LED Brightness and Color Control with XMC1

XMC™ microcontrollers

September

Agenda

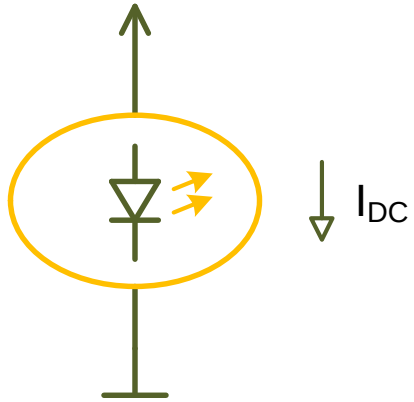
1. Overview
2. LED lighting basics
3. Modulation dimming
4. BCCU highlights
5. Hands-on training (HOT)
6. General information
7. Resource listing

LED Brightness and Color Control Overview

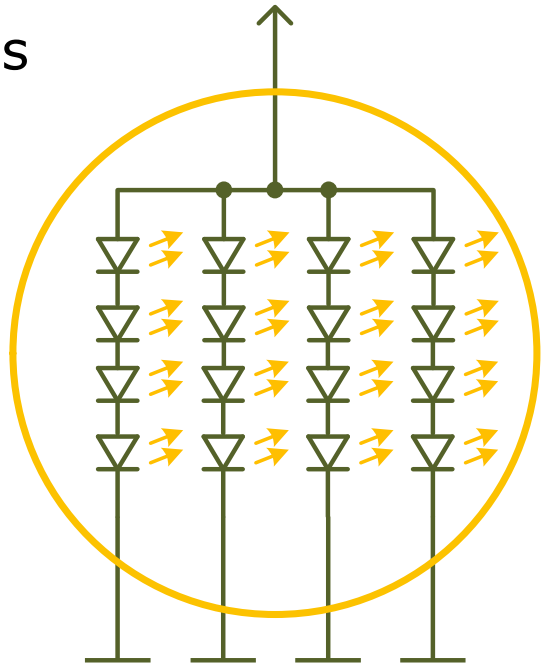
- The purpose of the training slides is to showcase the ease of using the Brightness and Color Control Unit (BCCU) in XMC1 to drive LEDs
- The HOT examples cover key BCCU features to achieve good quality light

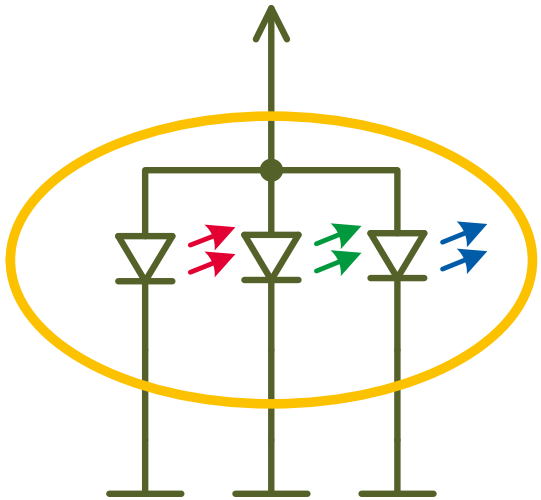
LED Brightness and Color Control
LED lighting basics (1/2)

- Light-emitting diode



- LED lamps

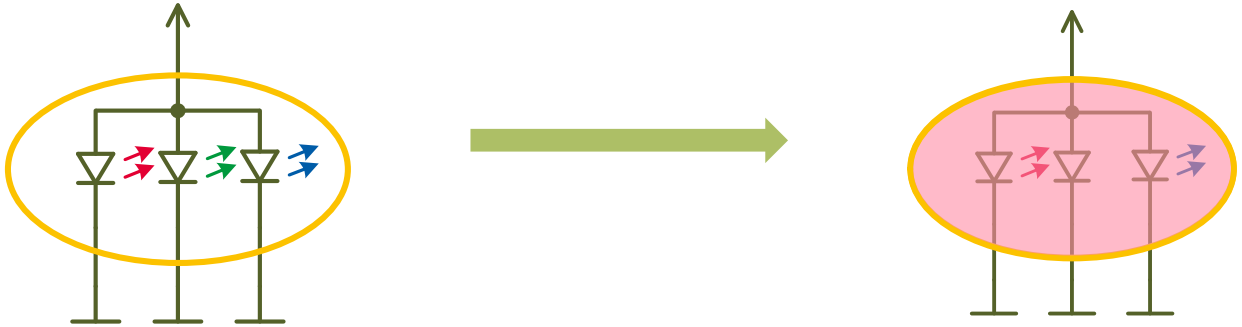




LED Brightness and Color Control
LED lighting basics (2/2)

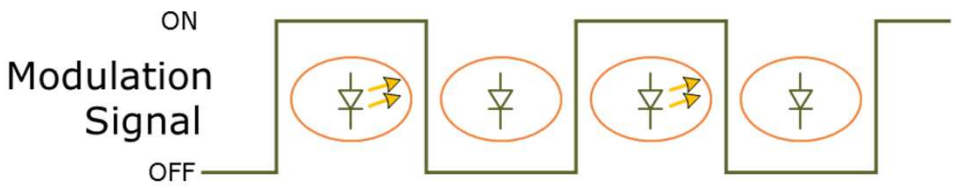
- Controlling LED lamp
  - Brightness

  - Color

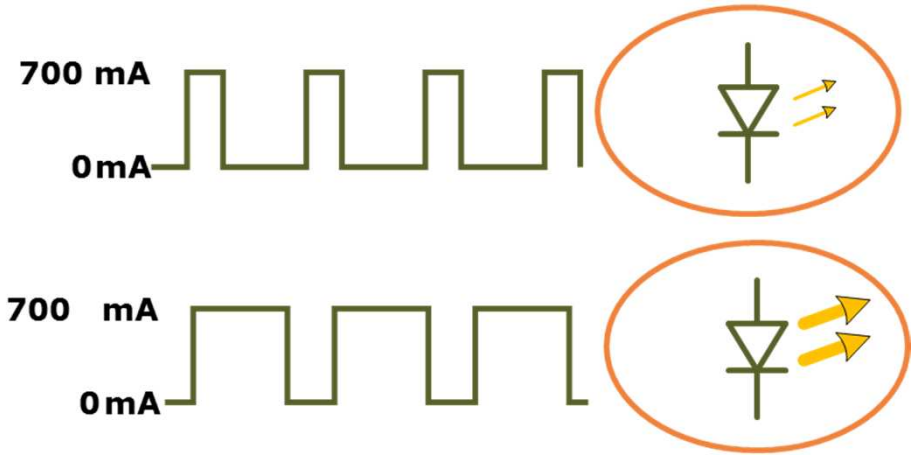


LED Brightness and Color Control
Modulation dimming

- Brightness control method:
  - Modulation dimming
    - Quickly turn current ON/OFF
    - Constant current when ON

