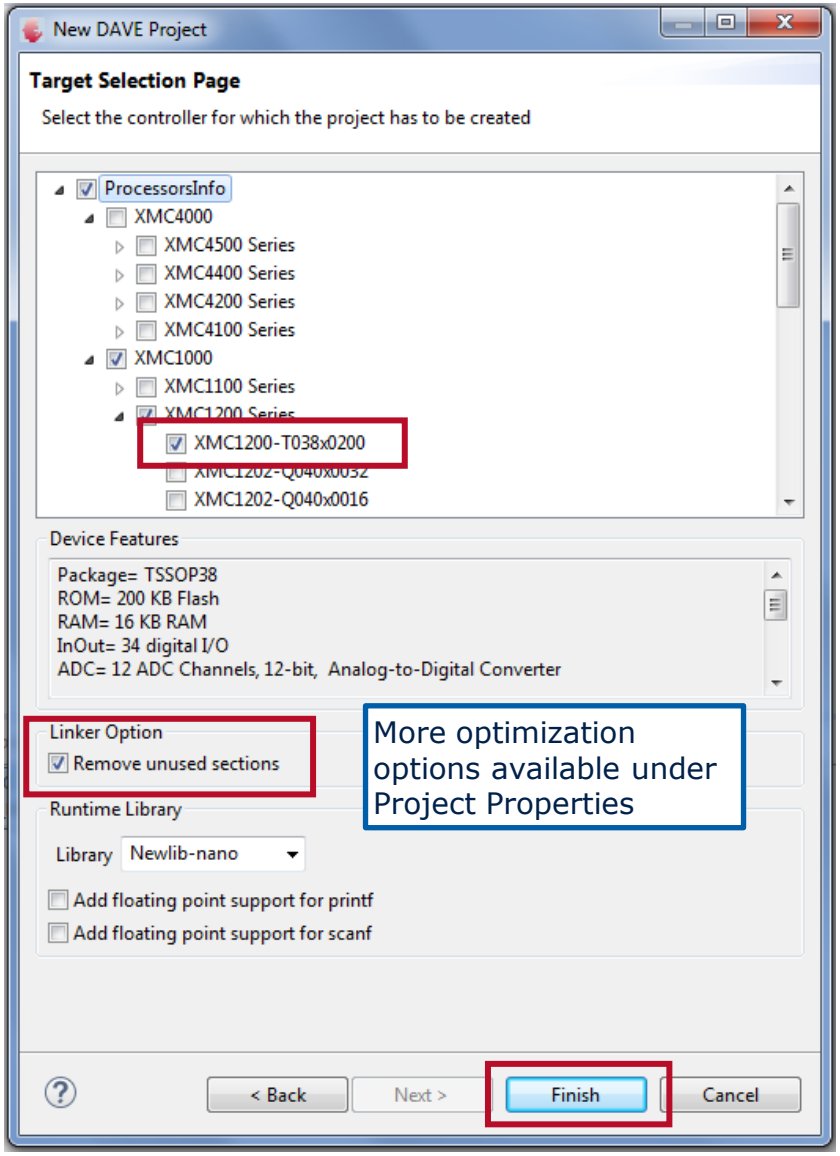
How to create a DAVE™ CE Project [1/2]

- Create DAVE™ Code Engine (CE) Project
  1. Go to File → New → DAVE Project
  2. Select DAVE CE Project
  3. Type a Project Name
  4. Click Next

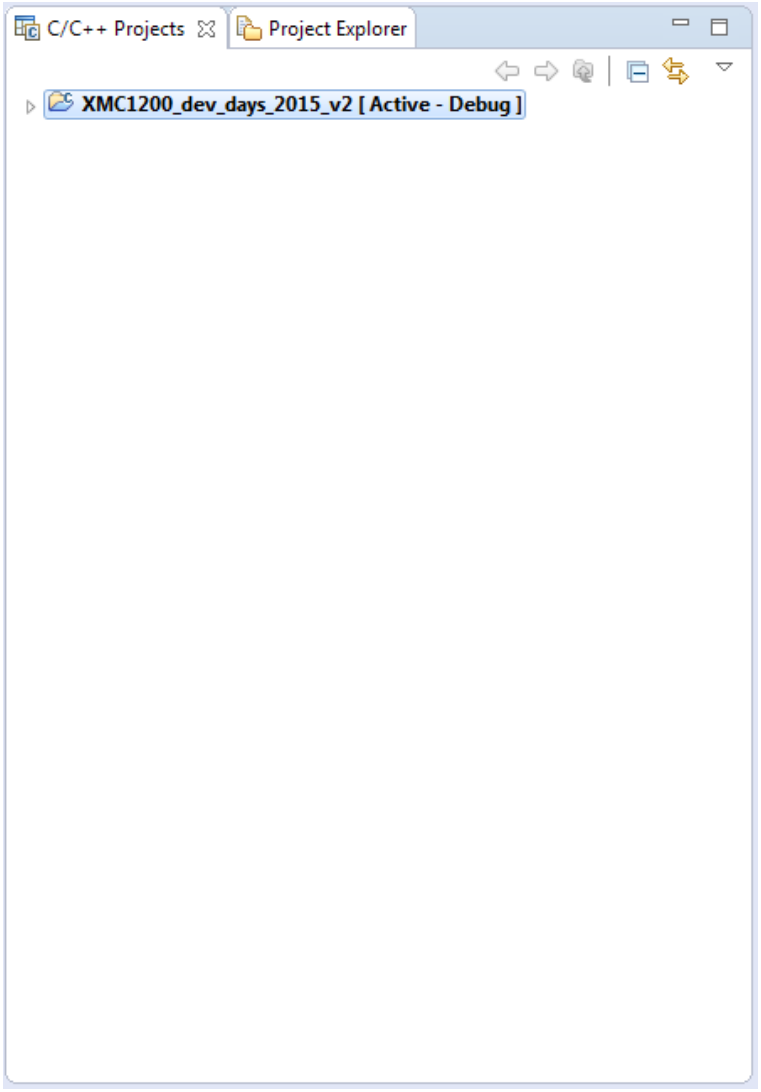


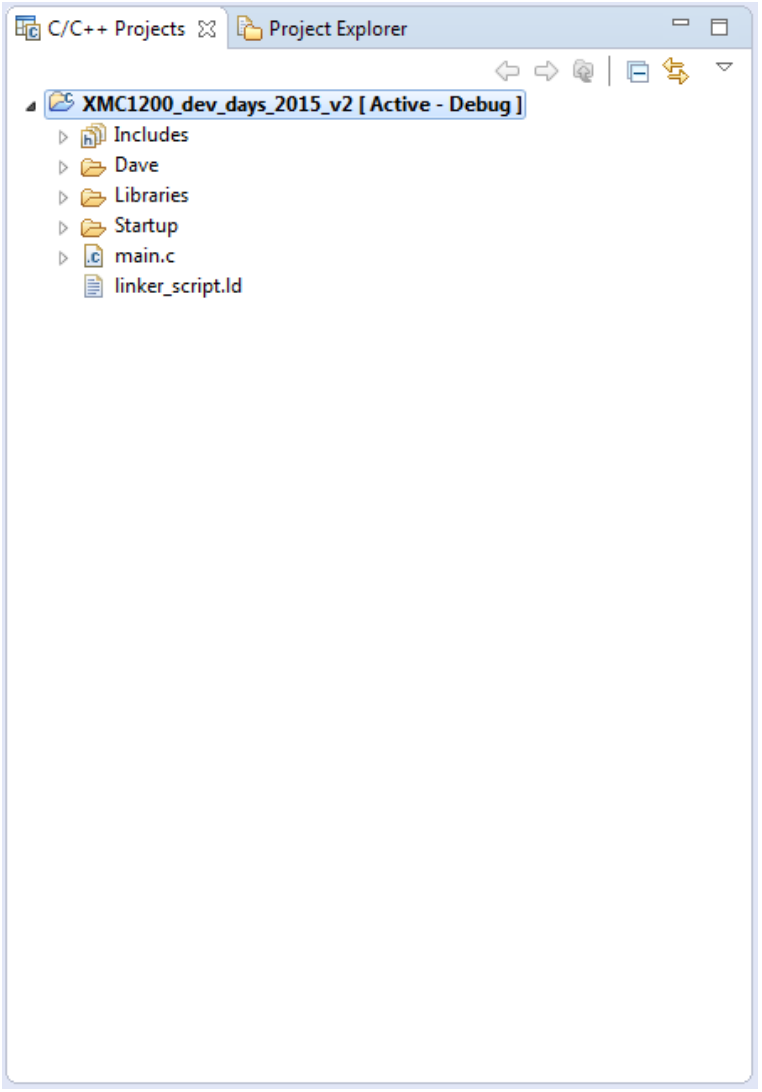
How to create a DAVE™ CE Project [2/2]

- Select the appropriate microcontroller
- In this case
  - XMC1200-T038x0200
- Click Finish



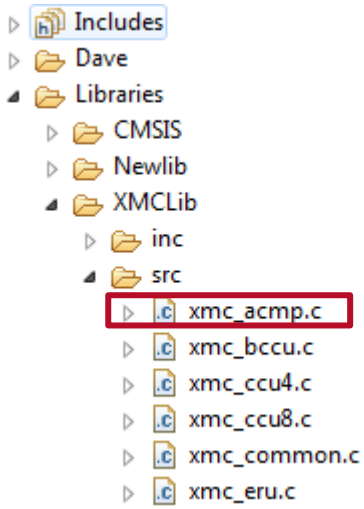
Project View



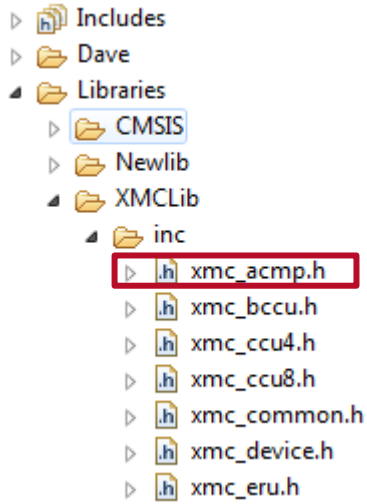


Work with LLD: project details [1/2]

- XMCLib is always available in each project
  - Source files



  - Header files

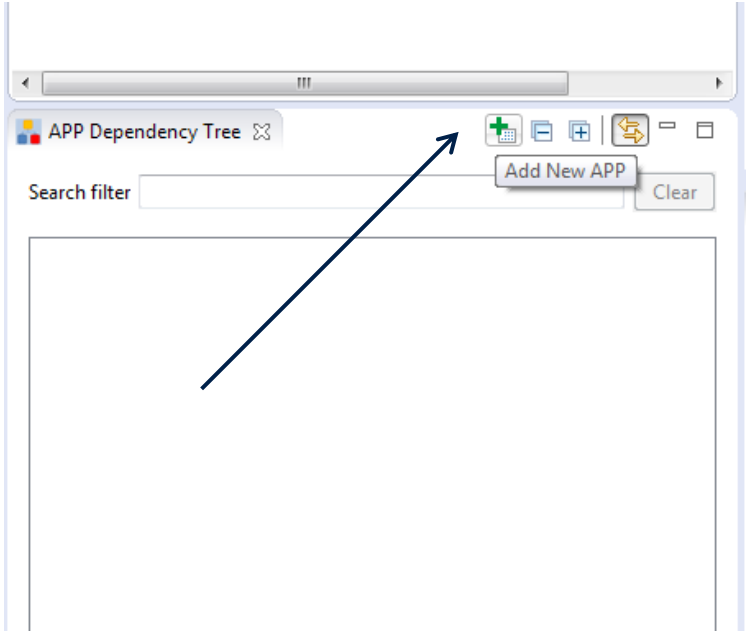


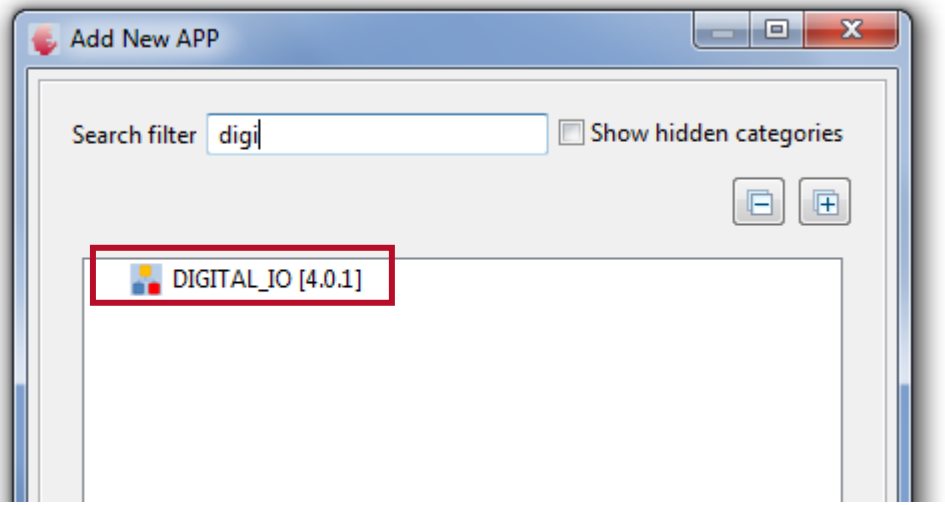
Work with LLD: project details [2/2]
LLD Code for ACMP

```
#include <XMC1200.h>
#include <DAVE.h>           //Declarations from DAVE Code Generation (includes SFR declaration)
#include <xmc_acmp.h>

#define ACMP_SLICE_NUMBER (2)

/* Hardware reference to ACMP Slice */
XMC_ACMP_t *peripheral;

/* ACMP Slice configuration */
XMC_ACMP_CONFIG_t g_acmp_config = {
                                    .filter_disable = 0U,
                                    .output_invert = 0U,
                                    .hysteresis = XMC_ACMP_HYSTERESIS_20
                                  };

int main(void)
{
  DAVE_STATUS_t status;

  peripheral = COMPARATOR;

  /* Enable and Initialize ACMP Slice-2 */
  XMC_ACMP_EnableComparator(peripheral,ACMP_SLICE_NUMBER);
  XMC_ACMP_Init(peripheral,ACMP_SLICE_NUMBER, &g_acmp_config);

  XMC_ACMP_SetInput(peripheral,ACMP_SLICE_NUMBER,XMC_ACMP_INP_SOURCE_STANDARD_PORT);

  status = DAVE_Init();     /* Initialization of DAVE Apps  */
```

Digital IO configuration [1/3]

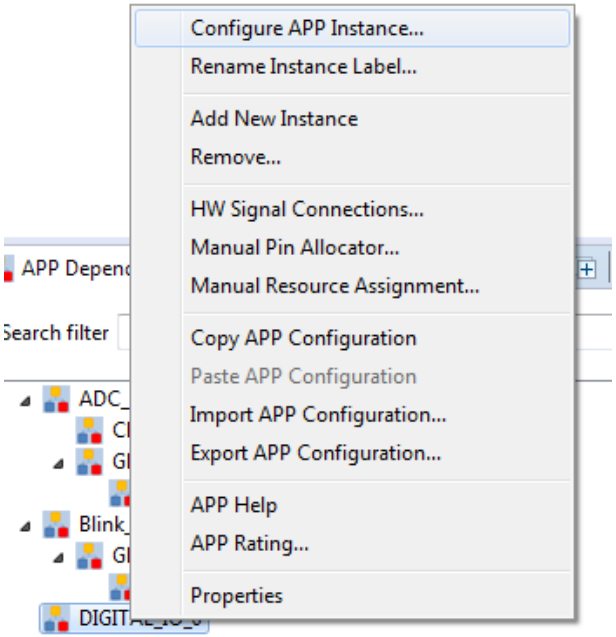
- The next step for the session is to configure a Digital IO pin as output of ACMP2 (P0.5) by the use of relative APP of DAVE
  - Click on “Add new APP” on bottom left menu
  - Type ”digital” on Search filter field
  - Double click on DIGITAL_IO



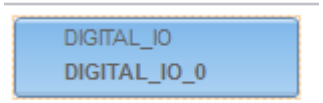


Digital IO configuration [2/3]

- Perform right-click on App name in the App Dependency Tree view
- Select Configure APP Instance

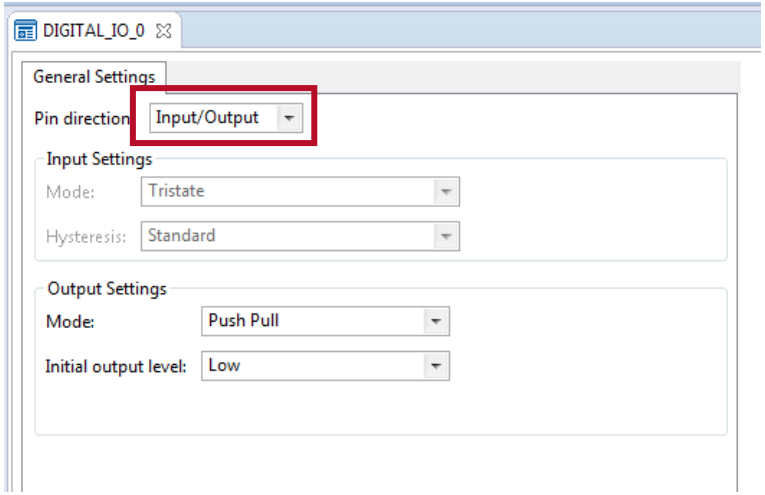


- Second Possibility:
- Double-click on App Name in App Dependency View

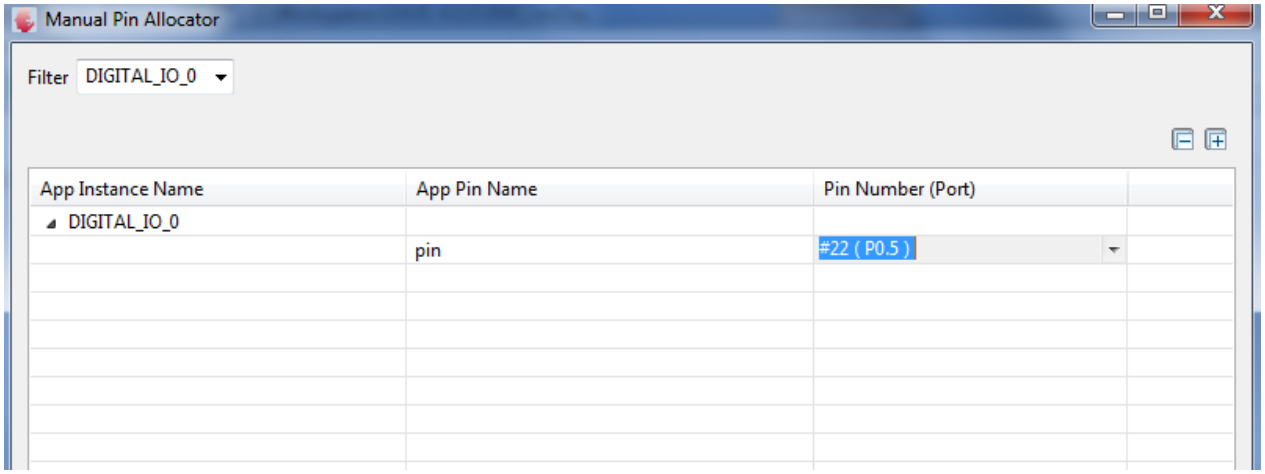


Digital IO configuration [3/3]