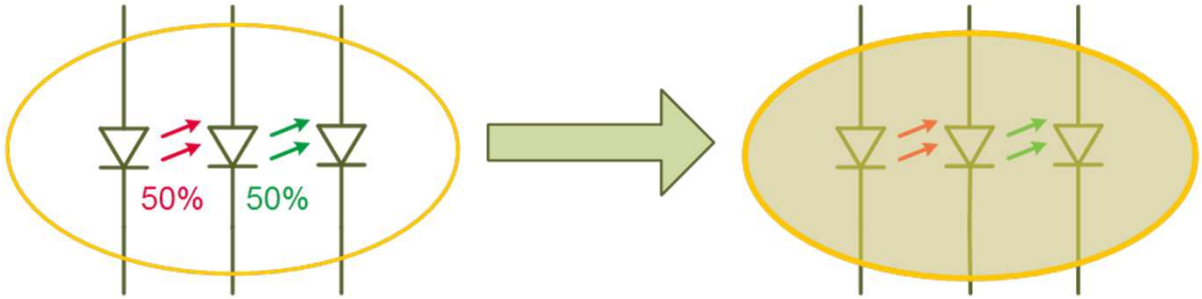


    - Brightness ~ average ON time

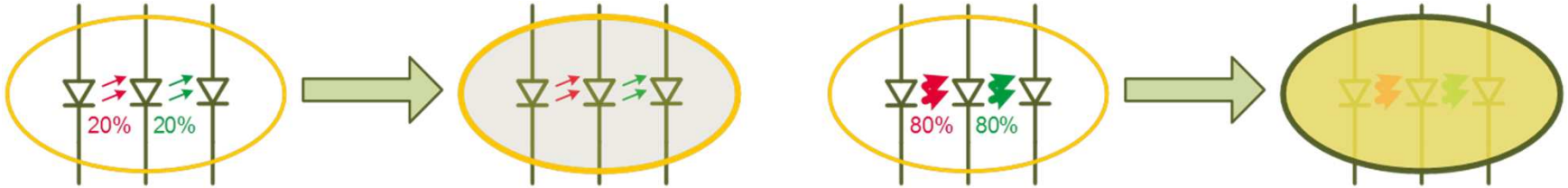


LED Brightness and Color Control
Lamp color

- Color
  - Relative channel intensities determine color

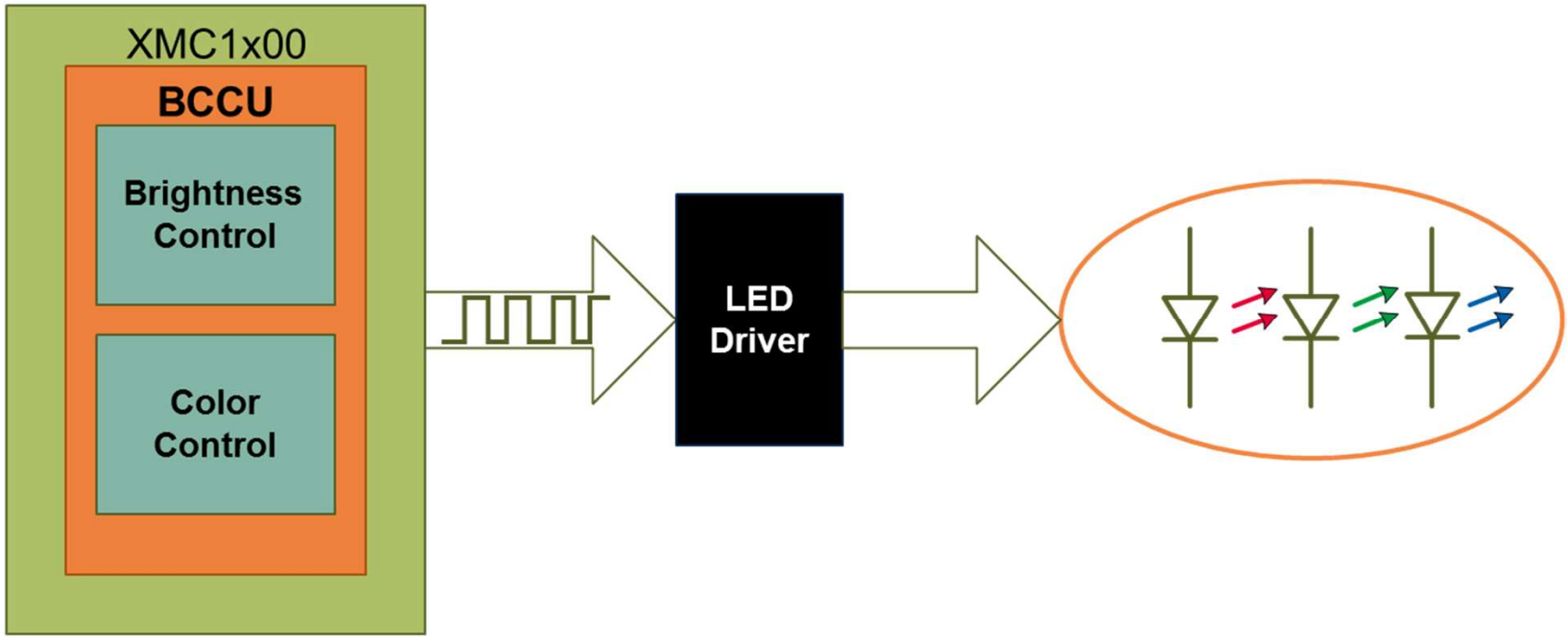


- Color vs dimming level
  - Dimming level determines the color brightness



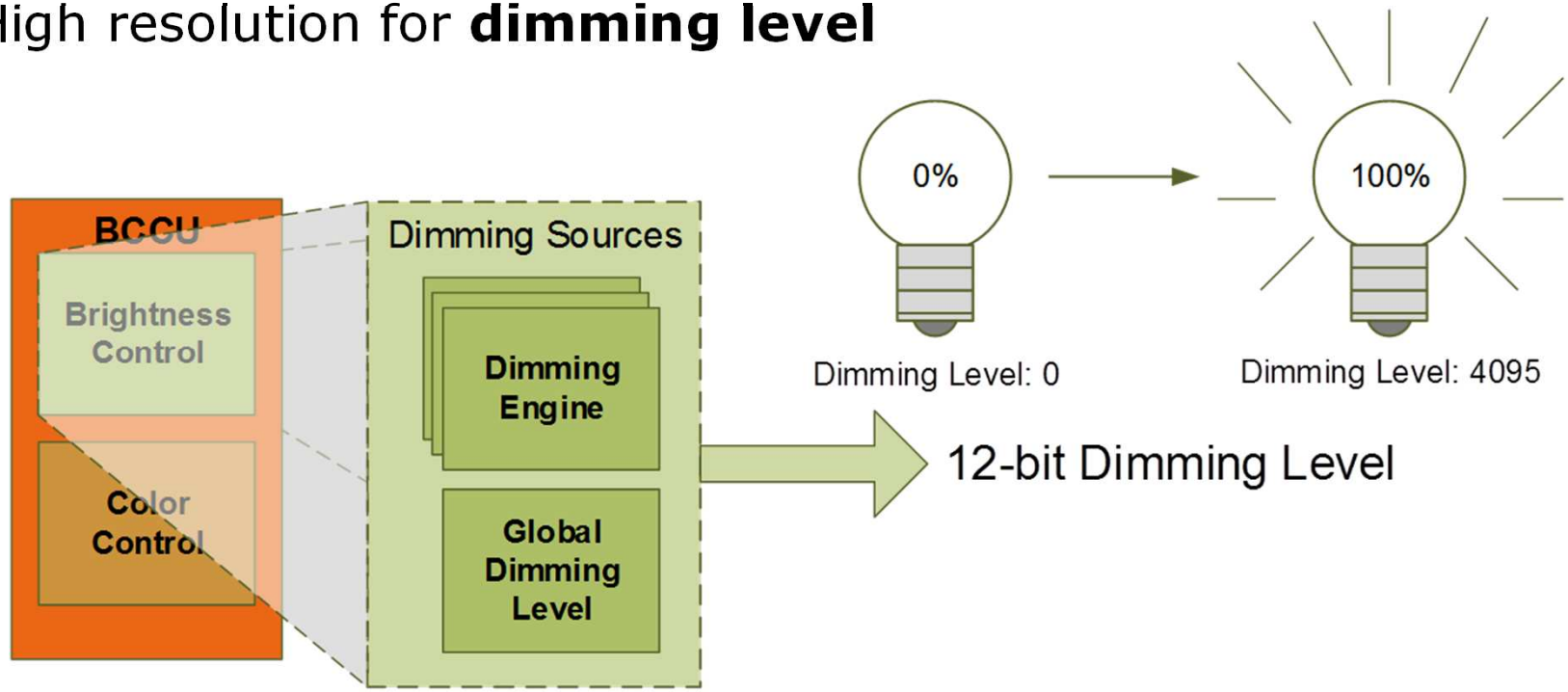
LED Brightness and Color Control
BCCU in XMC1000

- Brightness and Color Control Unit
  - Separate control blocks for brightness and color



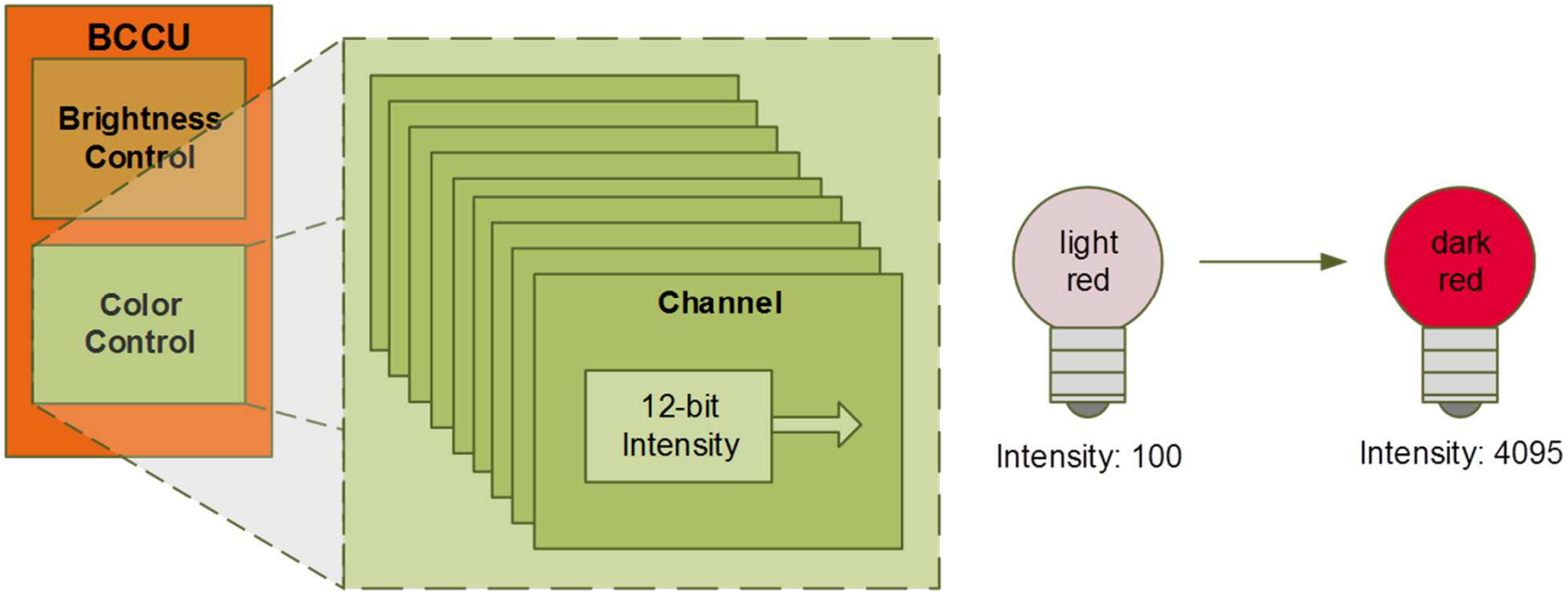
LED Brightness and Color Control
Brightness control with BCCU

- Dimming sources
  - Dimming engines
  - Global dimming level
- High resolution for **dimming level**



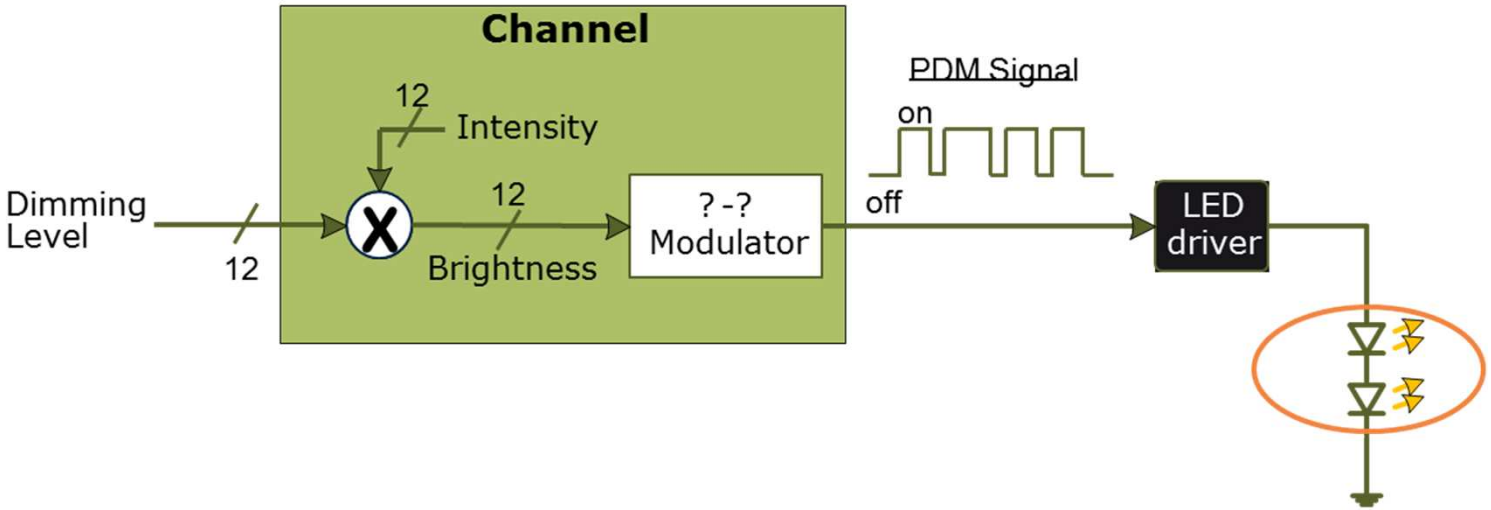
LED Brightness and Color Control
Color Control with BCCU

- BCCU channels
  - Color can be controlled using 12-bit **intensity** value



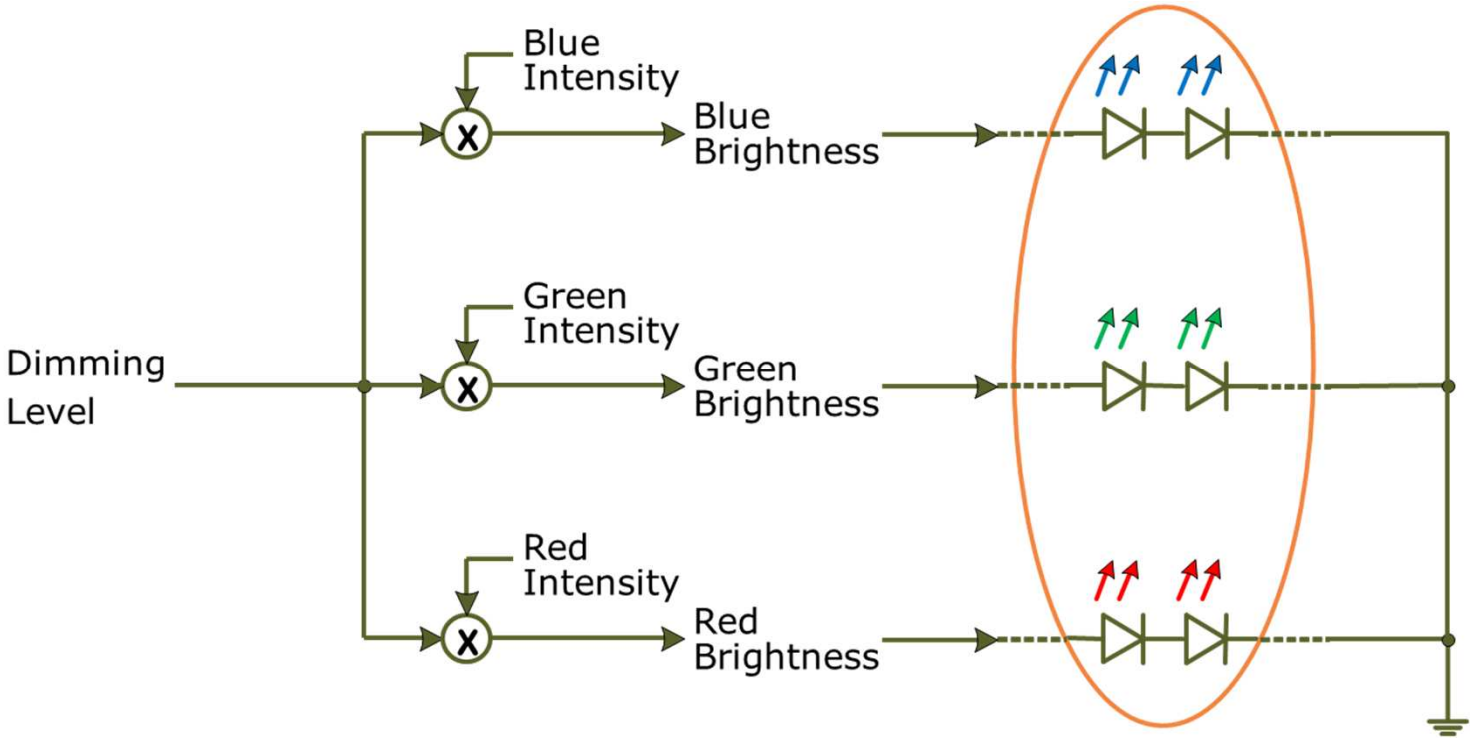
LED Brightness and Color Control
Dynamic control with BCCU (1/3)

- Channel **brightness**
  - Dimming level x channel intensity
  - Truncated 12-bit value
- Sigma-delta modulator
  - Converts 12-bit brightness value to a bit stream
  - Bit stream is a pulse-density modulated (**PDM**) signal



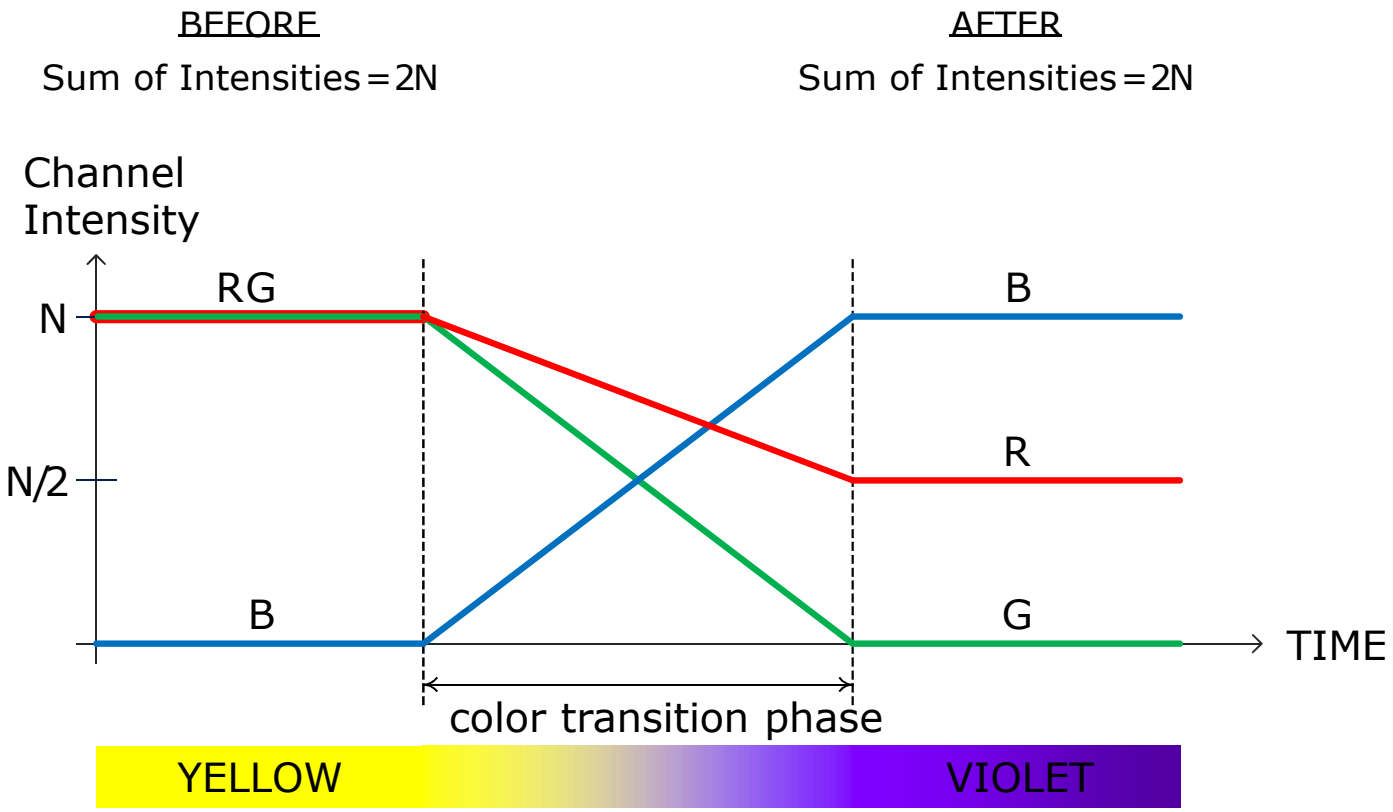
LED Brightness and Color Control
Dynamic control with BCCU (2/3)

- Changes to dimming level
  - Lamp color can be preserved if no changes to channel intensities