- Set pin as Input/Output



- Allocate P0.5 as output by “Manual Pin Allocator” menu



DAVE: Generation code

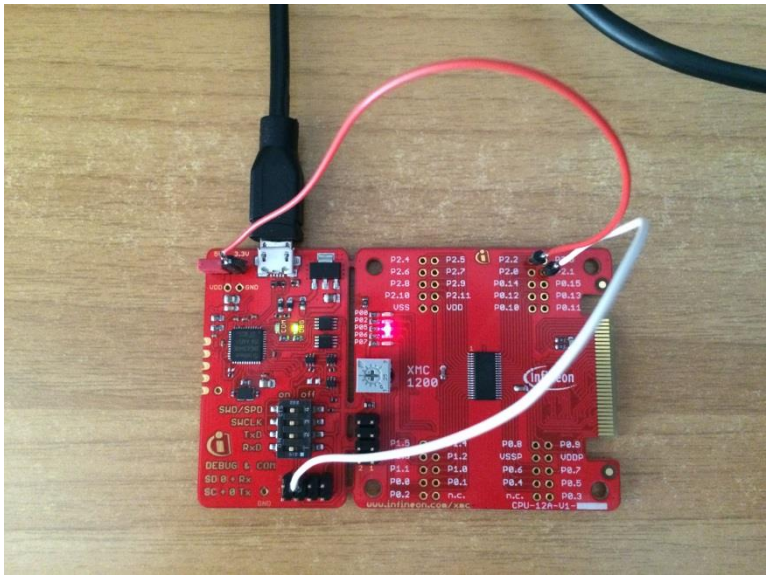
- One touch code generation
  1. Click  in the tool panel
  2. Generated code can be found under C/C++ Projects window, DAVE → Generated

- Start Compiler tools to build the project
  - Click  in the tool panel

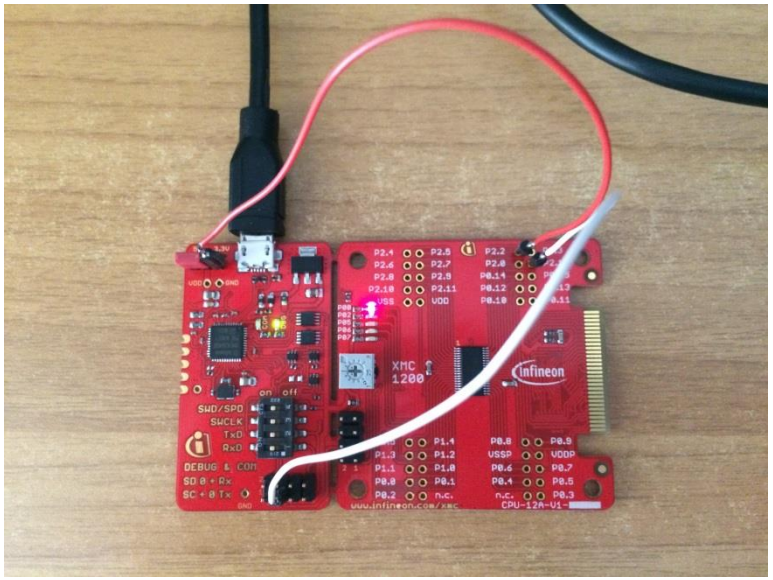
- Ensure that Compiler finished building in Console window

ACMP: Hardware connections

- Case 1:
  ACMP2.INN 3.3V (P2.2)
  ACMP2.INP 0V (P2.1)



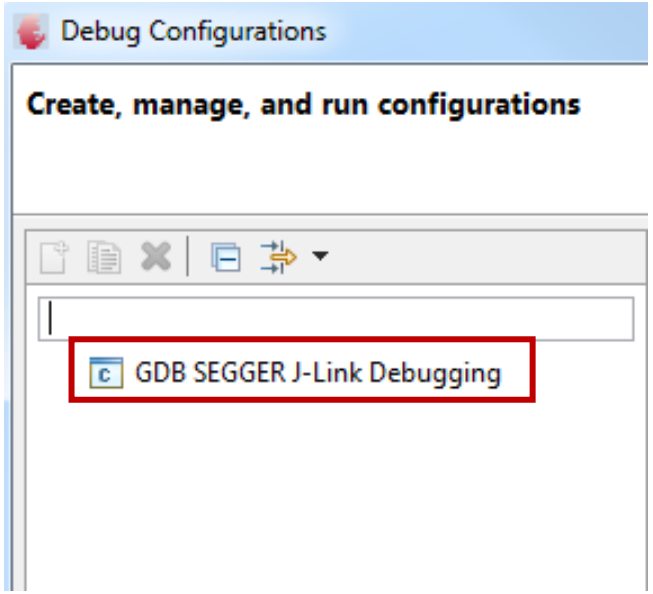
- Case 2:
  ACMP2.INN 3.3V (P2.2)
  ACMP2.INP 5V (P2.1)



DAVE: Flash and Debug [1/2]

- Start Debug Session
  - Click  in the tool panel

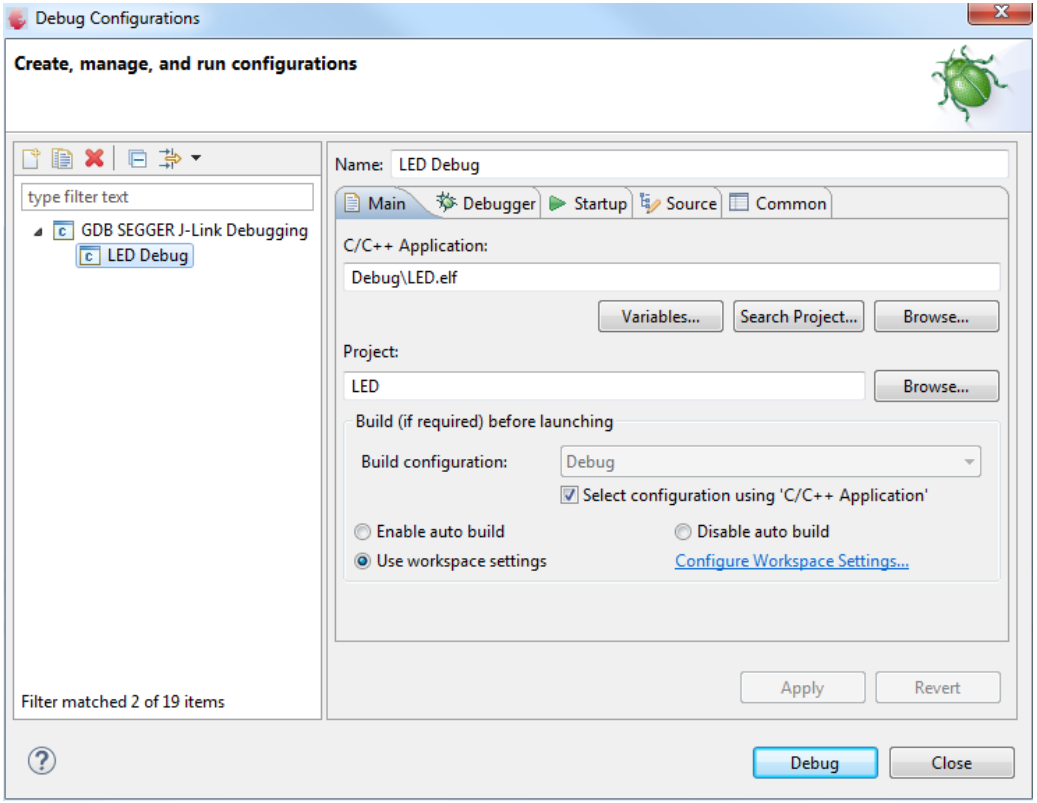
- Create a new Debug Configuration
  - Double-click "GDB SEGGER J-Link Debugging"



Segger J-link Driver software 4.96h or above needs to be installed

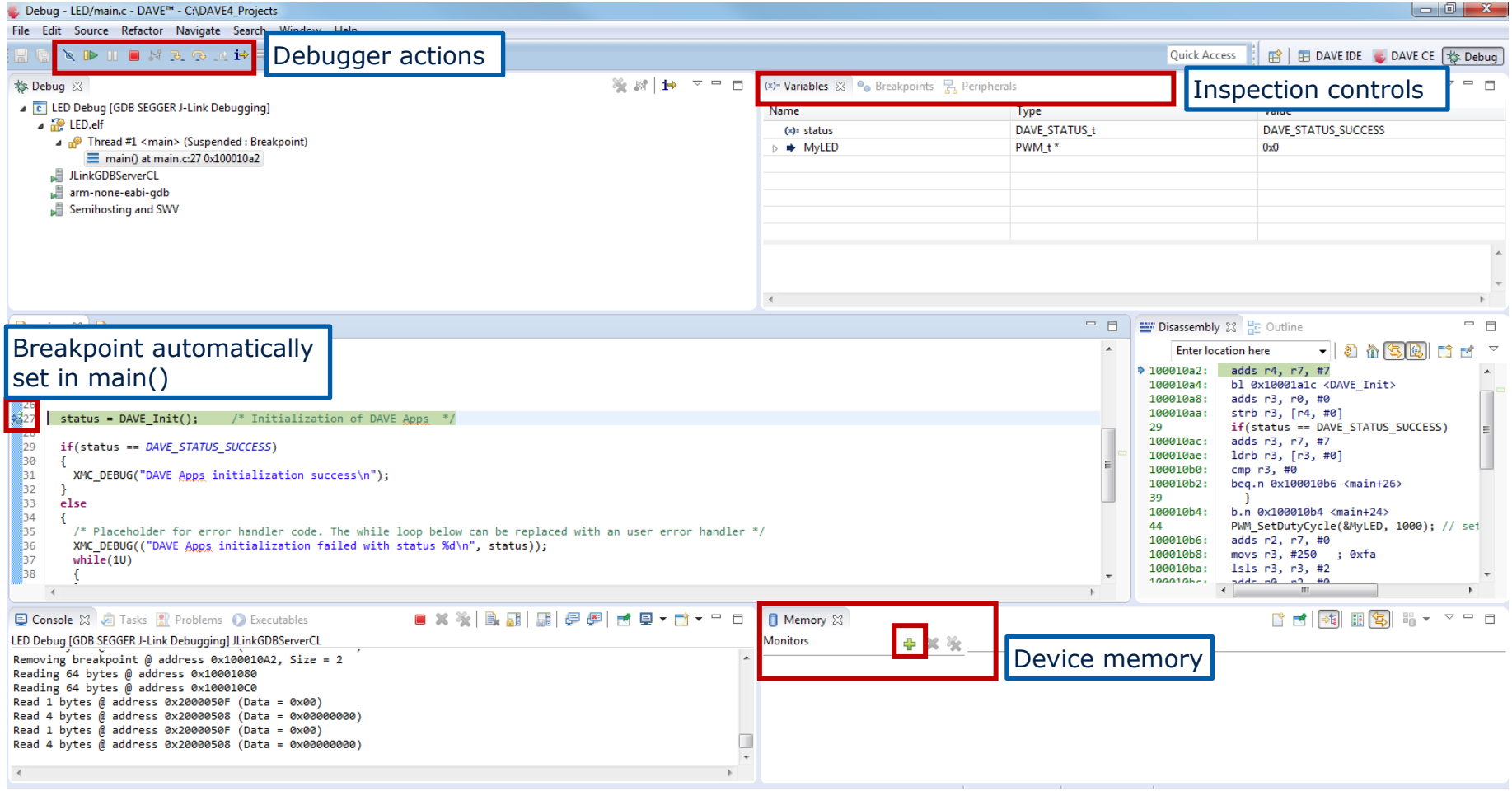
DAVE: Flash and Debug [2/2]

- Click “Debug”
- The flashing process is started and DAVE automatically switches to Debug Perspective

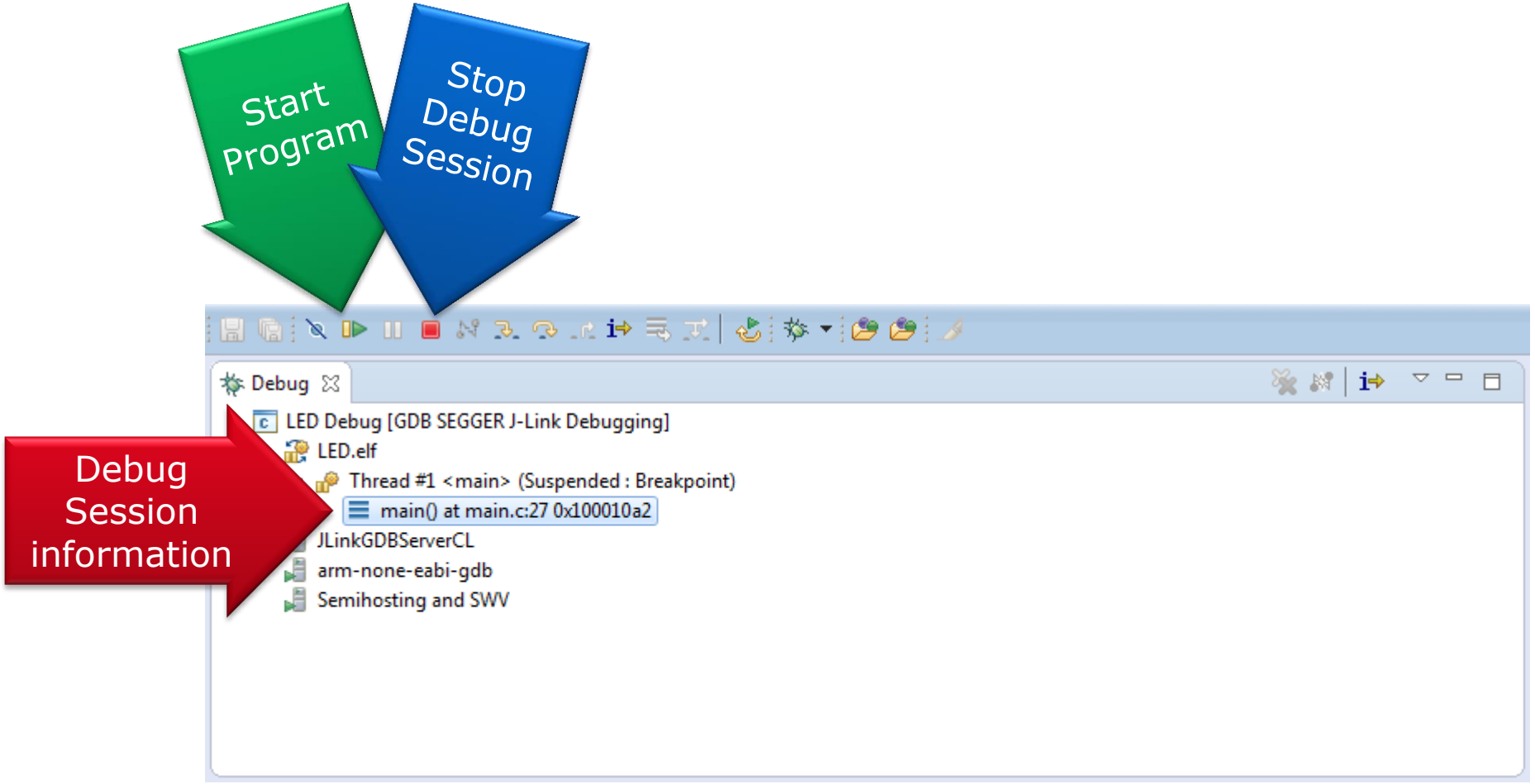


- Hint: To switch to Project Workspace Perspective, click DAVE CE at upper right corner of window

The Debug Perspective [1/4]
Debug Workspace

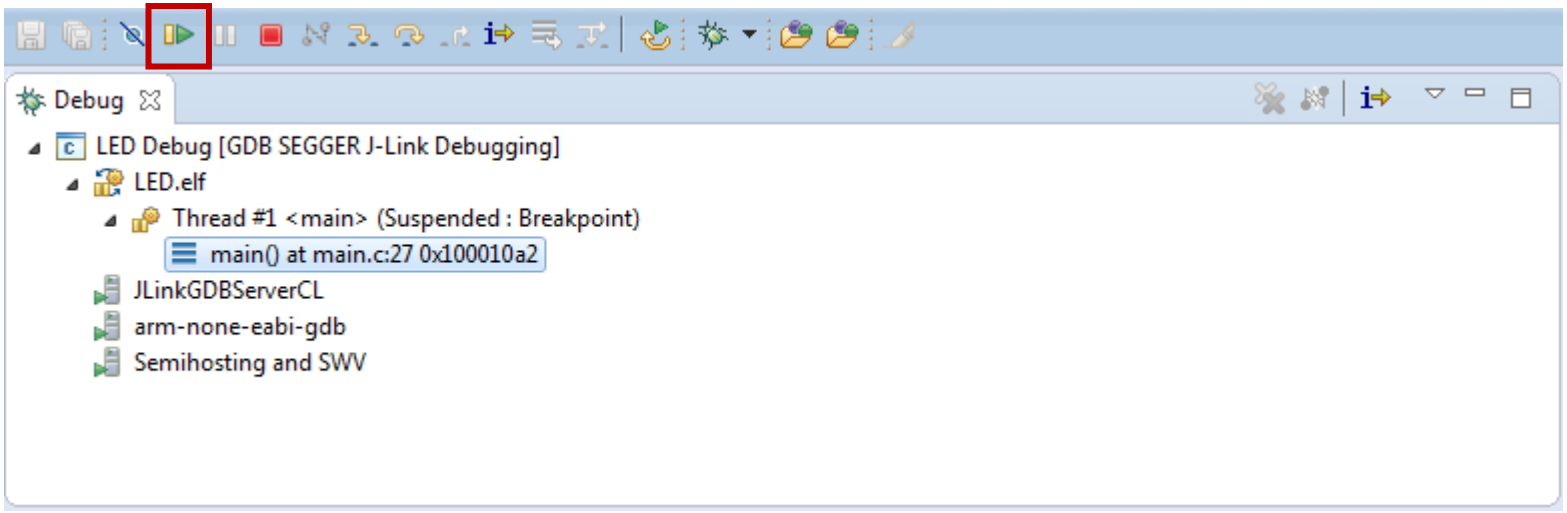


The Debug Perspective [2/4]
Debug Window



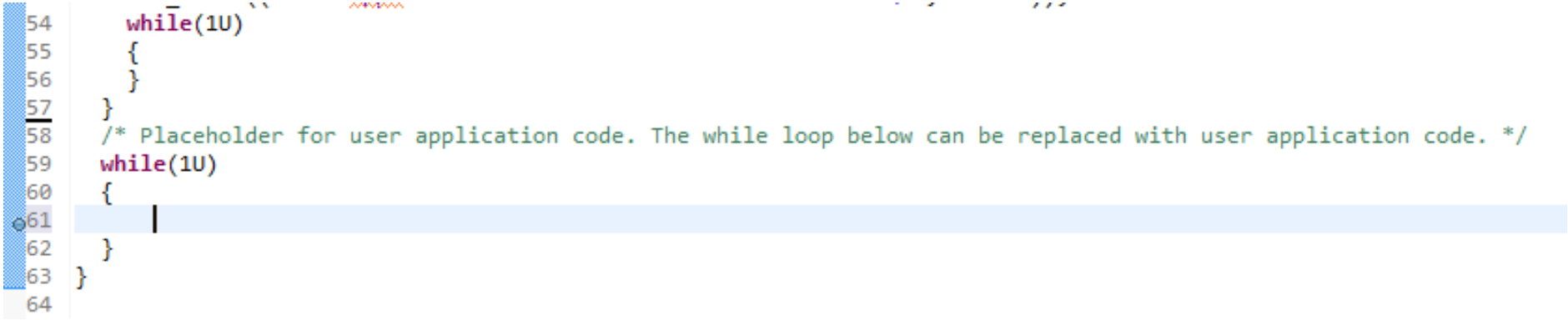
The Debug Perspective [3/4]
Start Program