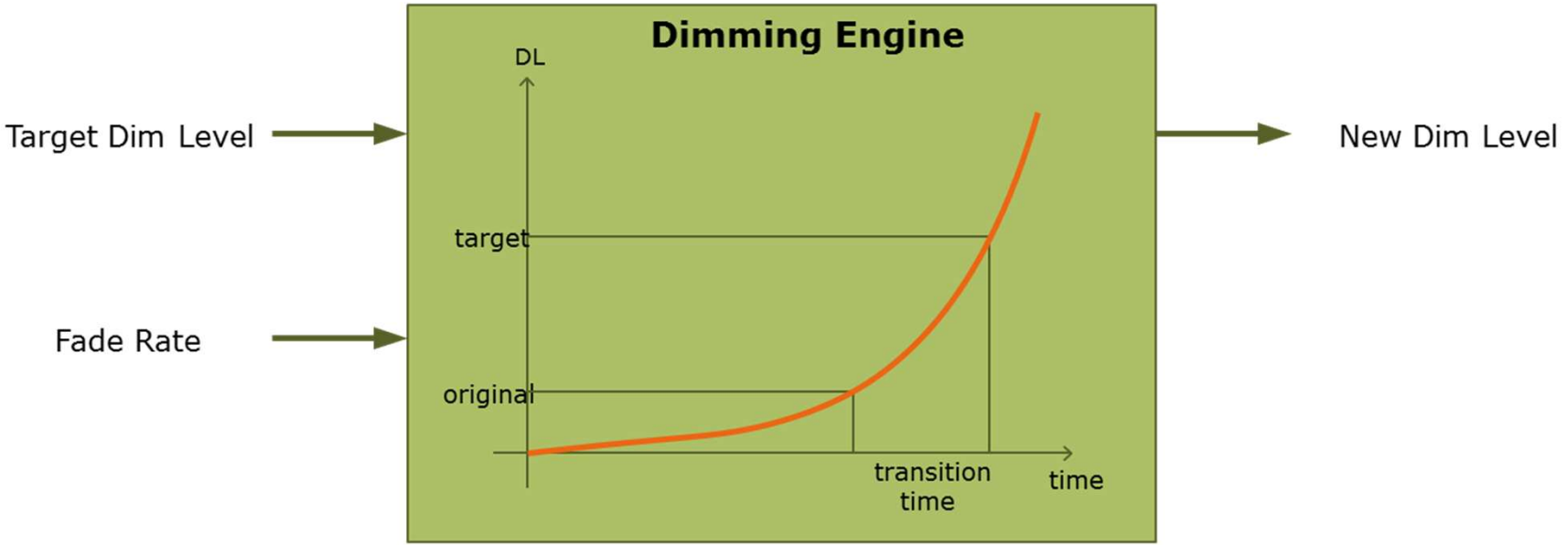


LED Brightness and Color Control
Dynamic control with BCCU (3/3)

- Changing lamp color
  - Overall lamp brightness can be preserved if sum of channel intensities remain the same



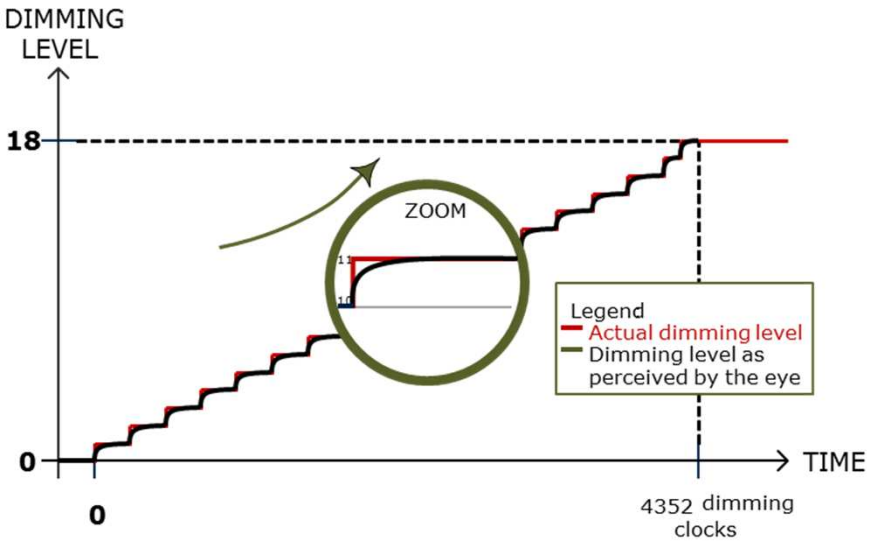
LED Brightness and Color Control
BCCU dimming sources (1/3)

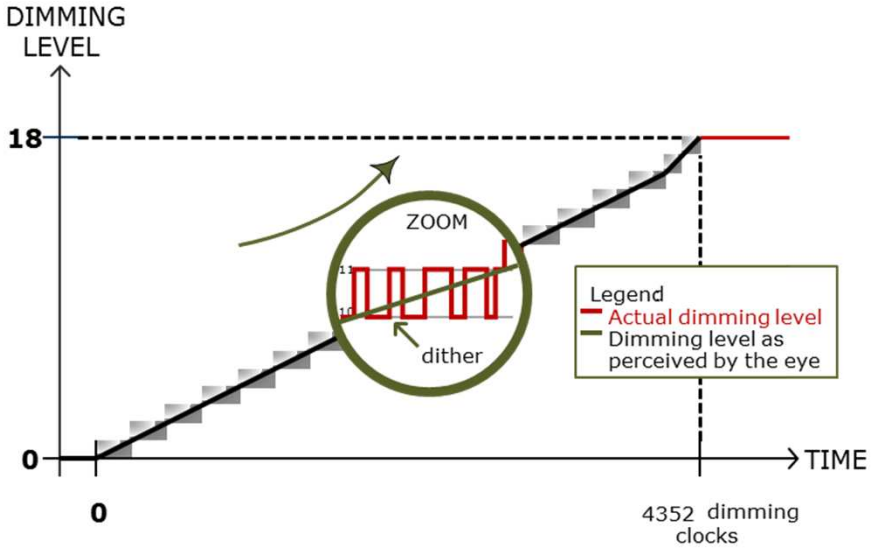


- Dimming engine
  - Automatic exponential dimming profile
    - Human eye logarithmic perception of brightness
  - Configurable fade rate
  - Easy to use:
    1. Program target level
    2. Program fade rate
    3. Start dimming process

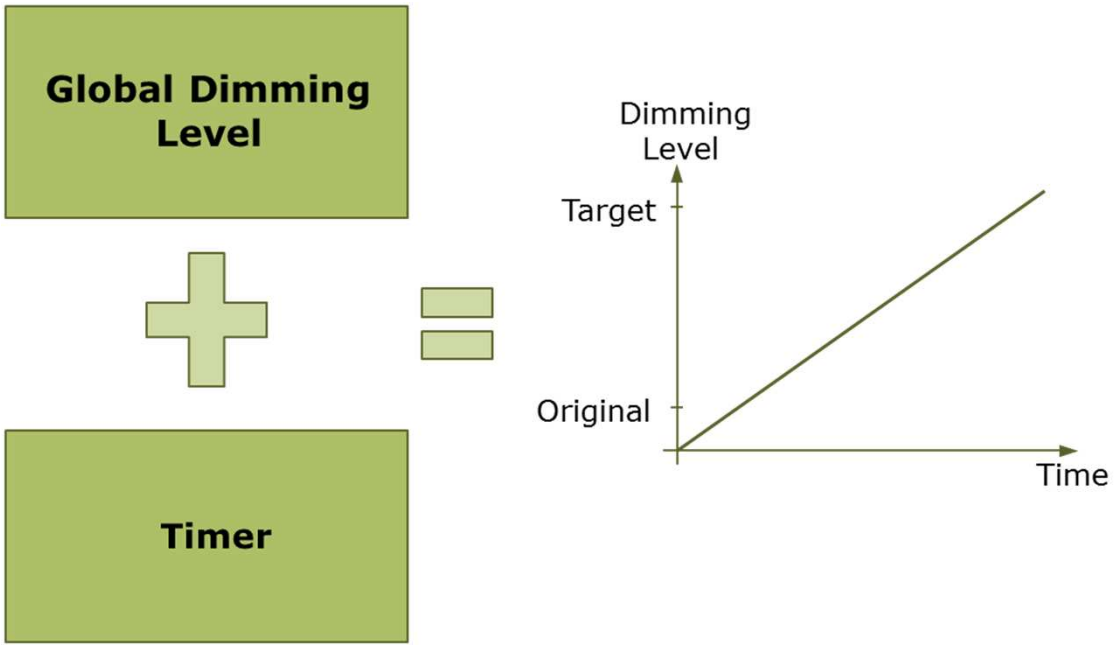
LED Brightness and Color Control
BCCU dimming sources (2/3)

- Dimming at low dim levels with slow fade rate
  - Human eye more sensitive to changes in brightness
  - Changes appear “steppy”
  - Dither function in dimming engines to make transitions smooth



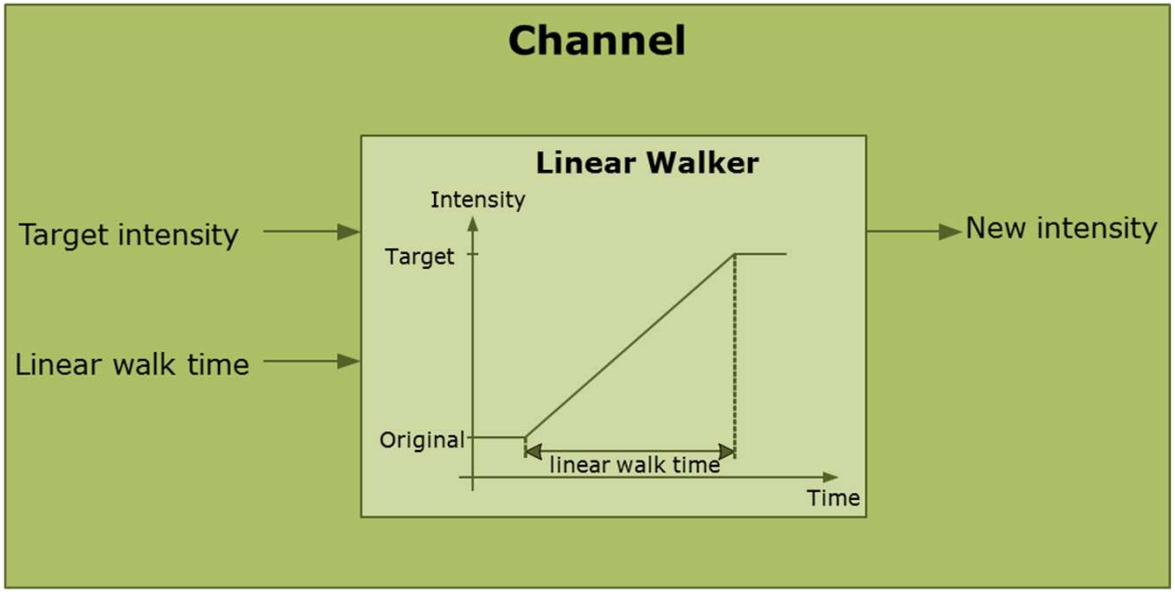


LED Brightness and Color Control
BCCU dimming sources (3/3)



- Global dimming level
  - Immediate dim level change
  - For manual or customized dimming pattern
    - Example: Use with periodic timer for linear dimming profile

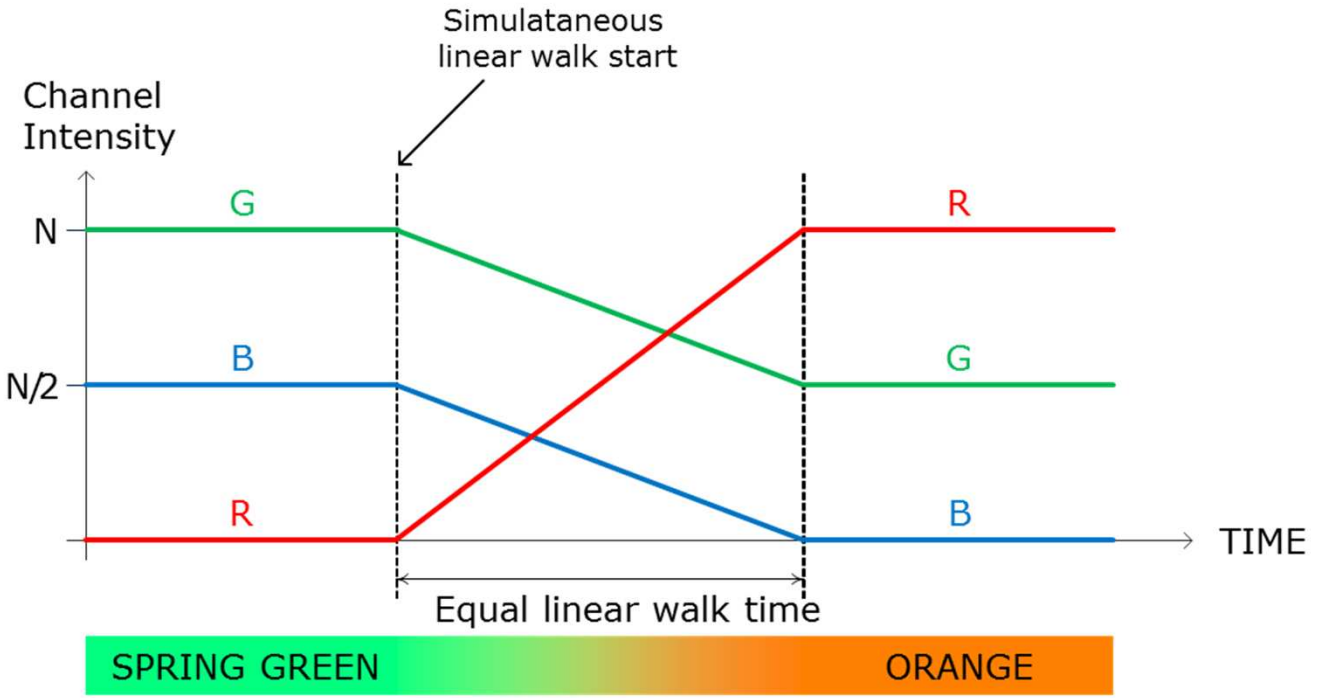
LED Brightness and Color Control
Linear walker (1/2)



- Linear Walker in Channel
  - Changes intensity linearly over time
  - Configurable color transition time
  - Easy to use:
    1. Set target intensity
    2. Set linear walk time
    3. Start linear walk

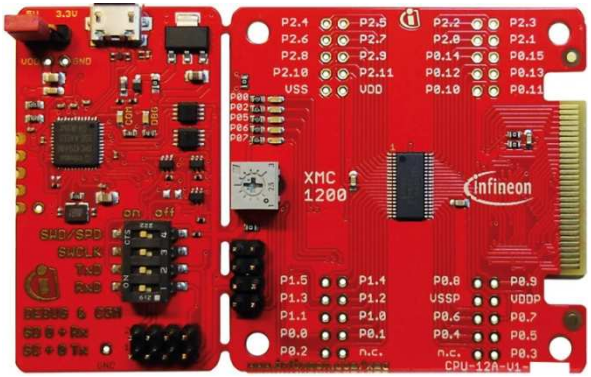
LED Brightness and Color Control
Linear walker (2/2)

- For smooth color change:
  - Linear walk time must be the same in all channels
  - Linear walks are started simultaneously in all channels

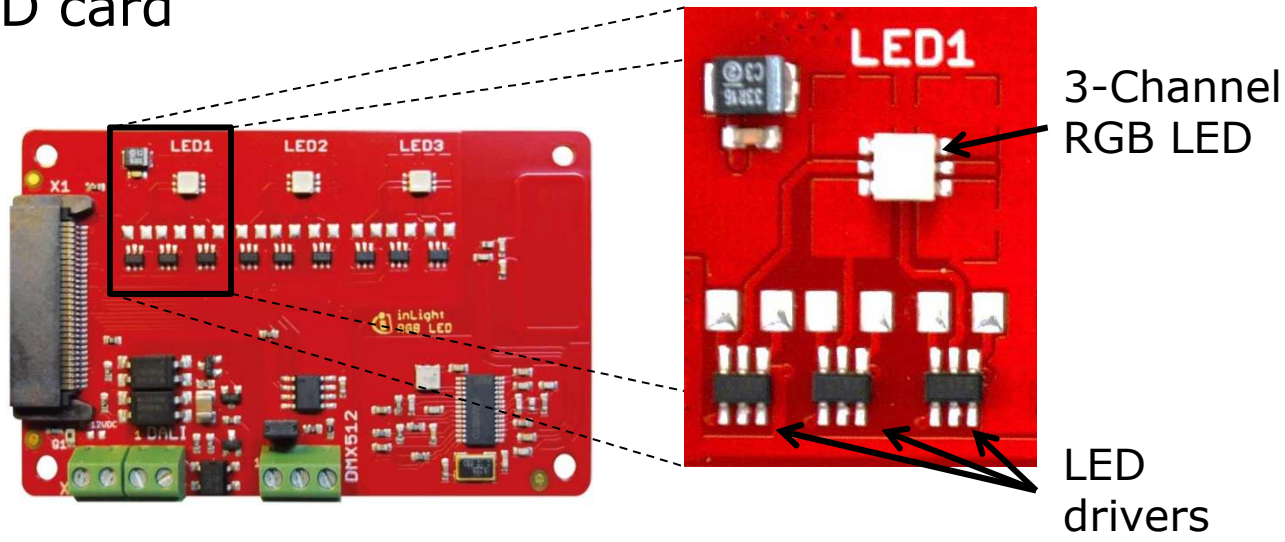


LED Brightness and Color Control Hands-on training

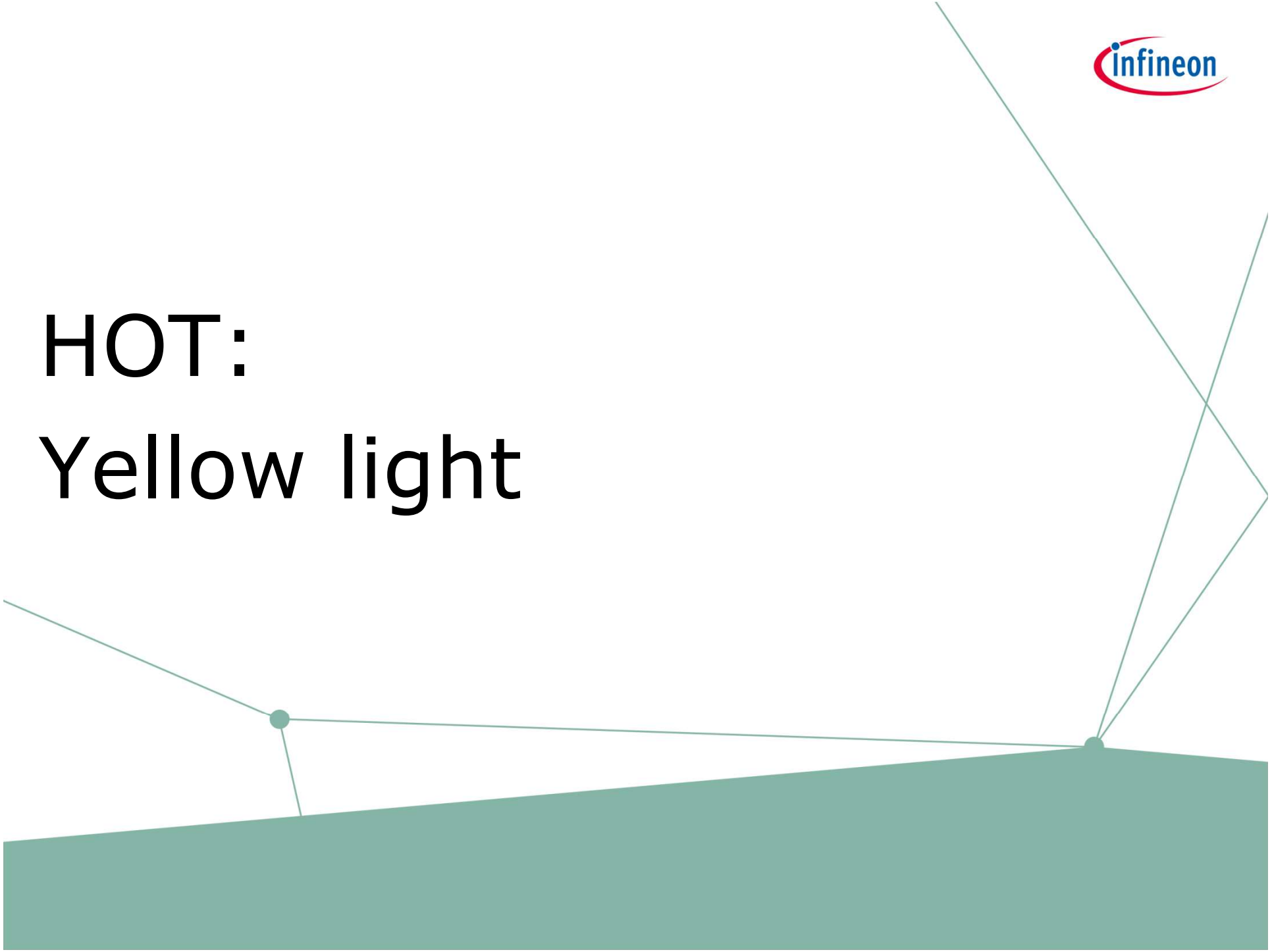
- XMC1200 boot kit



- Color LED card



HOT:
Yellow light



Objectives

- To demonstrate using DAVE™ APPs to turn on an RGB LED with yellow light
  - LED will immediately turn on at start with yellow light