- Click on the Resume button to start code execution



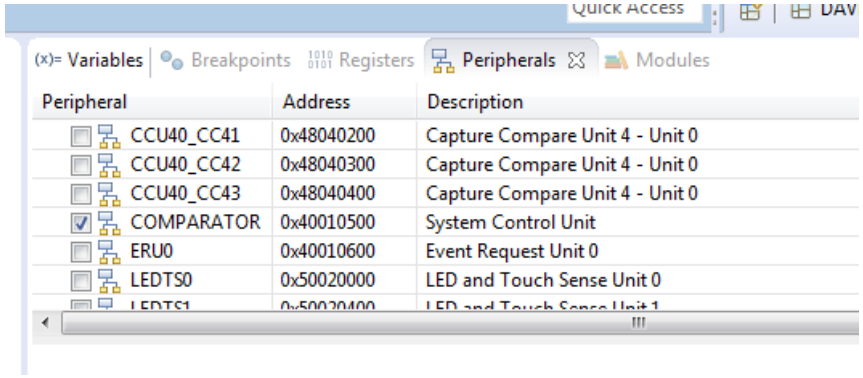
The Debug Perspective [4/4]
Breakpoints

- To place a breakpoint, double-click on the blue bar at the line of code

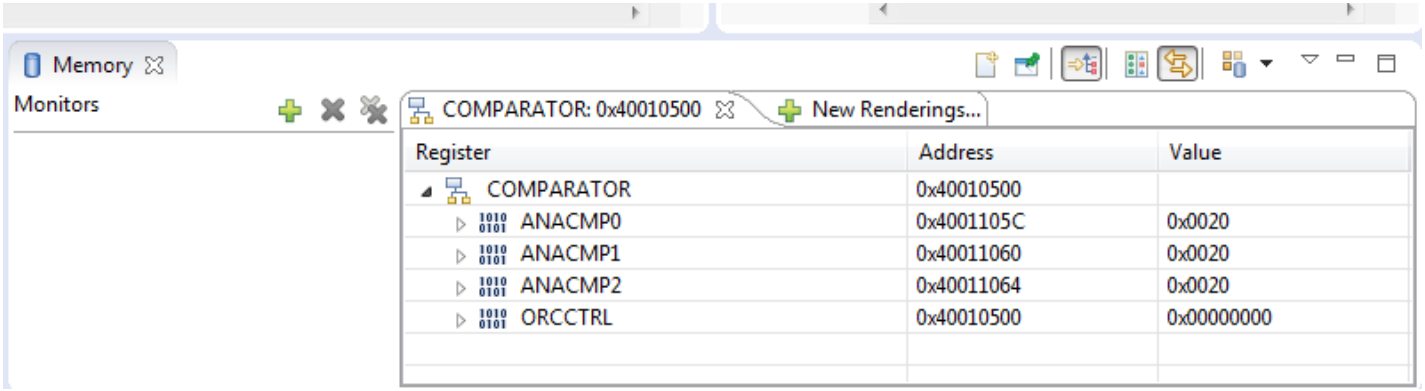


Debug Session

- Select COMPARATOR in the Peripherals menu of Inspection controls part of Debug Perspective

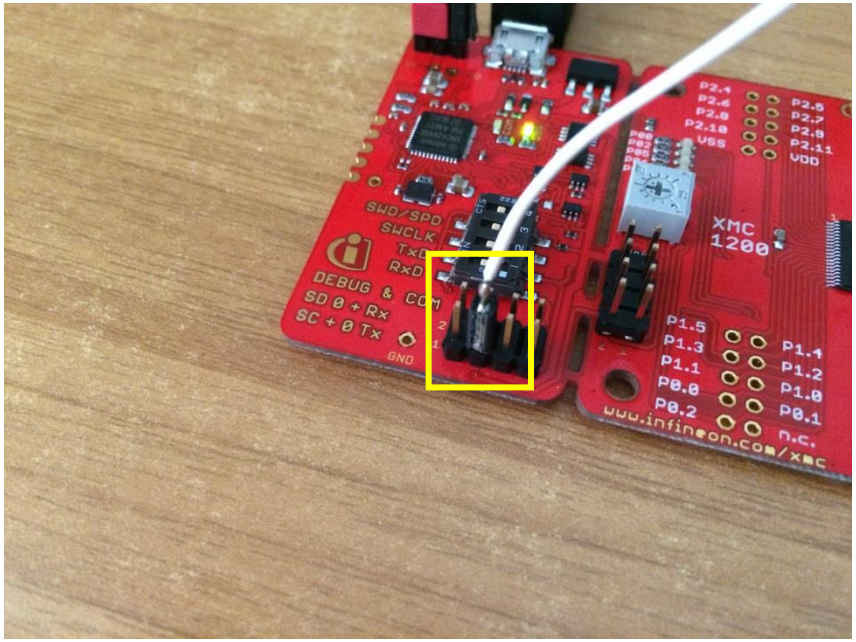


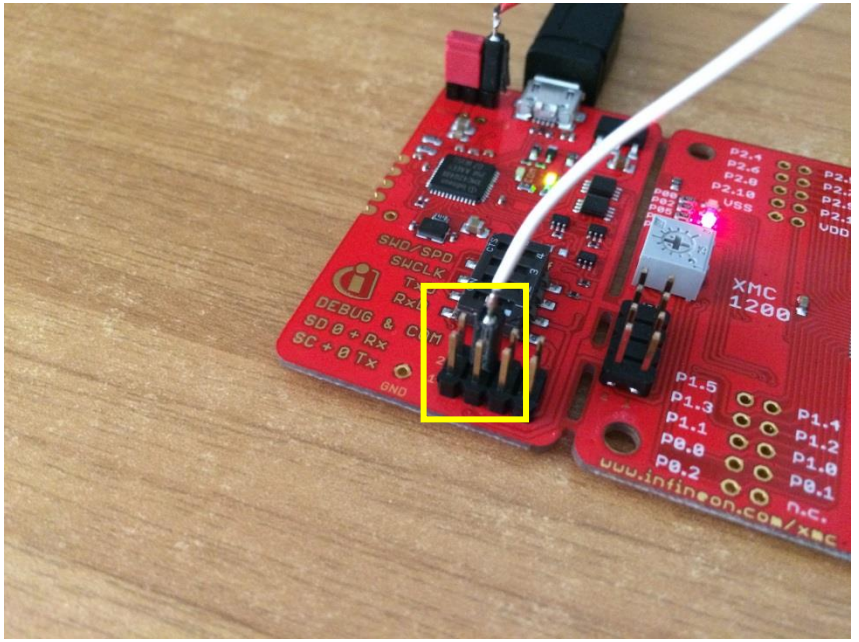
- In the Memory window will appear the registers of chosen peripheral



Debug session

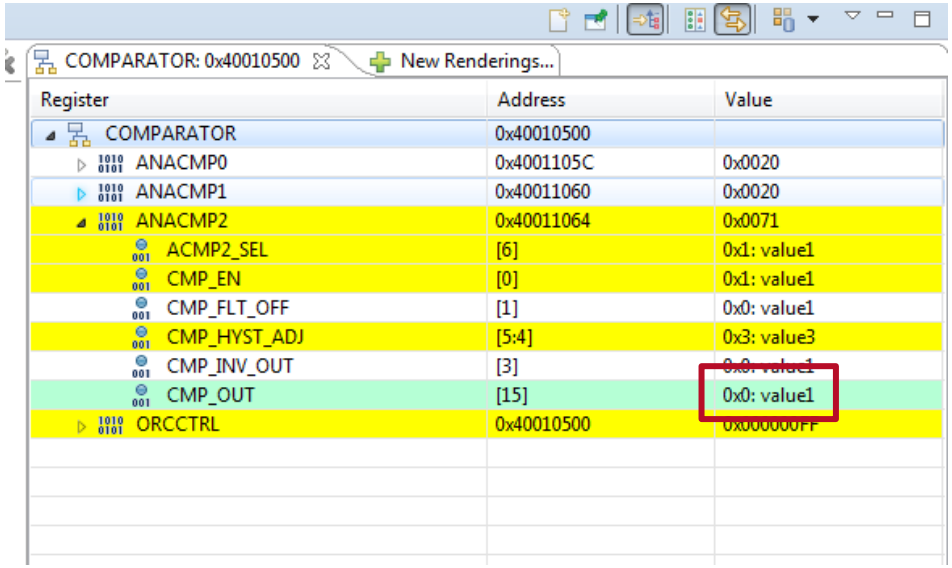
- Switching input P2.1 from “+” to “0” it is possible to see the change of CMP_OUT register



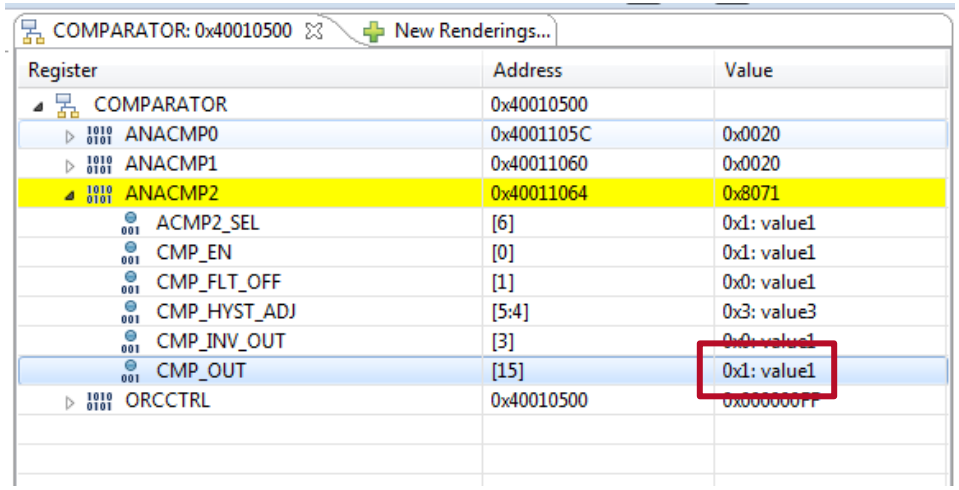


Debug Session

- Outputs of CMP_OUT
  - P2.1 connected to “+”



  - P2.1 connected to “0”



Debug Session

- The P0.5 doesn’t change its status. Why?
  - It is not set as ACMP2. OUT!!

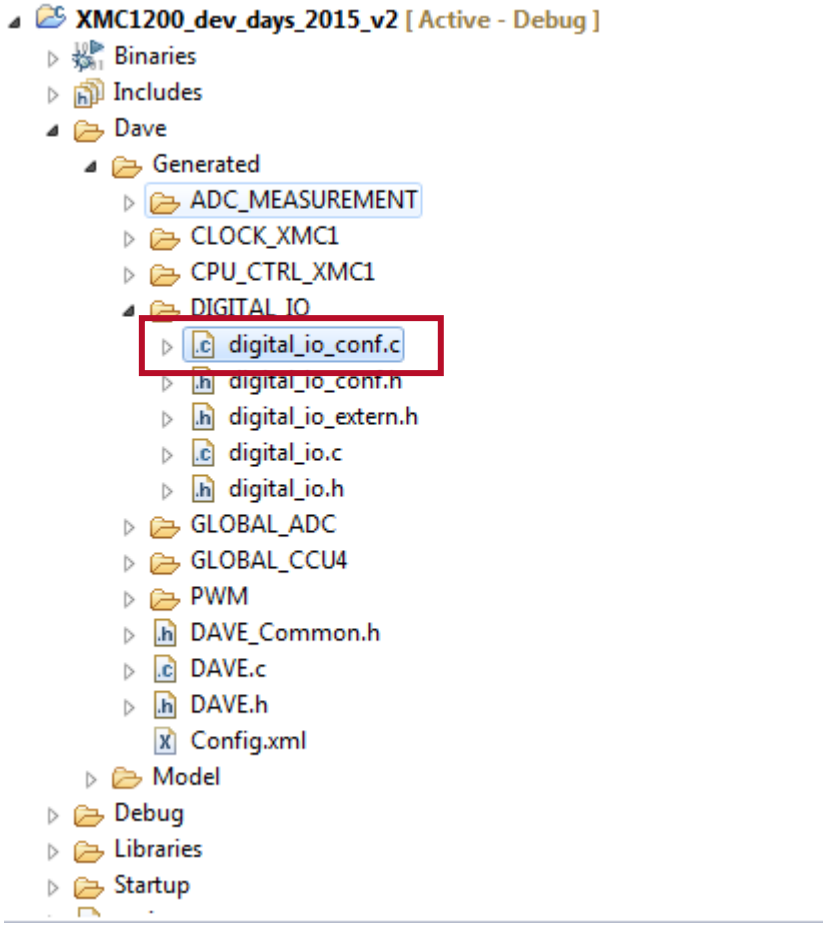
- Each pin has many functions, called Alternate Functions
- In every Reference Manual, it is present a table which describe this characteristics

Alternate Functions

Function	Outputs							Inputs							
	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	Input	Input	Input	Input	Input	Input	Input	Input
P0.0	ERU0.PDOUT0	LEDTS0.LINE7	ERU0.GOUT0	CCU40.OUT0		USIC0_CH0.SELO0	USIC0_CH1.SELO0	BCCU0.TRAPINB	CCU40.IN0C			USIC0_CH0.DX2A	USIC0_CH1.DX2A		
P0.1	ERU0.PDOUT1	LEDTS0.LINE6	ERU0.GOUT1	CCU40.OUT1		BCCU0.OUT8	SCU.VDROP		CCU40.IN1C						
P0.2	ERU0.PDOUT2	LEDTS0.LINE5	ERU0.GOUT2	CCU40.OUT2		VADC0.EMUX02			CCU40.IN2C						
P0.3	ERU0.PDOUT3	LEDTS0.LINE4	ERU0.GOUT3	CCU40.OUT3		VADC0.EMUX01			CCU40.IN3C						
P0.4	BCCU0.OUT0	LEDTS0.LINE3	LEDTS0.COL3	CCU40.OUT1		VADC0.EMUX00	WWDT.SERVICE_OUT								
P0.5	BCCU0.OUT1	LEDTS0.LINE2	LEDTS0.COL2	CCU40.OUT0		ACMP2.OUT									
P0.6	BCCU0.OUT2	LEDTS0.LINE1	LEDTS0.COL1	CCU40.OUT0		USIC0_CH1.MCLKOUT	USIC0_CH1.DOUT0		CCU40.IN0B			USIC0_CH1.DX0C			
P0.7	BCCU0.OUT3	LEDTS0.LINE0	LEDTS0.COL0	CCU40.OUT1		USIC0_CH0.SCLKOUT	USIC0_CH1.DOUT0		CCU40.IN1B			USIC0_CH0.DX1C	USIC0_CH1.DX0D	USIC0_CH1.DX1C	
P0.8	BCCU0.OUT4	LEDTS1.LINE0	LEDTS0.COLA	CCU40.OUT2		USIC0_CH0.SCLKOUT	USIC0_CH1.SCLKOUT		CCU40.IN2B			USIC0_CH0.DX1B	USIC0_CH1.DX1B		
P0.9	BCCU0.OUT5	LEDTS1.LINE1	LEDTS0.COL6	CCU40.OUT3		USIC0_CH0.SELO0	USIC0_CH1.SELO0		CCU40.IN3B			USIC0_CH0.DX2B	USIC0_CH1.DX2B		
P0.10	BCCU0.OUT6	LEDTS1.LINE2	LEDTS0.COL5	ACMP0.OUT		USIC0_CH0.SELO1	USIC0_CH1.SELO1					USIC0_CH0.DX2C	USIC0_CH1.DX2C		
P0.11	BCCU0.OUT7	LEDTS1.LINE3	LEDTS0.COL4	USIC0_CH0.MCLKOUT		USIC0_CH0.SELO2	USIC0_CH1.SELO2					USIC0_CH0.DX2D	USIC0_CH1.DX2D		
P0.12	BCCU0.OUT6	LEDTS1.LINE4	LEDTS0.COL3	LEDTS1.COL3		USIC0_CH0.SELO3		BCCU0.TRAPINA	CCU40.IN0A	CCU40.IN1A	CCU40.IN2A	CCU40.IN3A	USIC0_CH0.DX2E		
P0.13	WWDT.SERVICE_OUT	LEDTS1.LINE5	LEDTS0.COL2	LEDTS1.COL2		USIC0_CH0.SELO4						USIC0_CH0.DX2F			
P0.14	BCCU0.OUT7	LEDTS1.LINE6	LEDTS0.COL1	LEDTS1.COL1		USIC0_CH0.DOUT0	USIC0_CH0.SCLKOUT					USIC0_CH0.DX0A	USIC0_CH0.DX1A		