Things you need for the training

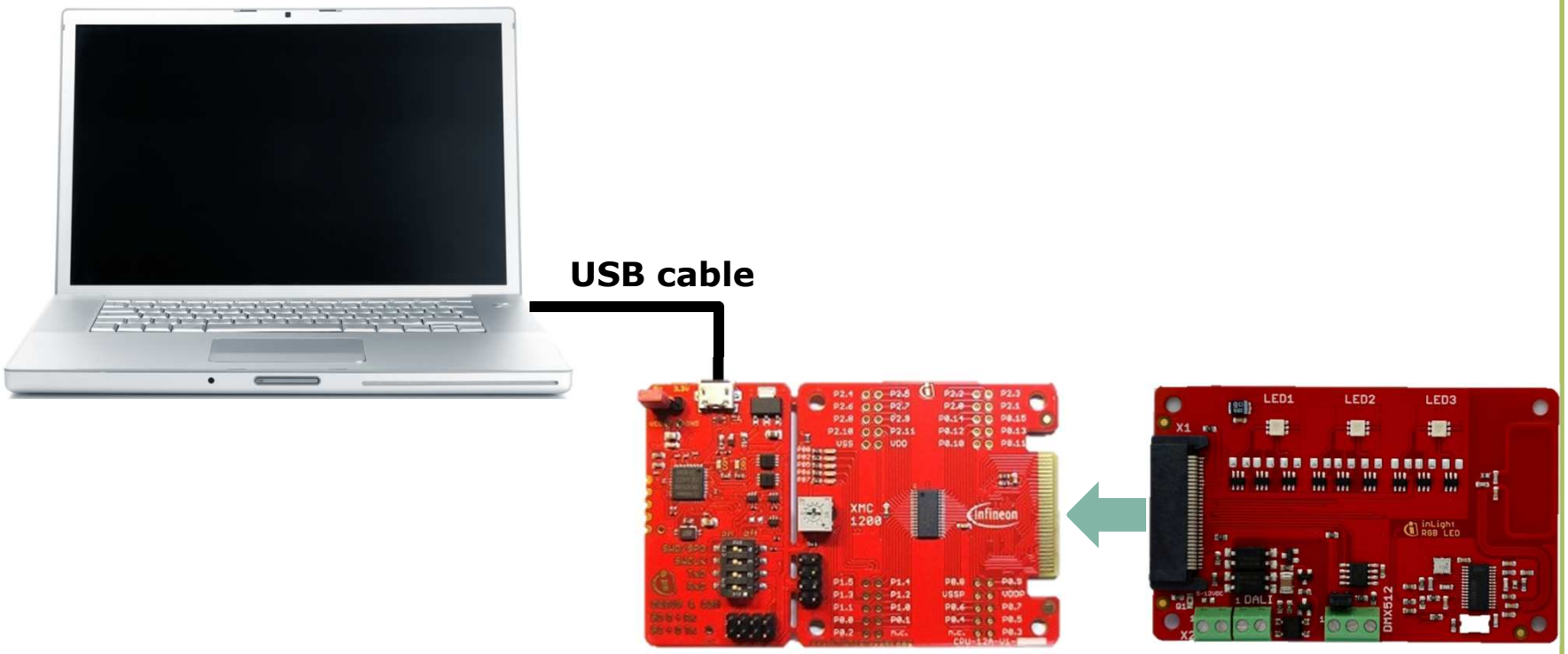
- Hardware
  - XMC1200 boot kit
  - Color LED card
    - Only LED1 will be used
  - PC
  - USB cable

- Software/tools
  - DAVE™ version 4

- **Things you must know before starting on this training**
  - We assume that you are equipped with the basic knowledge to create a DAVE™ version 4 project, compile a software and enter into a debug environment.

For more information on the kit and tool, please refer to General Information.

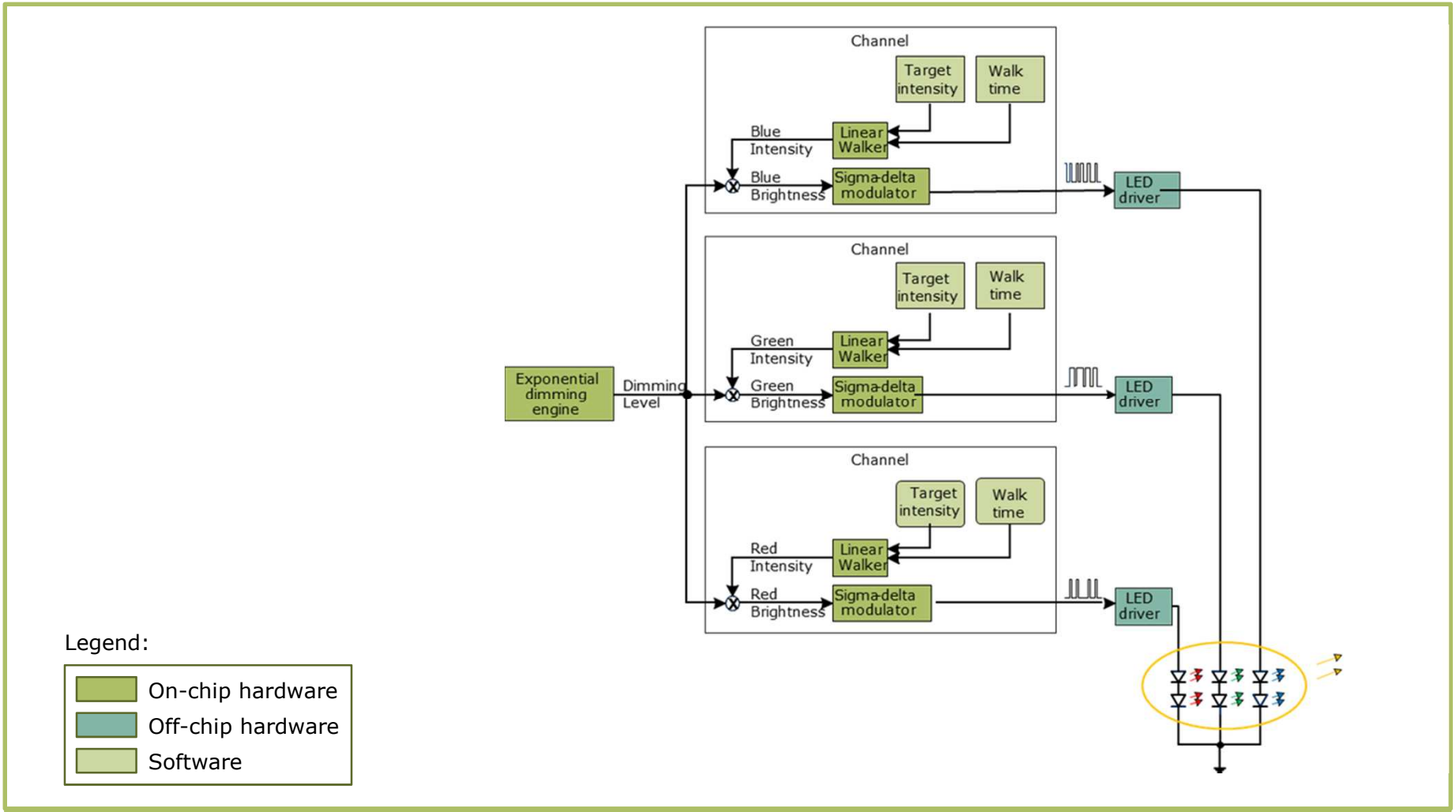
Hardware setup



1. Plug the color LED card onto the XMC1200 boot kit.
2. Connect the debug USB of the XMC1200 boot kit to the PC using the USB cable.

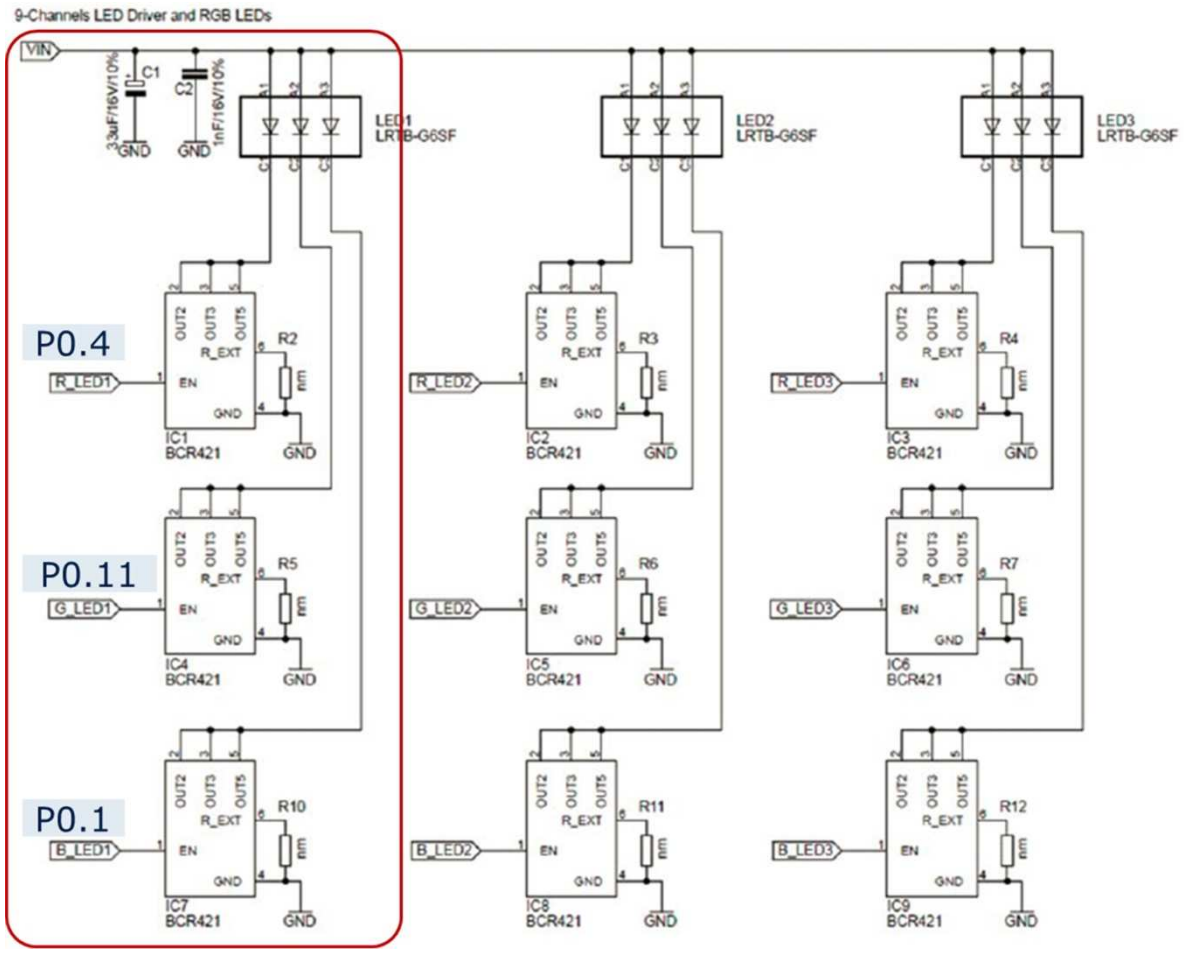
Hardware setup: yellow light

Peripheral configuration



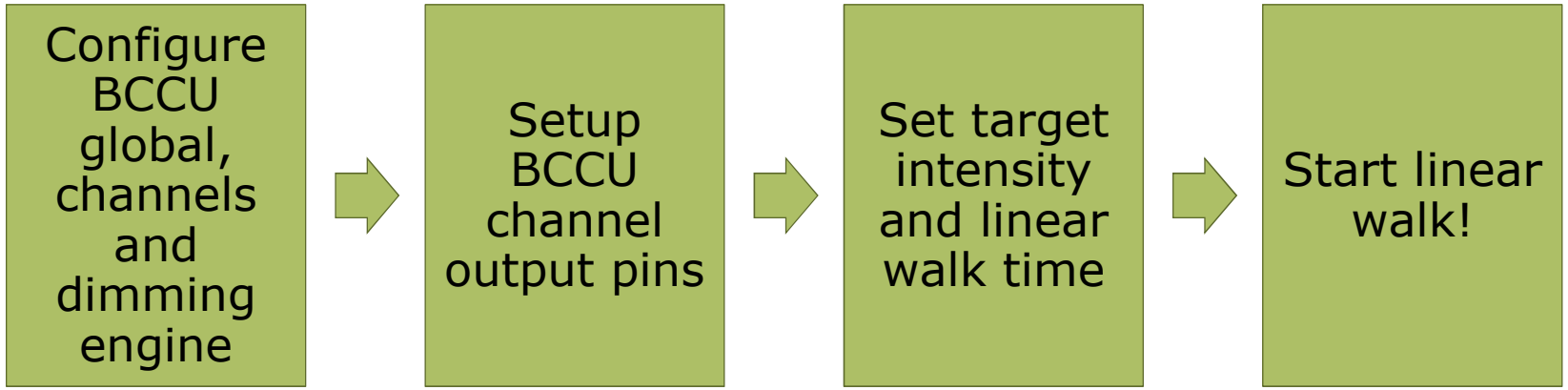
Block diagram: yellow light

Board schematics



Schematic: yellow light

Software overview



Flow chart: yellow light

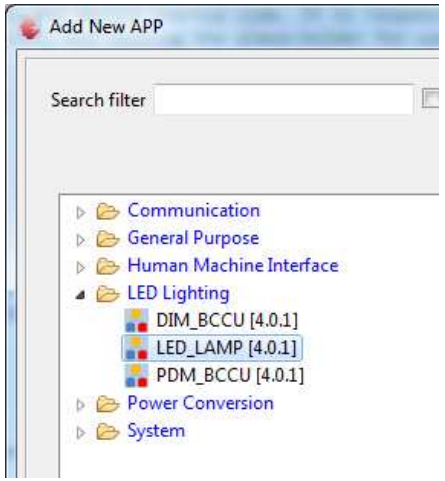
Software, list of DAVE™ APPs

- LED_LAMP
  - Configures the system and peripheral clocks
  - Configures BCCU global, channels and dimming engine
  - Provides control for channels and dimming engine

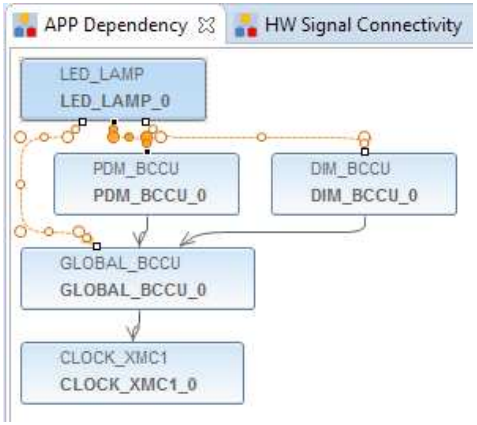
- DIGITAL_IO
  - Initializes IO pins that are connected to the RGB LEDs on board

HOT (1/13)
Create new DAVE™ project

- First, create a new DAVE™ CE project
- Add LED_LAMP APP to the project

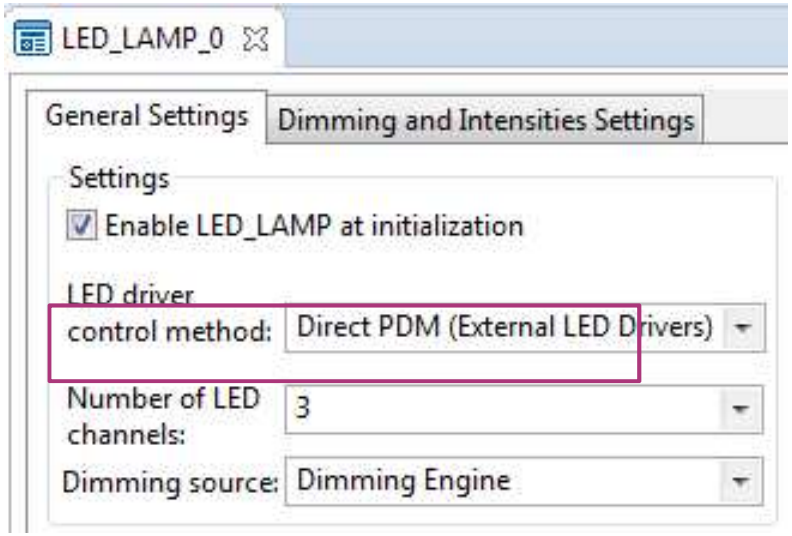


- Double-click LED_LAMP_0 to open the configuration UI



HOT (2/13)
APP configuration (LED_LAMP)

- RGB LED has 3 Channels
- Under “General Settings” tab
  - Configure 3 LED channels



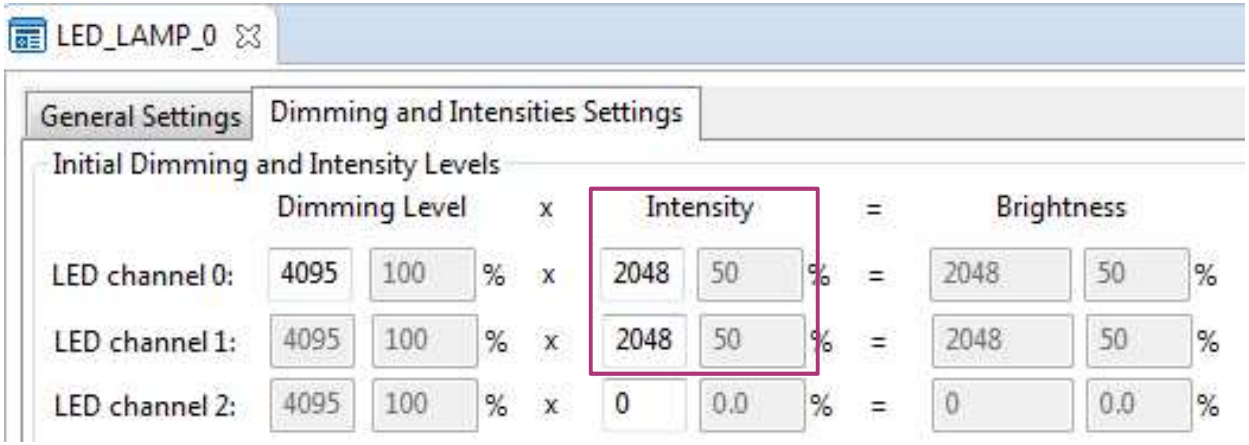
HOT (3/13)
APP configuration (LED_LAMP)

- Set initial color
  - For this project, we will be using a simple color scheme
    - Concept: Sum of channel intensities = max intensity (4095)

Desired color	RED channel intensity	GREEN channel intensity	BLUE channel intensity
Red	4095	0	0
Green	0	4095	0
Blue	0	0	4095
White	1365	1365	1365
Yellow	2048	2048	0