Alternate Function: how to configure

- Step 1: change perspective to DAVE CE and open the digital_io_conf.c file with double click



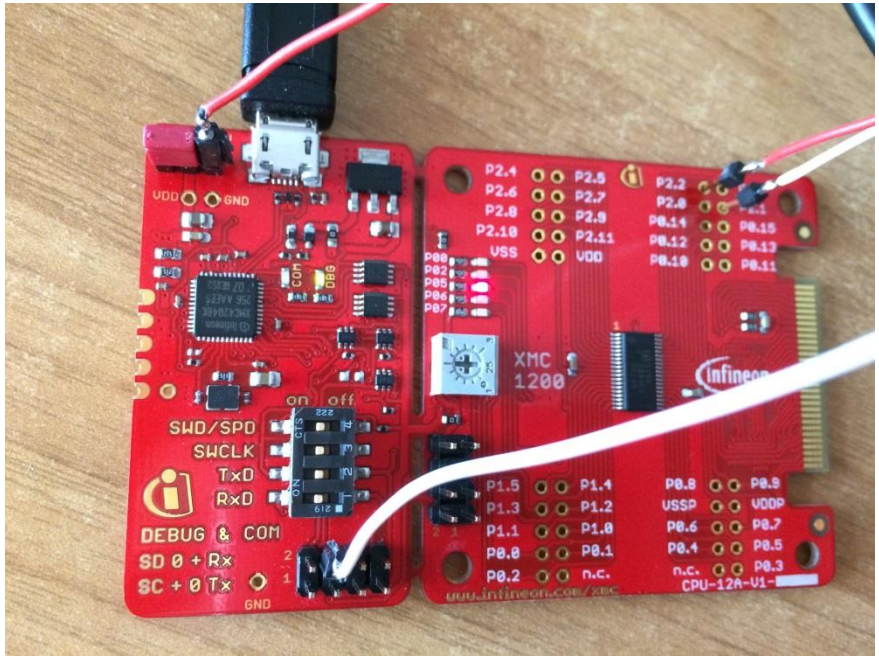
Alternate Function: how to configure

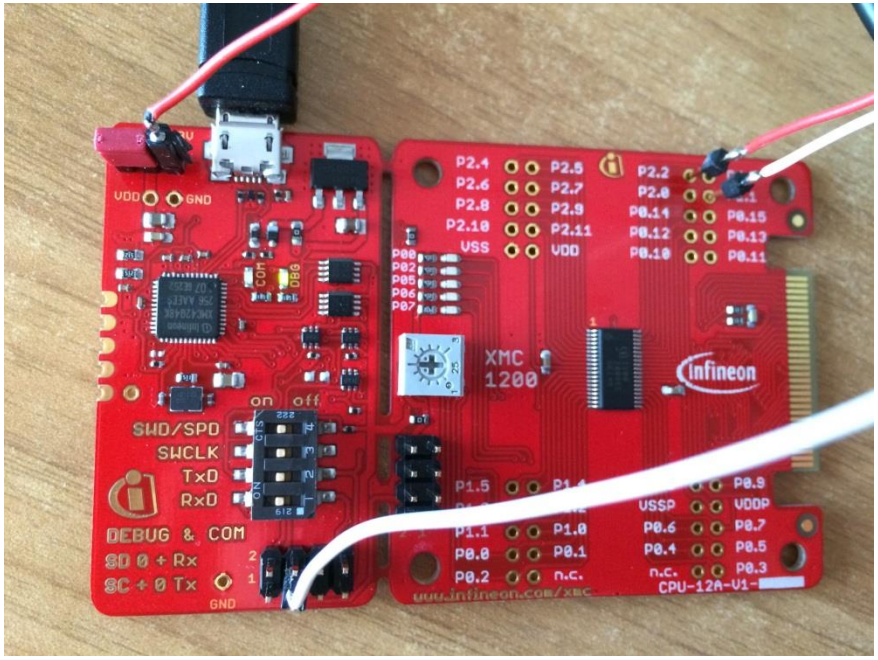
- Step 2: Change the conf mode of the DIGITAL_IO from ***XMC_GPIO_MODE_OUTPUT_PUSH_PULL*** *to* ***XMC_GPIO_MODE_OUTPUT_PUSH_PULL_ALT6***

```
78
79  XMC_GPIO_CONFIG_t DIGITAL_IO_0_config  =
80  {
81    .mode = XMC_GPIO_MODE_OUTPUT_PUSH_PULL_ALT6,
82    .output_level = XMC_GPIO_OUTPUT_LEVEL_LOW,
83
84  };
85
86  DIGITAL_IO_t DIGITAL_IO_0 =
87  {
88    .gpio_port = XMC_GPIO_PORT0,
89    .gpio_pin = 5U,
90    .gpio_config = &DIGITAL_IO_0_config,
91  };
92
93
```

Alternate Function: how to configure

- Step3: Re-Compile the code **without any Re-generation of code**
- Step4: Flash and Debug
- Step5: P0.5 works accordingly with the output of ANACMP peripheral



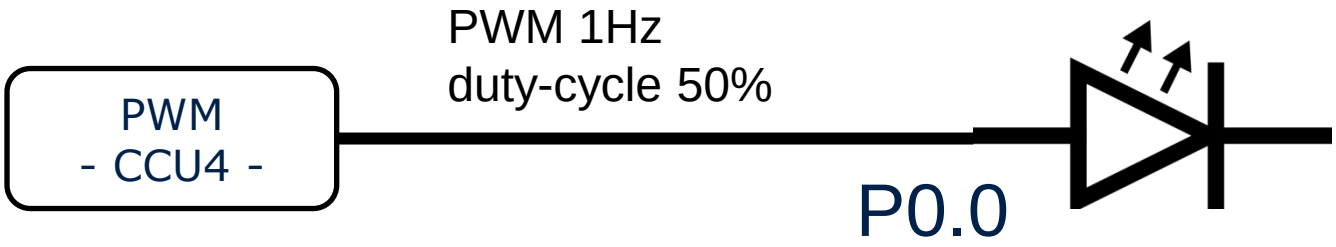


Agenda

- LLD of XMCLib: use of ACMP
- Blink an LED with PWM APP
- Use ADC APP to update PWM dutycycle

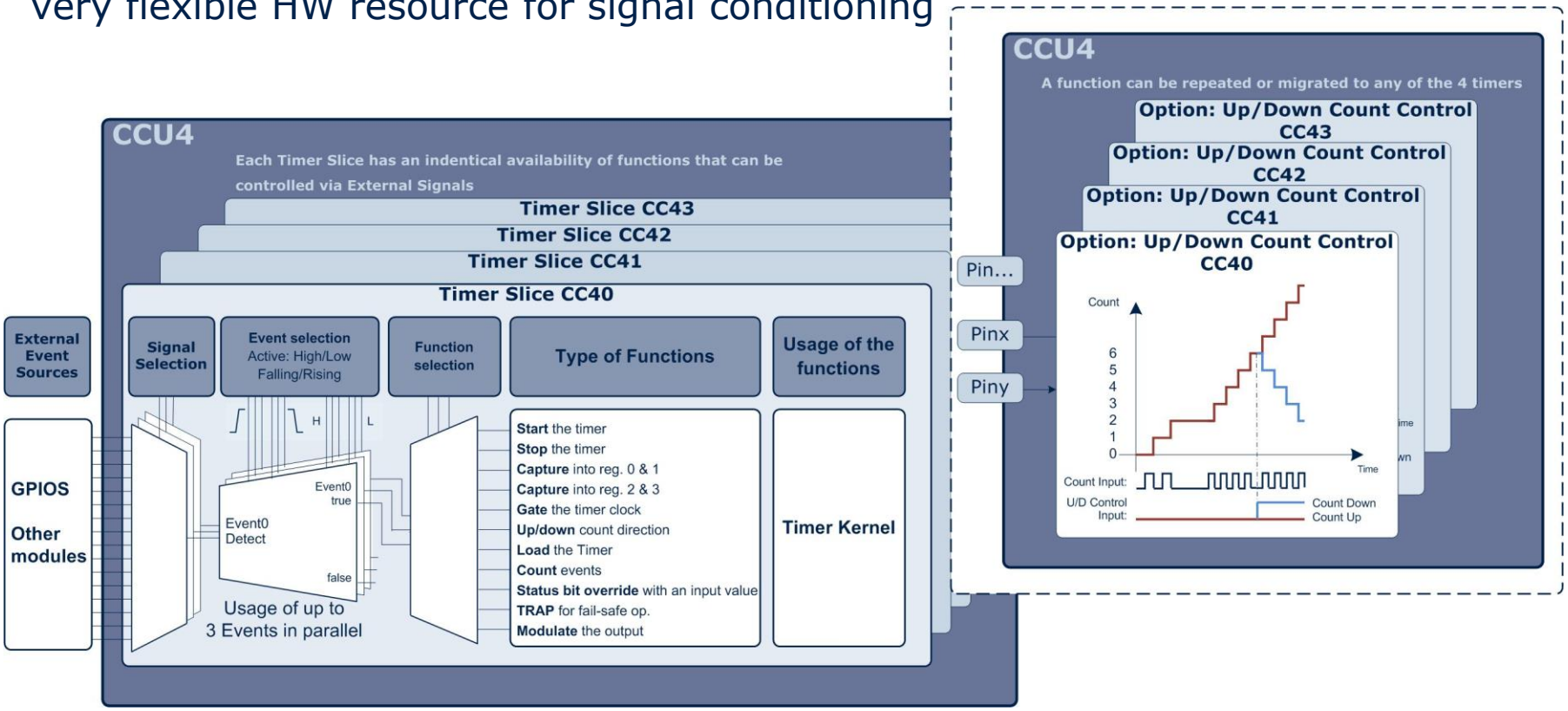
PWM to Blink an LED

- Creation of a PWM signal that control a LED



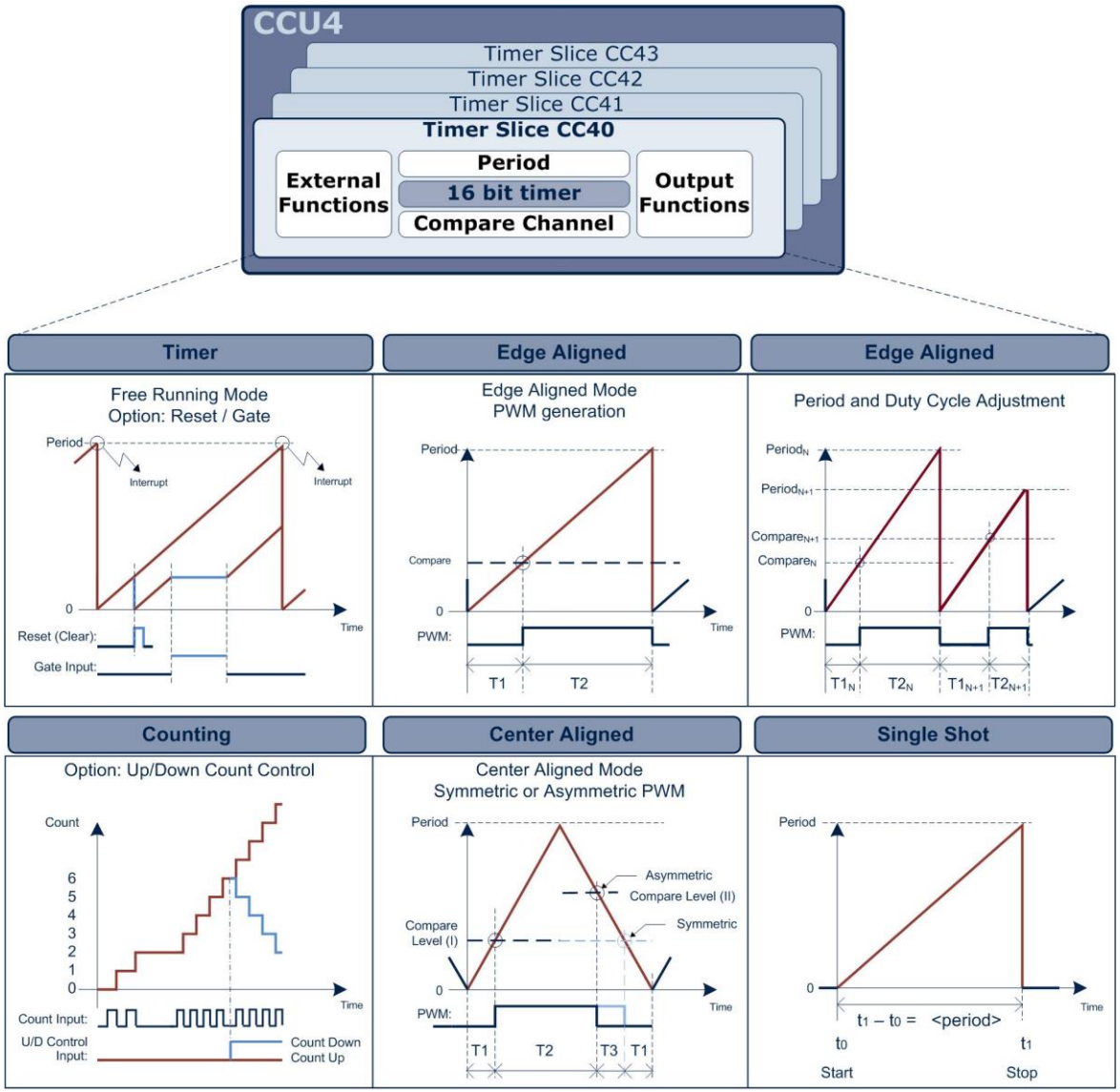
CCU4
Modular Timer Approach

- Equal structure and same availability of features for each of the 4 Timer Slices
- Functions controllable via external signals do not depend on the selected signal
- Portability of code is not dependent on the used Timer Slice
- High amount of configurable external functions (11), make each Timer Slice a very flexible HW resource for signal conditioning



CCU4
Flexible PWM generation

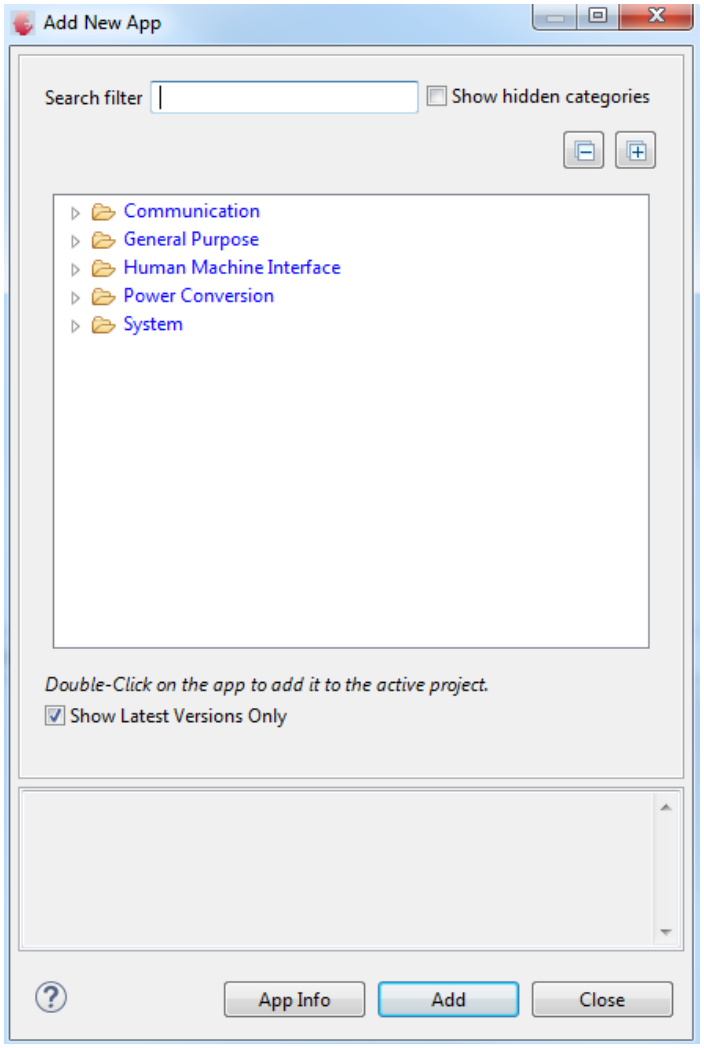
- Each Timer Slice of the CCU4 can operate in center aligned or edge aligned mode
- Additional operation modes like single shot, counting or dithering modes are also available
- Update of the Duty Cycle and Period can be done on-the-fly to accommodate different operation requirements
- Additional external controllable functions give another degree of PWM manipulation (e.g. timer gate, timer load, timer clear, etc)



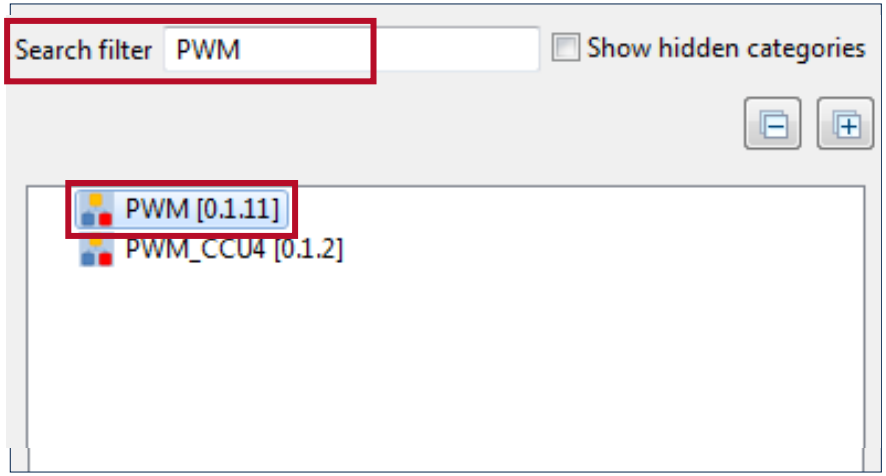
Learning Outcome

- Learn the basic principles of DAVE™:
  - Required XMC kit
  - Create DAVE™ Project
  - GUI based DAVE™ APPs configuration
  - Graphical pin mapping
  - One touch code generation
  - Download and debug code
  - DAVE™ updates
  - Expert support

Add DAVE™ APP from Code Repository

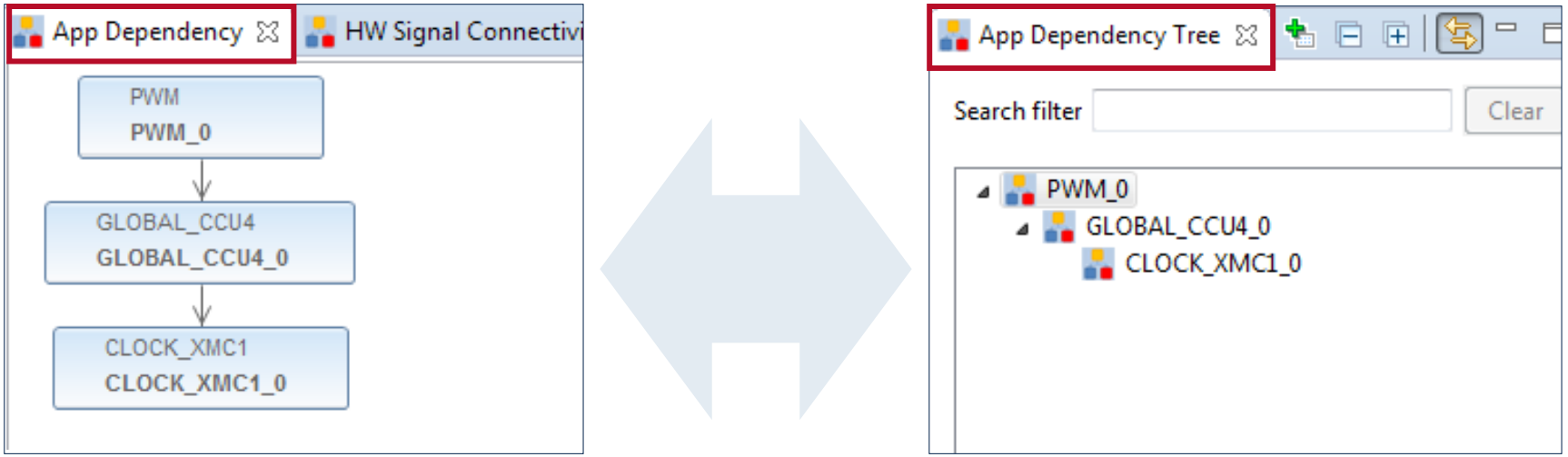


- Add DAVE™ APP to project
  1. Click  in Tool Panel, or
  2. DAVE → APP New APP
  3. Type "PWM" in the search filter field, and double-click PWM APP

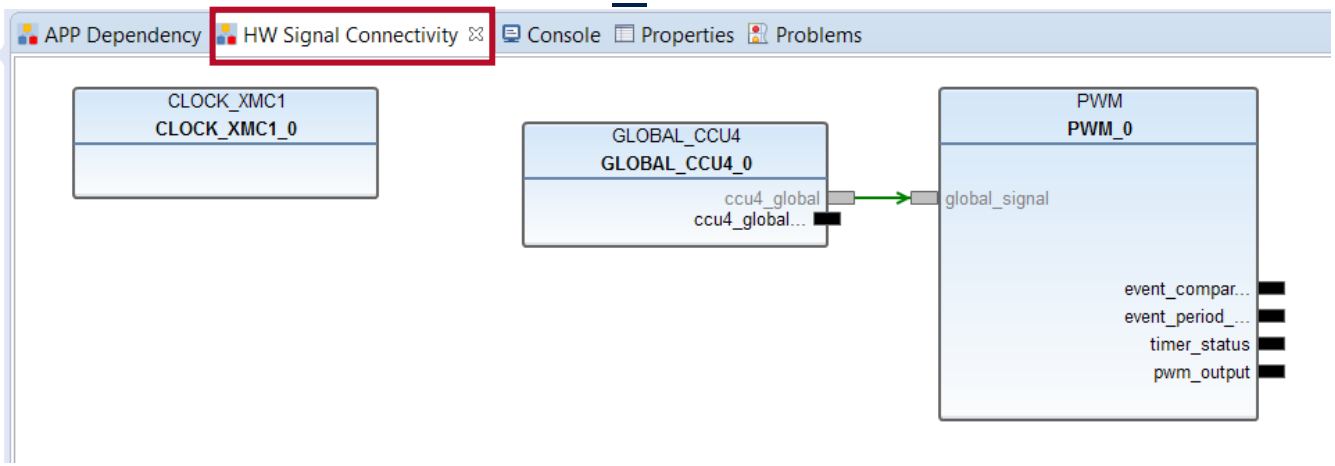


More Project Views

- All APPs included in the Project are displayed in different views:

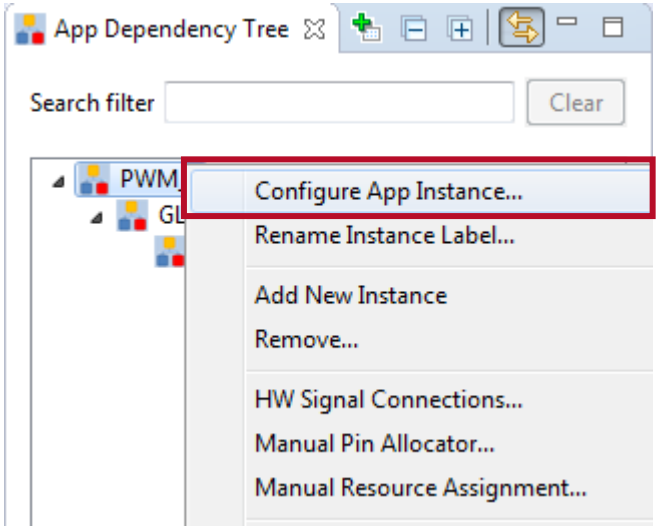


- The number behind "_" identifies the instance of an APP

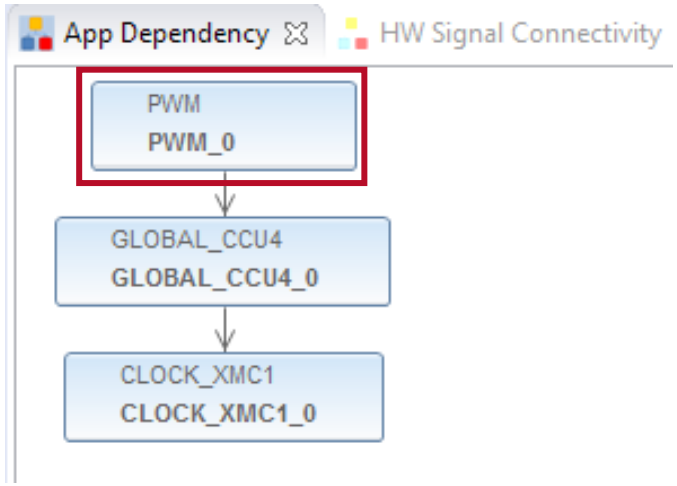


DAVE™ APPs Editor View

- Either
  1. Right-click APP name in the App Dependency Tree view
  2. Select Configure App Instance

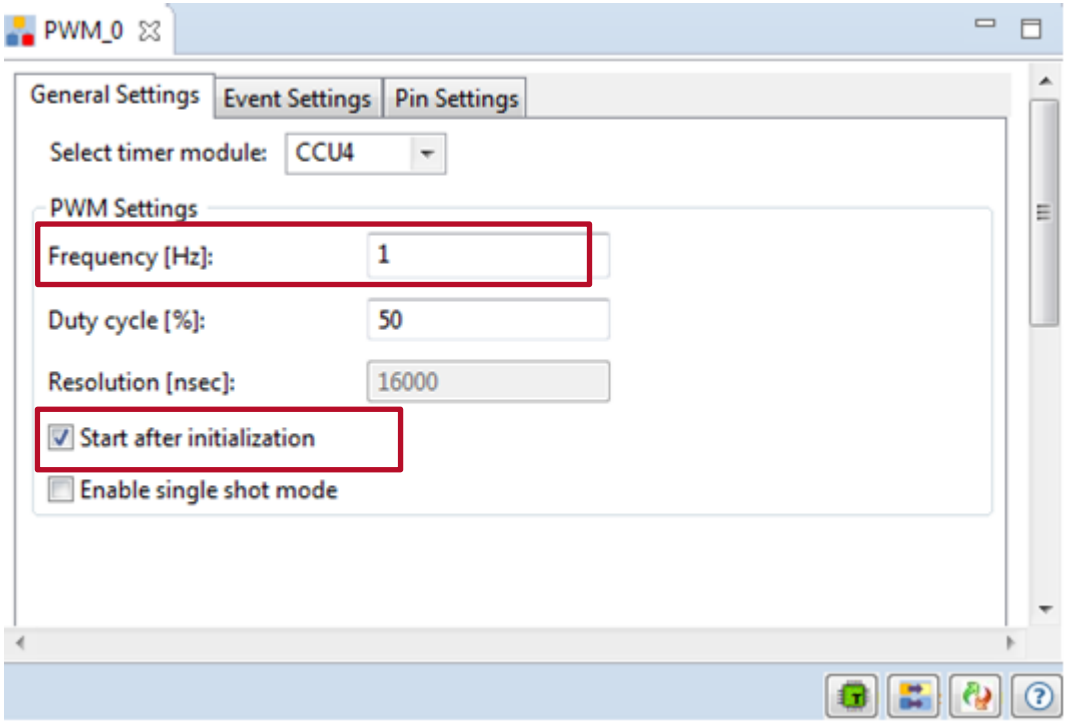


- Or
  - Double-click APP name in the App Dependency view



PWM APP Configuration

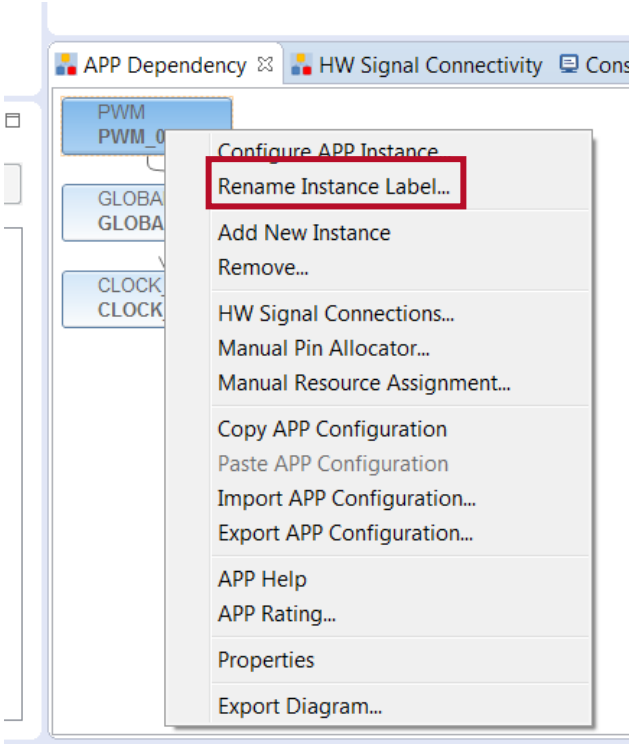
- Configure PWM APP via graphical user interface editor
  1. Set PWM Frequency to 1 Hz
  2. LED blinks every 0.5 second
  3. Check “Start Timer After Initialization”



Rename the Instance Label of the PWM APP

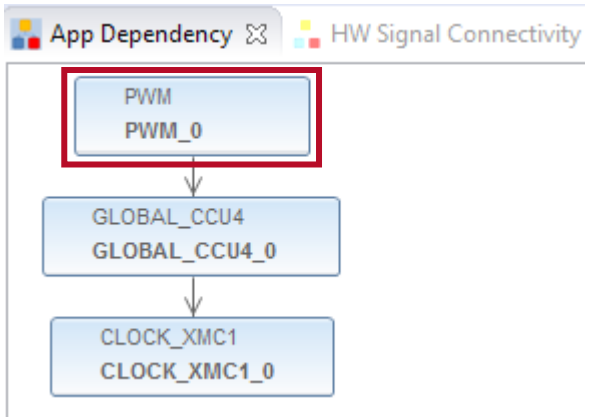
- Right click on the PWM APP
- Select
  Rename Instance Label…
- Type in: MyLED

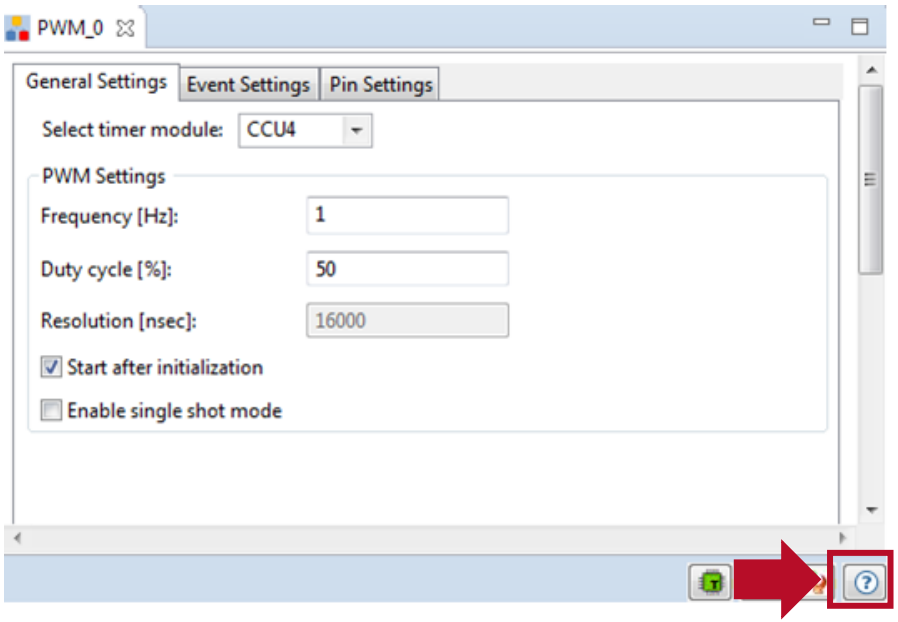
- Now “MyLED” can be used as handler in the APIs of the PWM APP to reference this instance

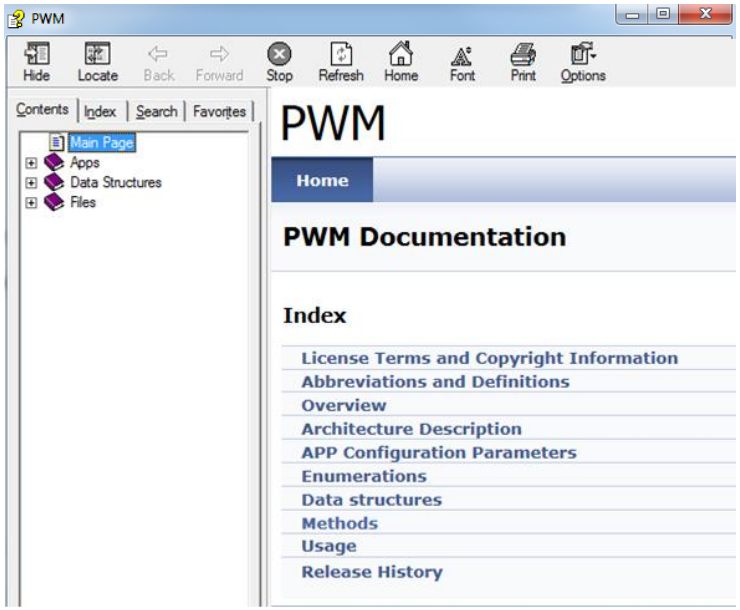


Hint: Additional Information about APPs

- Reference to DAVE™ APP information
  1. Double-click DAVE™ APP (e.g. PWM_0) in App Dependency View
  2. Click Help icon 

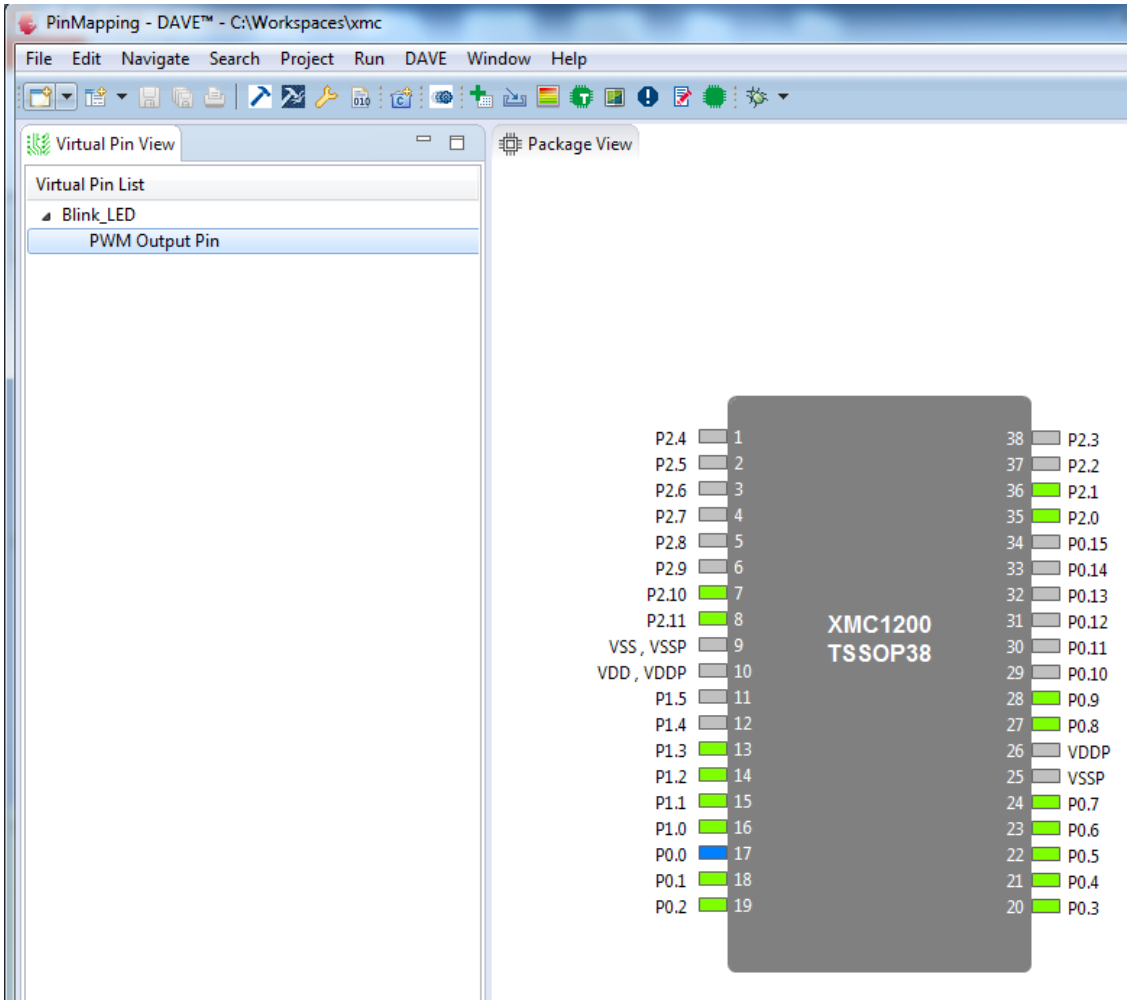






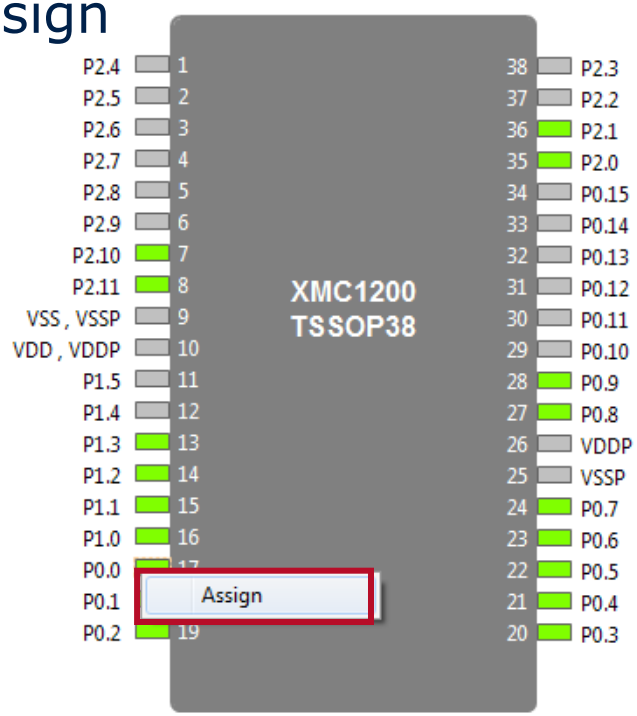
Pin Mapping for PWM App (1/2)

- Assign signal to pin with graphical pin mapping view
  1. Click  to open Pin Mapping Perspective
  2. Under Virtual Pin List, select PWM Output Pin
     - Green pin: All possible pins for selected signal
     - Blue pin: User assigned pin

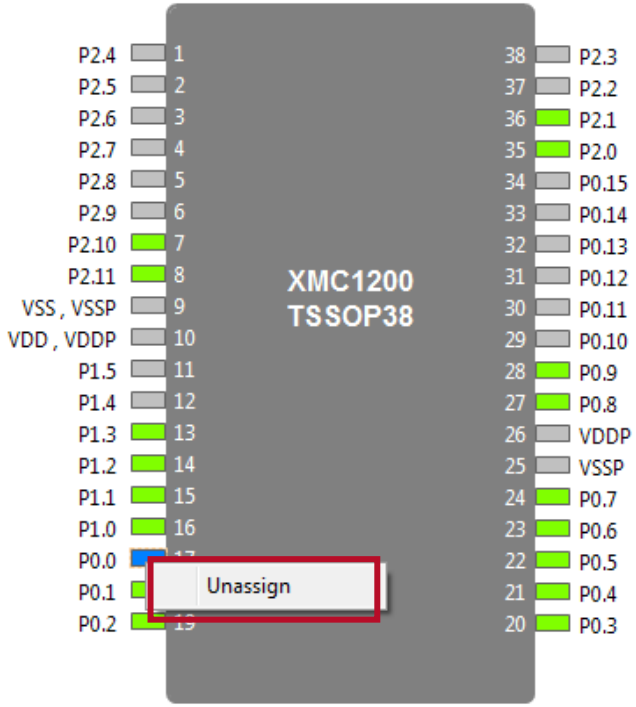


Pin Mapping for PWM App (2/2)

- To assign pin:
  - Right-click on a green pin → Assign

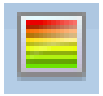


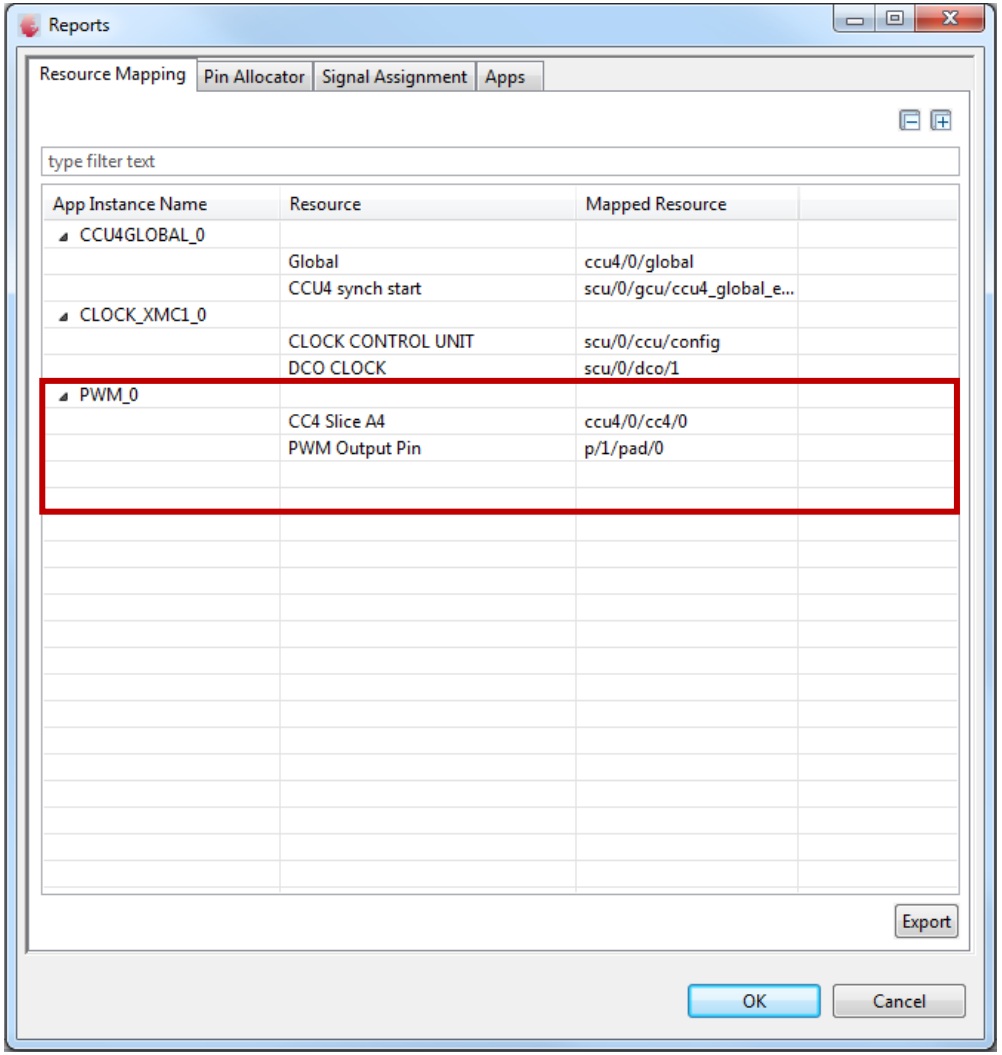
- To unassign pin:
  - Right-click on a blue pin → Unassign



- Assign PWM Output Pin to User LED1 at P0.0, #17
  - Right-click on pin 17 → Assign
  - In case you use a different board / device please select a pin that is connected to a LED

Hint: Check Correct Resource Mapping

- Check resource mapping
  - Click  to open Reports in DAVE CE perspective



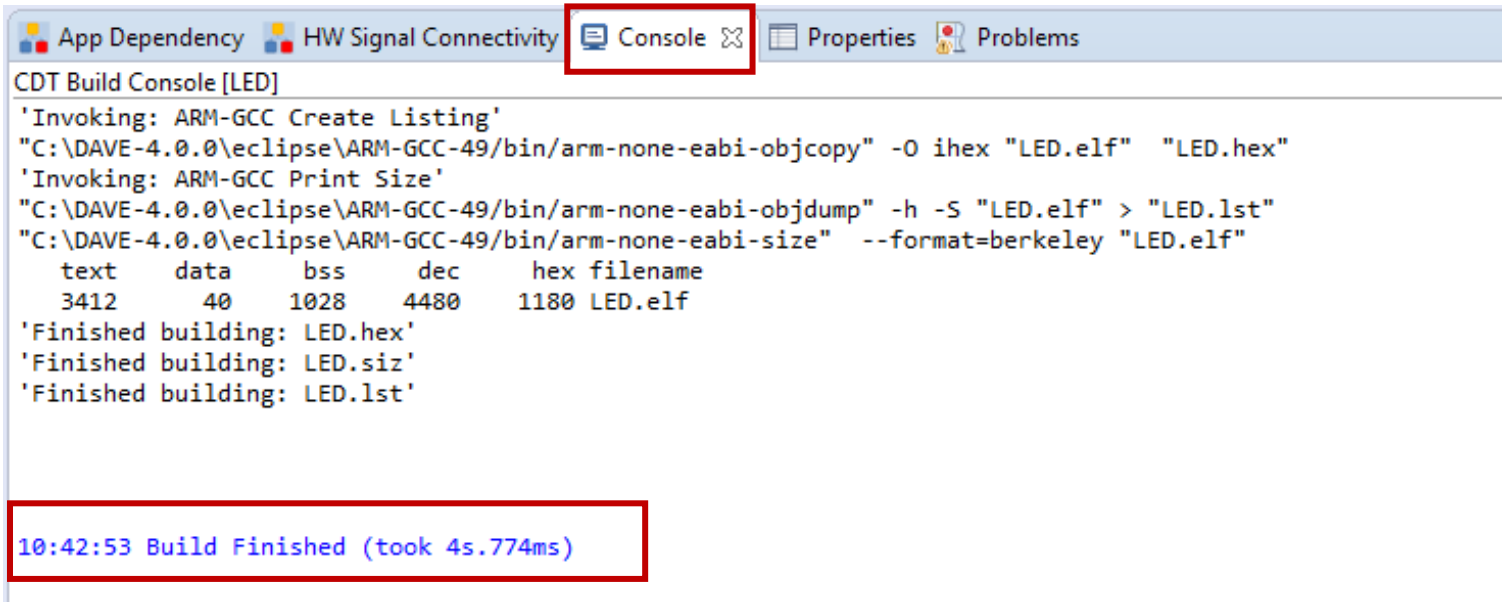
Generate Code and add a few Lines of Code to change the Duty Cycle of the PWM and Compile Code

- One touch code generation
  1. Click  in the tool panel
  2. Generated code can be found under C/C++ Projects window, DAVE → Generated

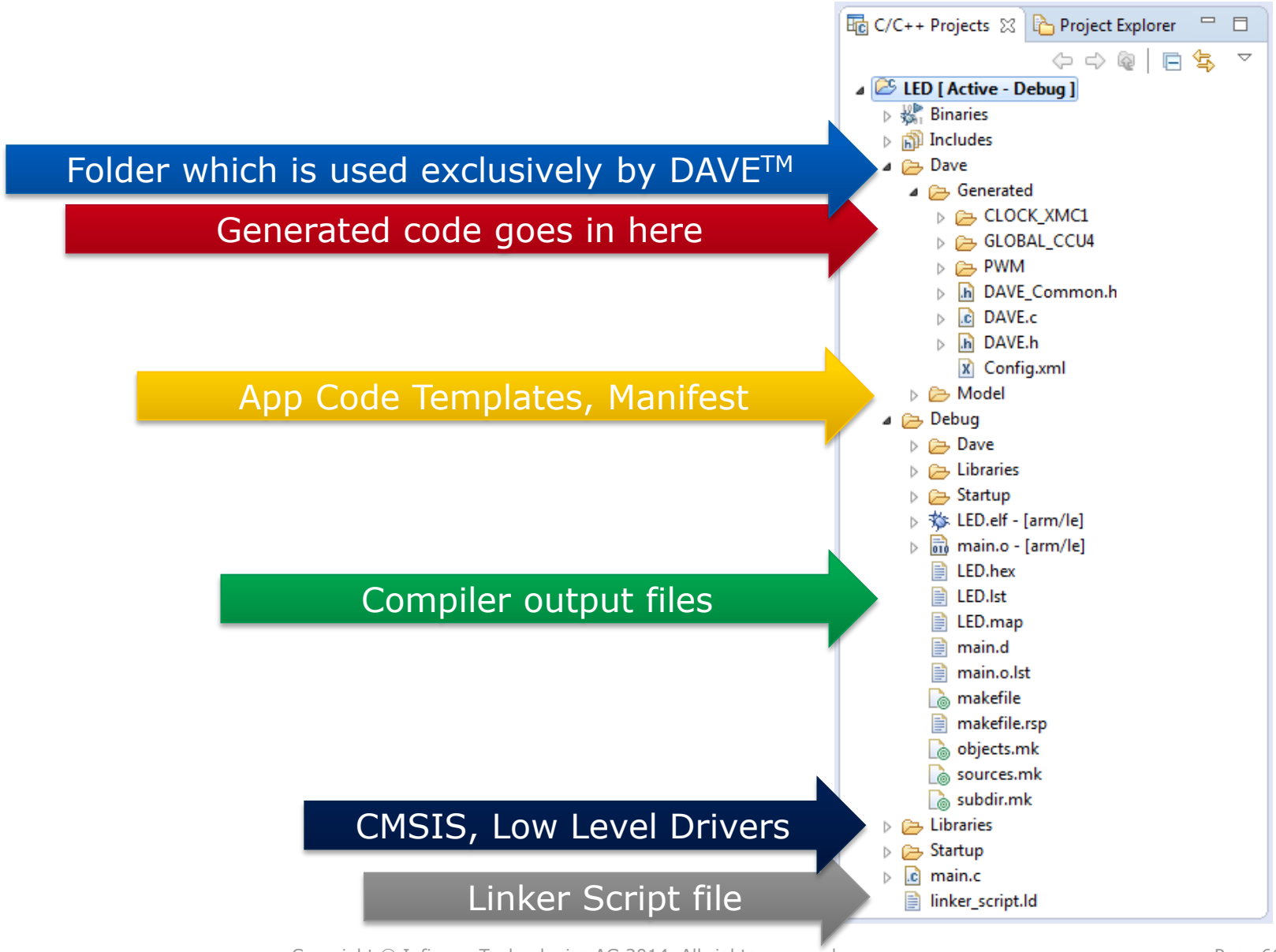
- Start Compiler tools to build the project
  - Click  in the tool panel

Check Compiler Results

- Ensure that Compiler finished building in Console window

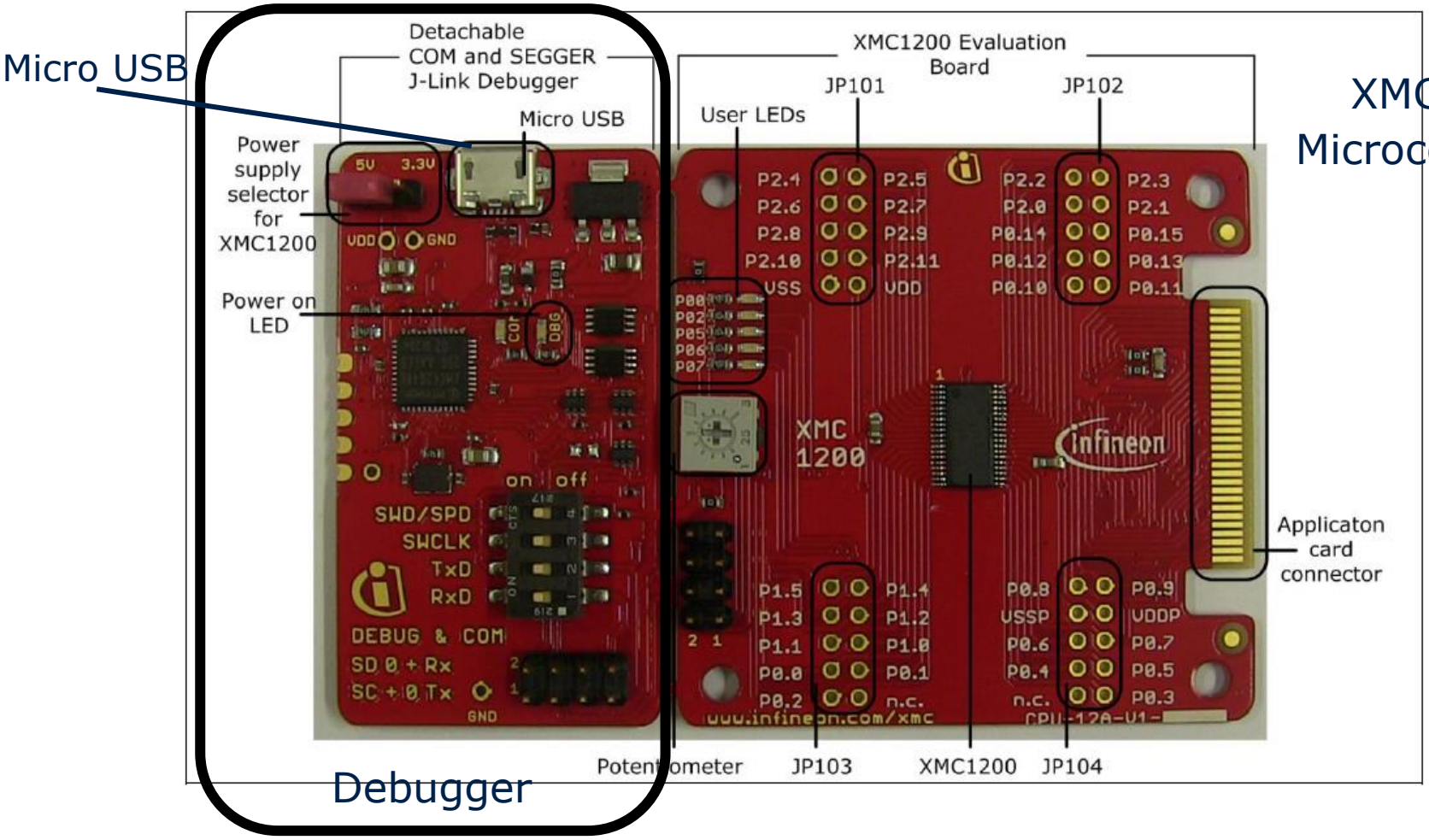


The Project Folder



Flash and Debug (1/3)

- Ensure the Debugger of the XMC 1200 BootKit is connected to your PC via USB

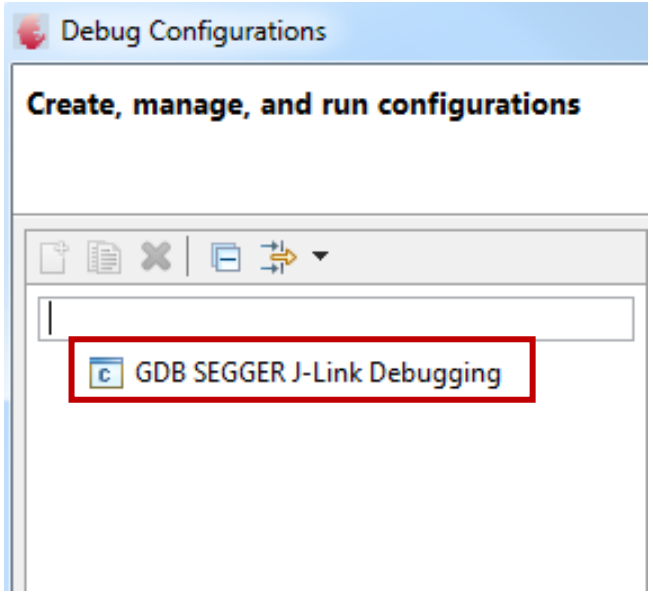


Flash and Debug (2/3)

- Start Debug Session
  - Click  in the tool panel

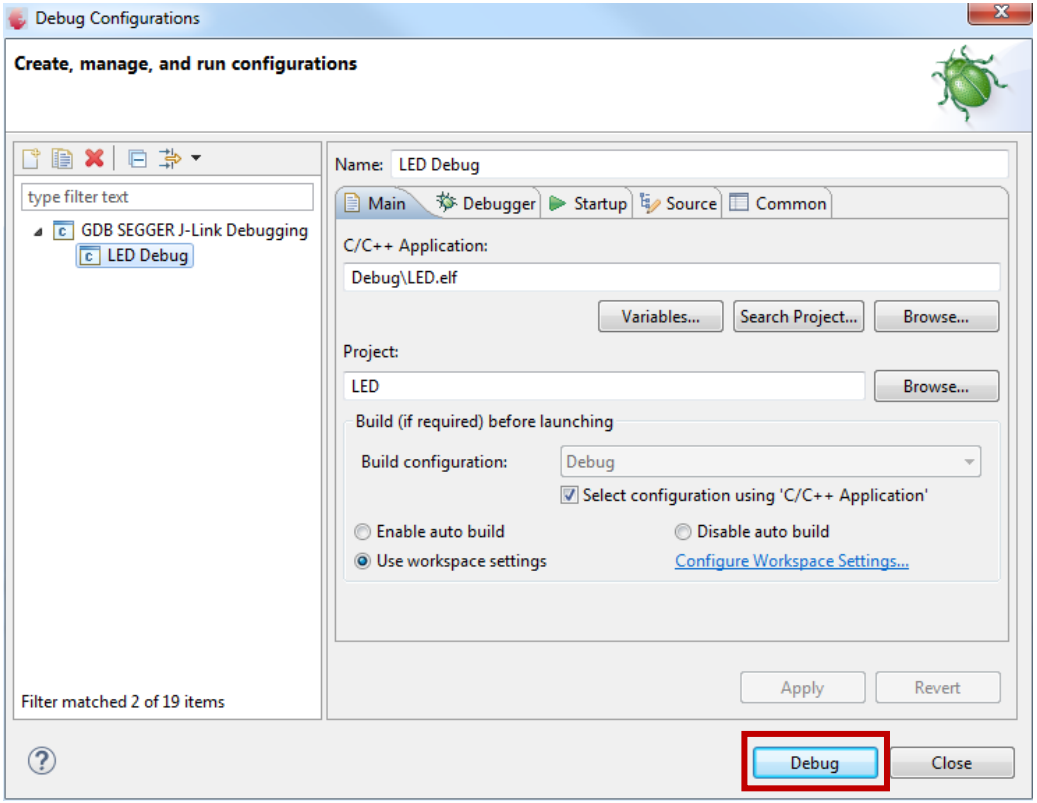
- Create a new Debug Configuration
  - Double-click "GDB SEGGER J-Link Debugging"

Segger J-link Driver software 4.96h or above needs to be installed



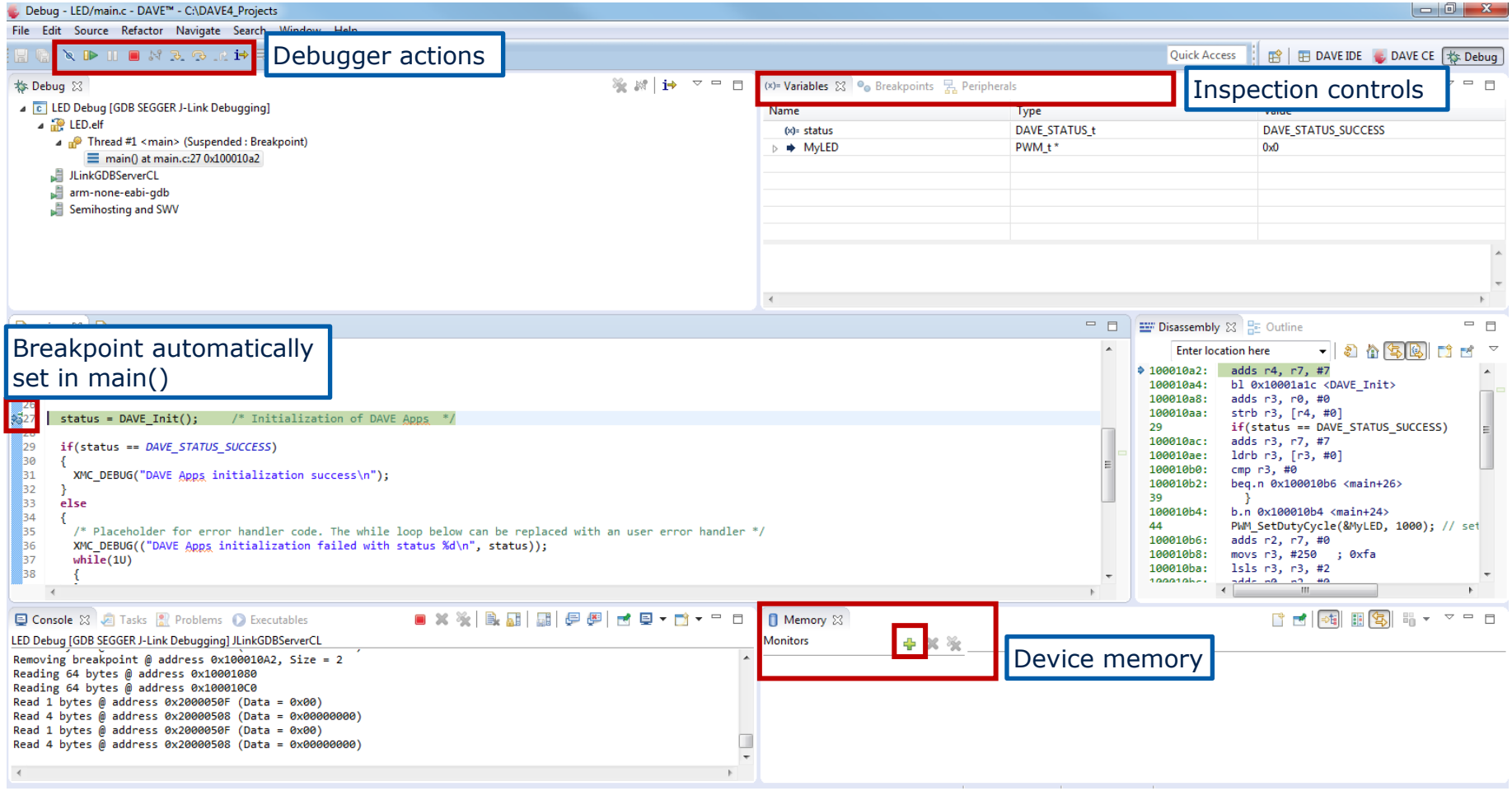
Flash and Debug (3/3)

- Click "Debug"
- The flashing process is started and DAVE automatically switches to Debug Perspective

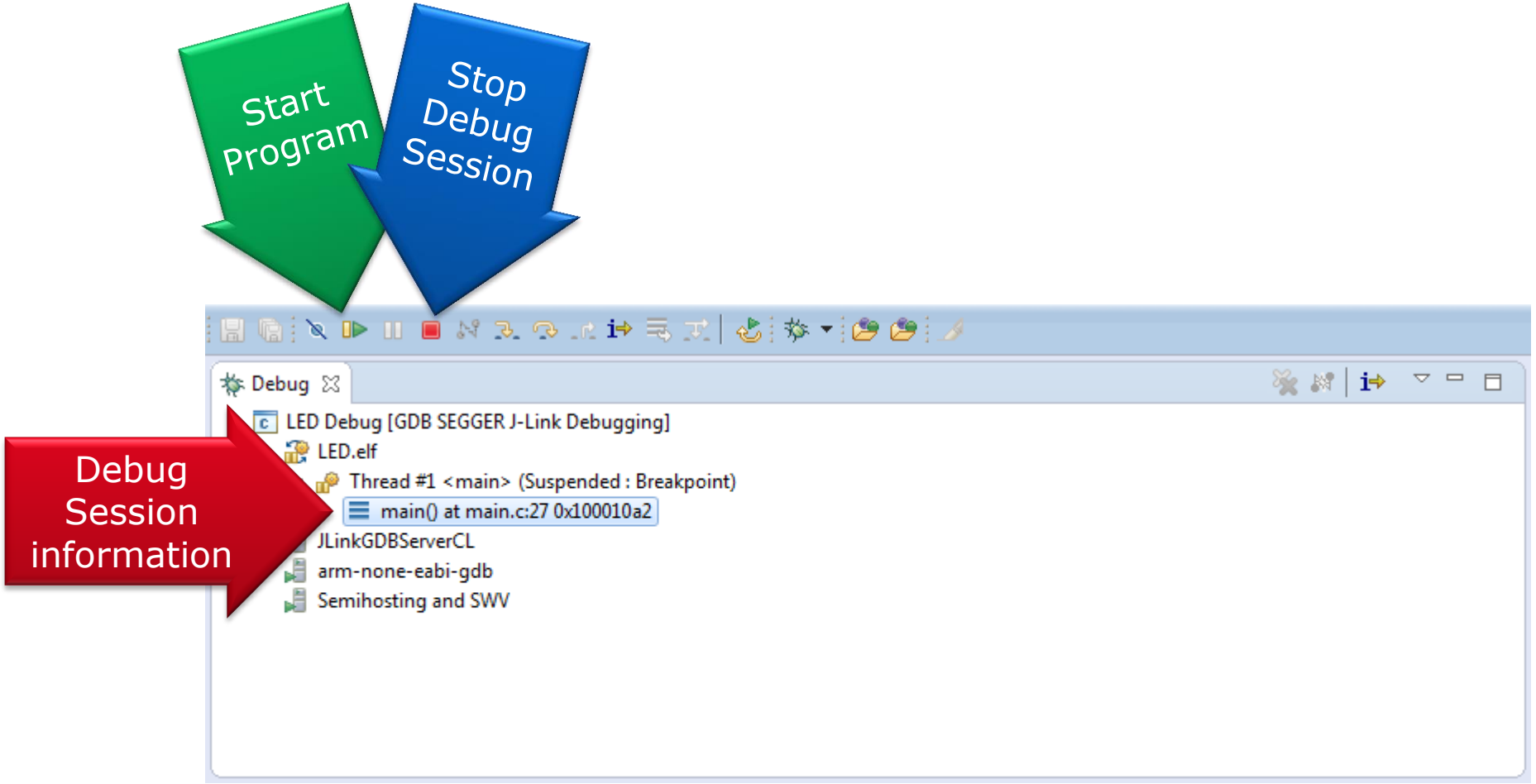


- Hint: To switch to Project Workspace Perspective, click DAVE CE at upper right corner of window

The Debug Perspective (1/6)
Debug Workspace

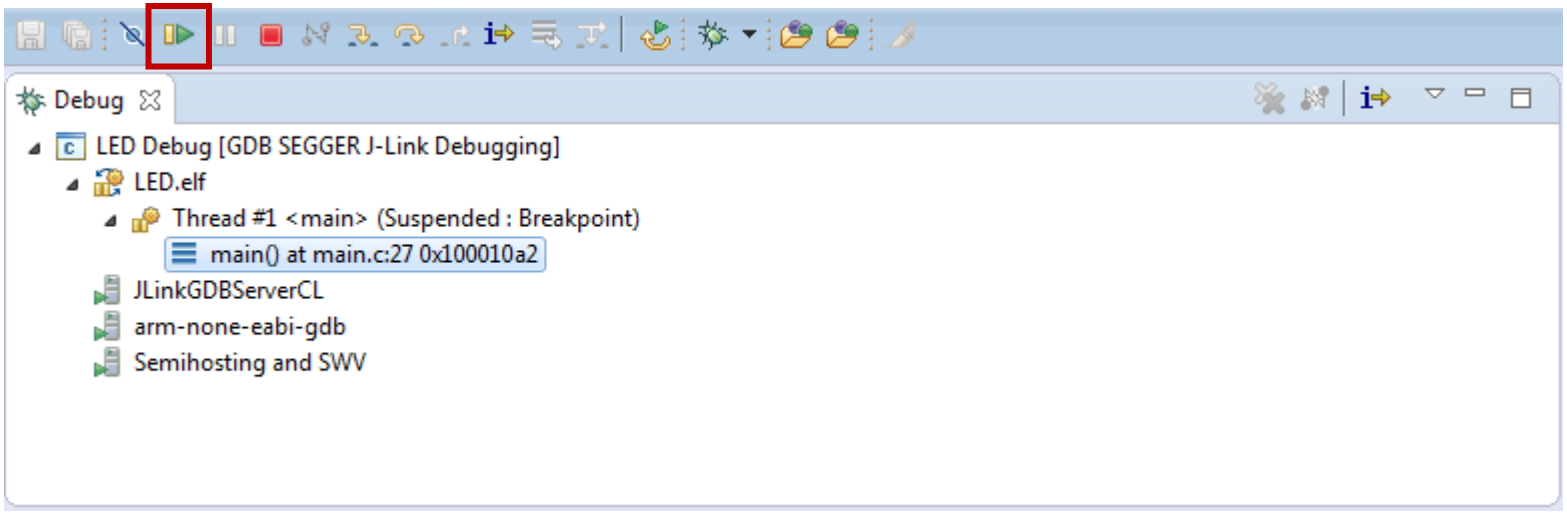


The Debug Perspective (2/6)
Debug Window



The Debug Perspective (3/6)
Start Program

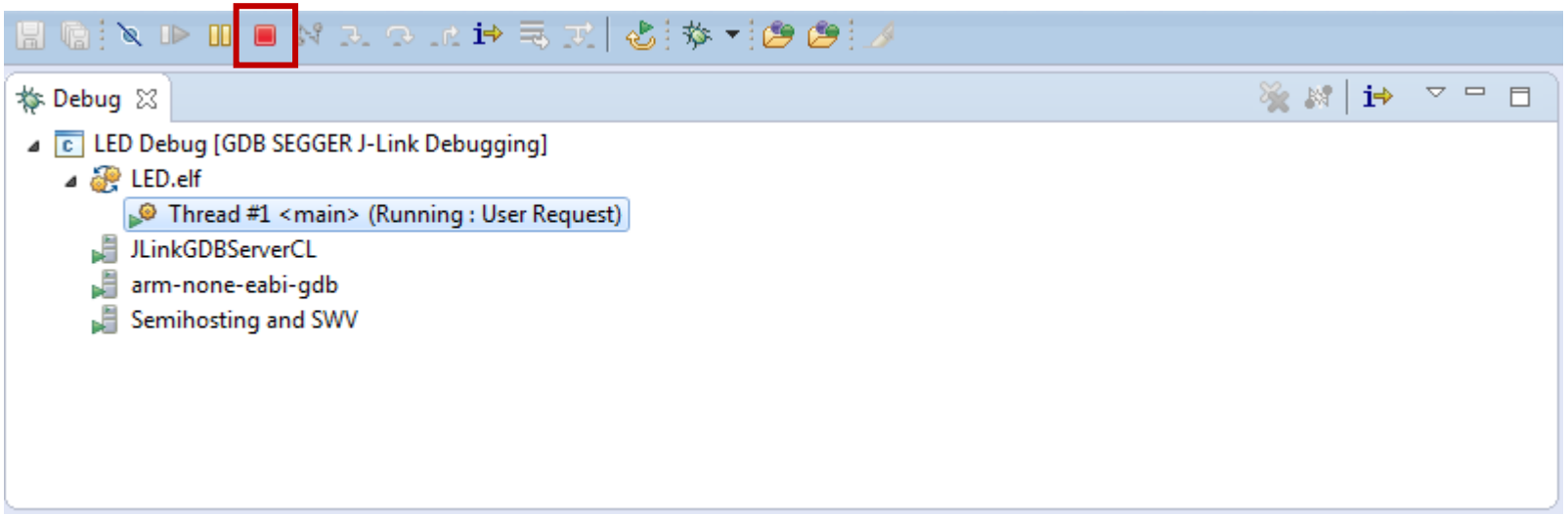
- Click on the Resume button to start code execution



- User LED1 (P0.0) on XMC1200 BootKit board should be blinking

The Debug Perspective (6/6)
End Debug Session

- Always end a debug session by clicking the Terminate Button

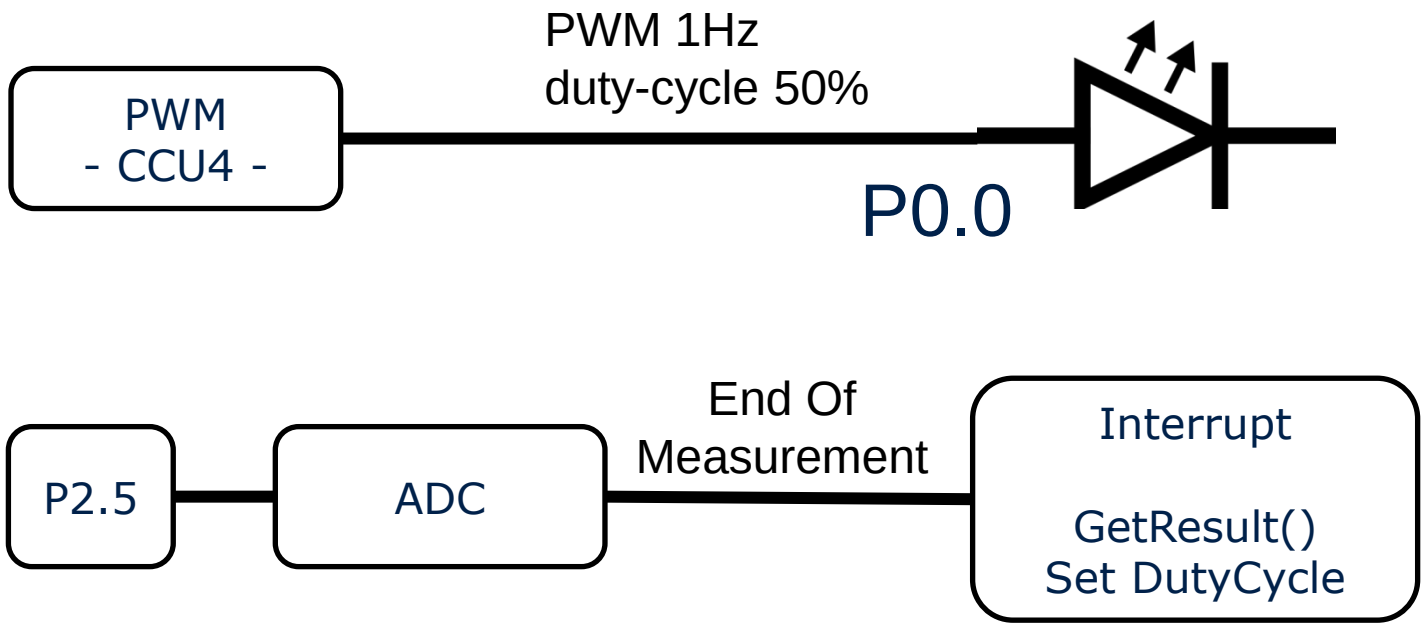


Agenda

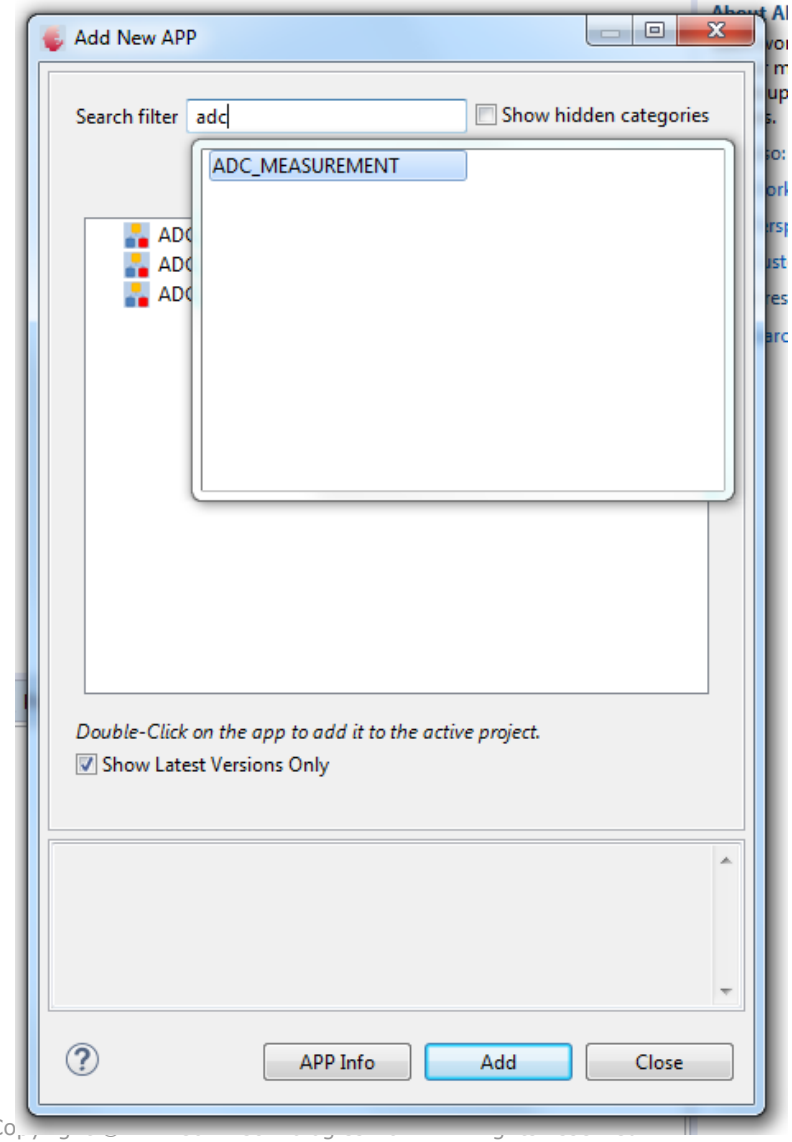
- LLD of XMCLib: use of ACMP
- Blink an LED with PWM APP
- Use ADC APP to update PWM dutycycle

PWM to Blink an LED

- Add ADC Measurement APP to read potentiometer, then update PWM dutycycle when ADC is ready.

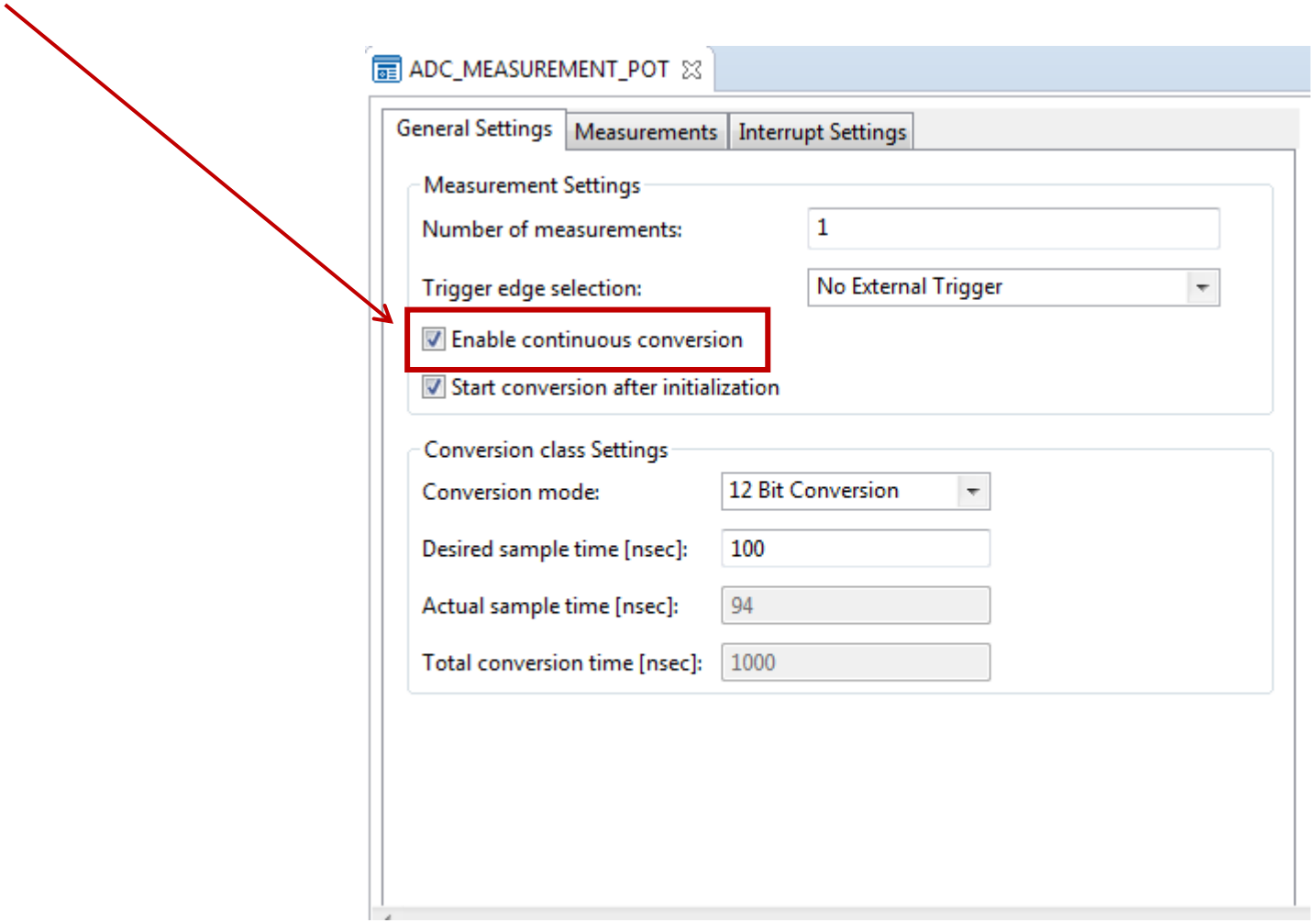


ADD ADC_MEASUREMENT APP



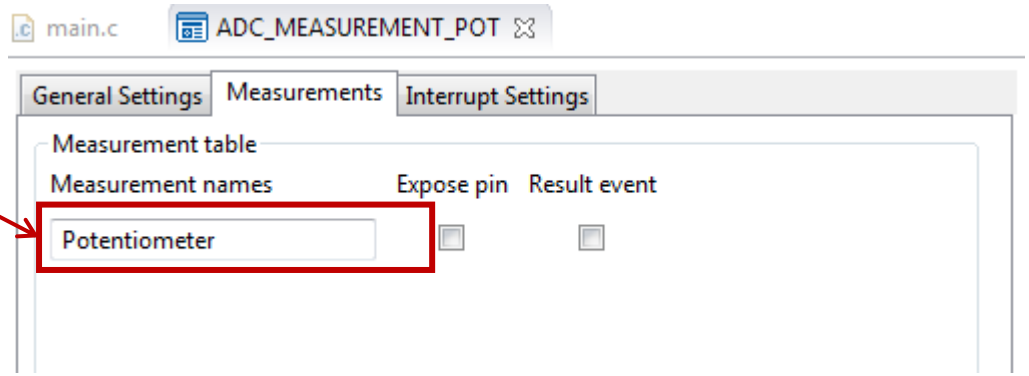
Configure ADC_MEASUREMENT APP [1/2]

- Enable Continuous conversion

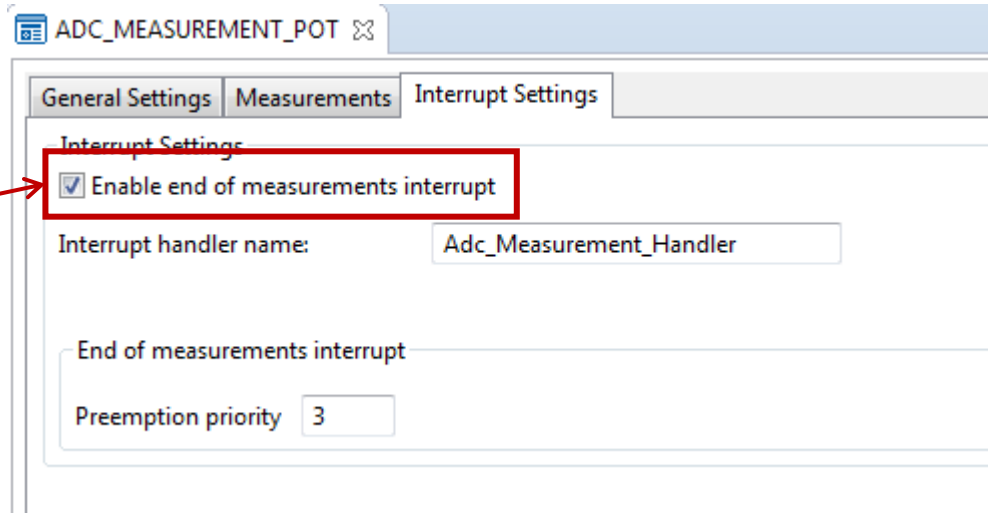


Configure ADC_MEASUREMENT APP [2/2]

- In "Measurements" Change name from "Channel_A" to "Potentiometer"



- In "Interrupt setting" Set "Enable end of measurements interrupt"



Generate Code and add a few Lines of Code to change the Duty Cycle of the PWM

- One touch code generation
  1. Click  in the tool panel
- ADD this code

```
58    ADC_MEASUREMENT_StartConversion(&ADC_MEASUREMENT_0);
59    /* Placeholder for user application code. The while loop below can be replaced with user application code. */
60    while(1U)
61    {
62    }
63  }
64
65  void Adc_Measurement_Handler()
66  {
67      uint32_t result;
68      result = ADC_MEASUREMENT_GetResult(&ADC_MEASUREMENT_Potentiometer_handle);
69      result = (result*MyLED.config_ptr->config_value->period_value)>>12;
70      PWM_SetDutyCycle(&MyLED, result);
71  }
```

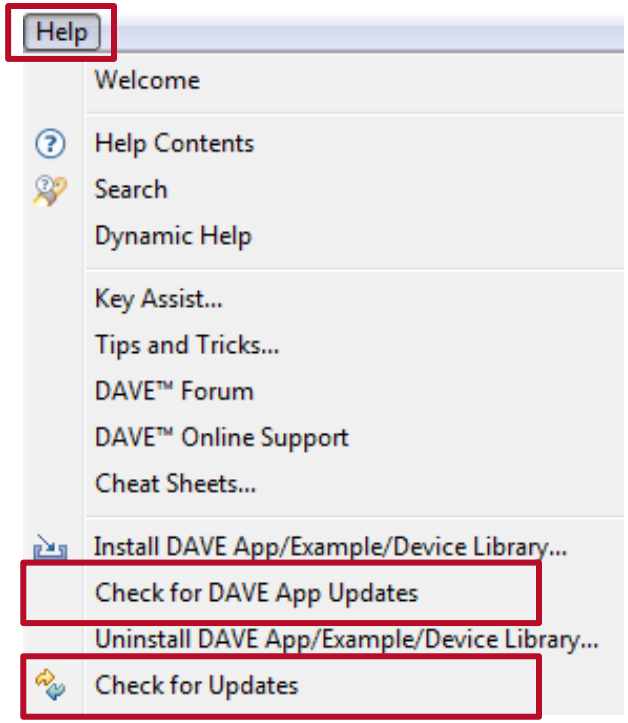
- Start Compiler tools to build the project
  - Click  in the tool panel
- Start Debug Session
  - Click  in the tool panel

One-click DAVE™ update

- DAVE™ APPs and device support can be updated locally
- Re-installation not required

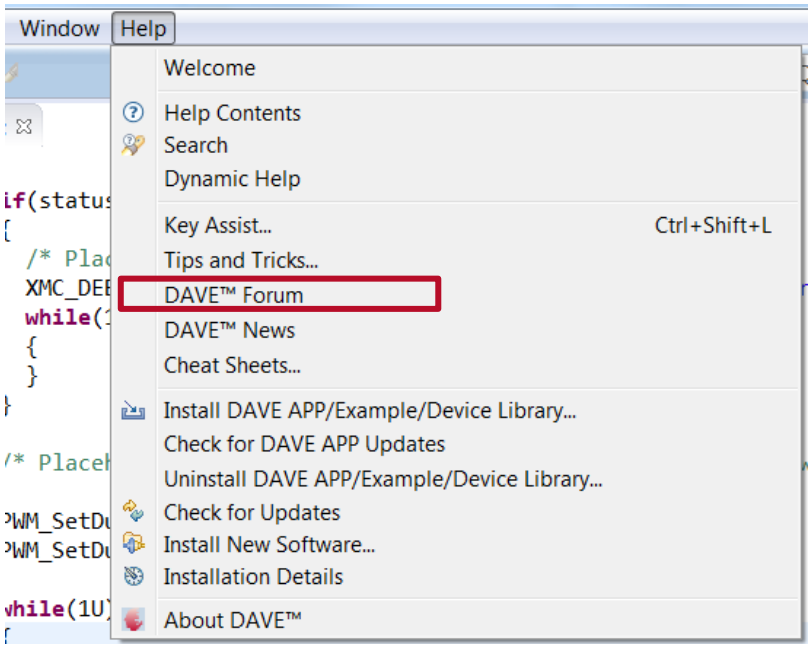
- Update DAVE™ system
  - Help → Check for Updates

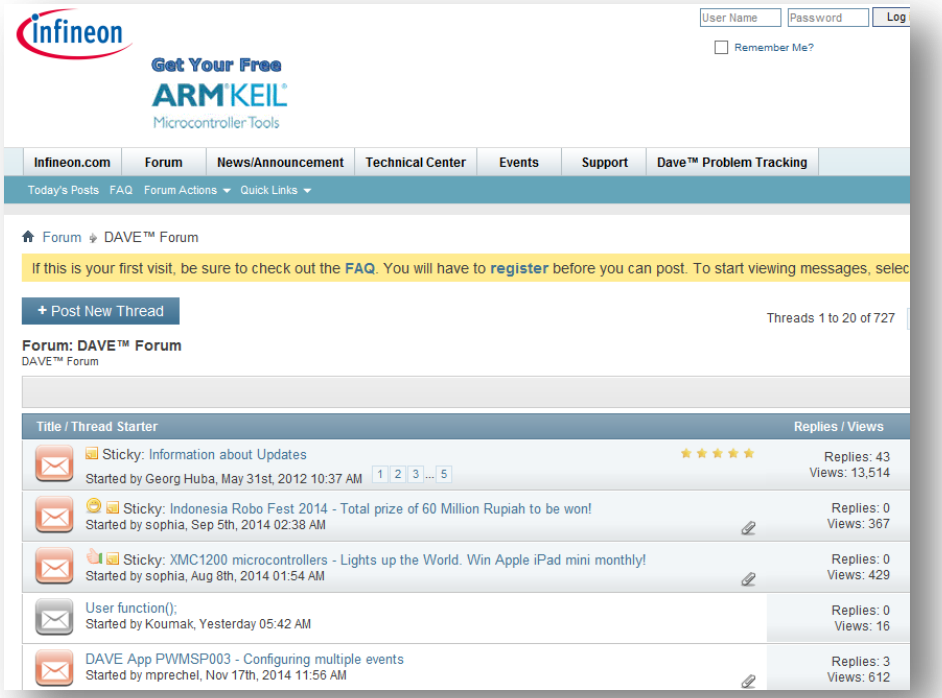
- Update DAVE™ APPs and device support
  - Help → Check for DAVE App Updates



Expert Support

- Easy access to DAVE™ technical support, downloads and information updates





DAVE™ Forum

ENERGY EFFICIENCY
MOBILITY
SECURITY

Innovative semiconductor solutions for energy efficiency, mobility and security.