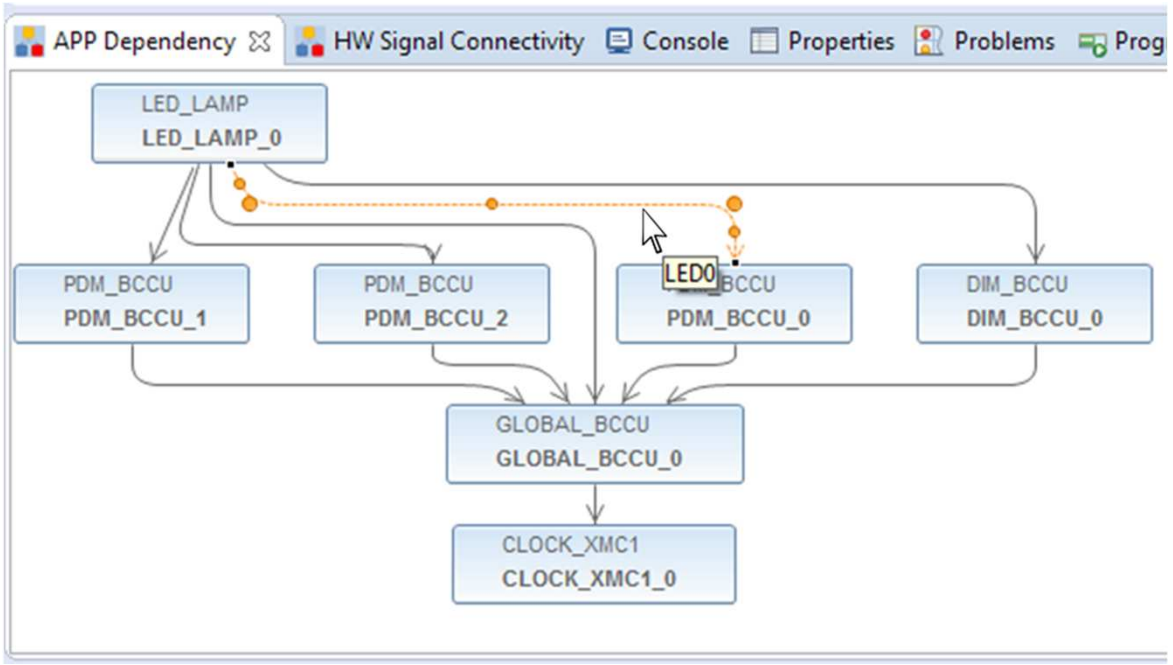
HOT (4/13)
APP configuration (LED_LAMP)

- Under “Dimming and Intensities Settings” tab
  - Set intensities of Channel 0 and Channel 1 to 2048
  - Set intensity of Channel 2 to 0



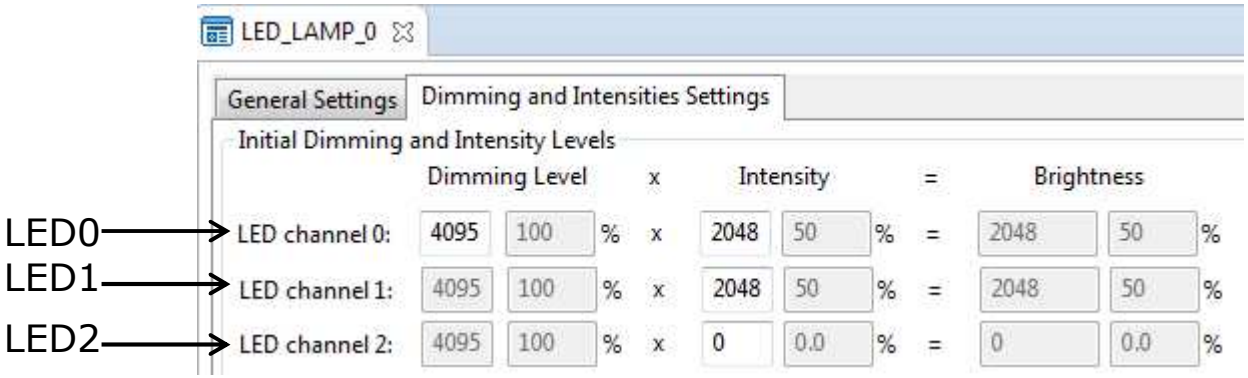
HOT (5/13)
Label re-assignment (PDM_BCCU)

- Assign PDM_BCCU APPs to the right channels
  - Hover mouse cursor over the connecting arrow to a PDM_BCCU APP
  - A label will appear momentarily e.g. LED0/LED1/LED2



HOT (6/13)
Label re-assignment (PDM_BCCU)

- The labels correspond to the LED channels in the UI

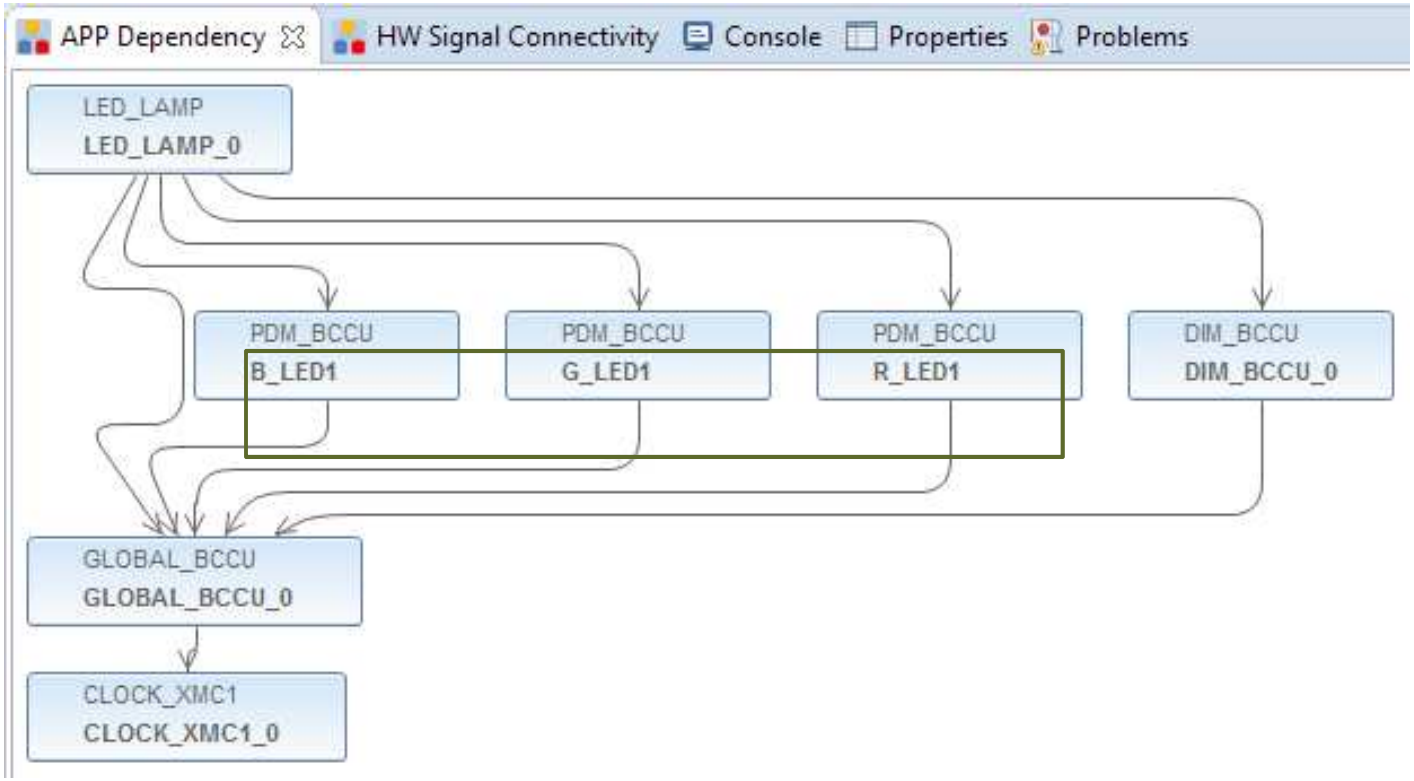


- Rename the PDM_BCCU instance label according to the table below
  - Right-click PDM_BCCU APP
  - Select "Rename Instance Label"

Label	New Label
LED0	R_LED1
LED1	G_LED1
LED2	B_LED1

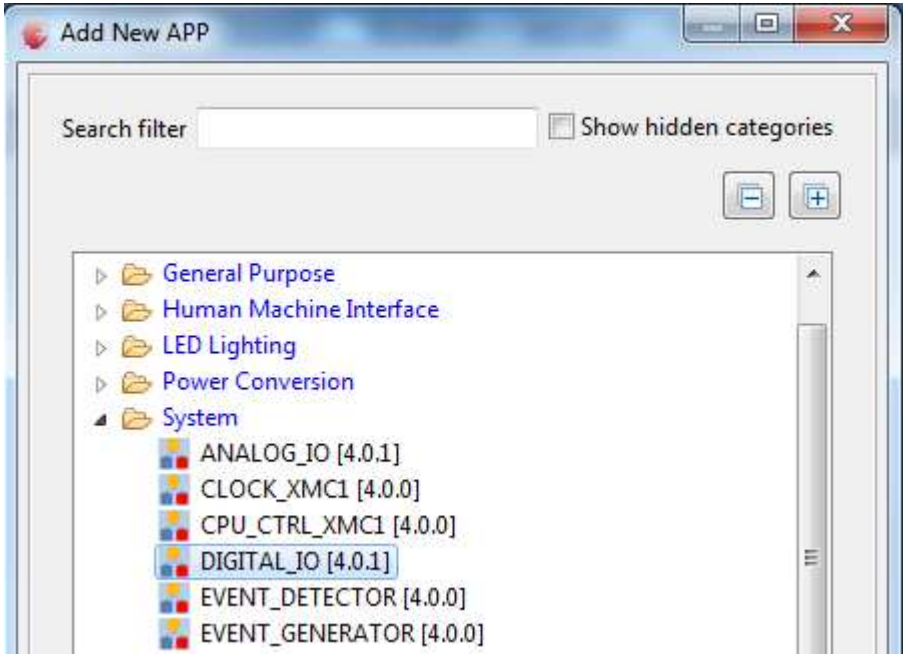
- Repeat the above steps with the other 2 PDM_BCCU APP instances

HOT (7/13)
Label re-assignment (PDM_BCCU)

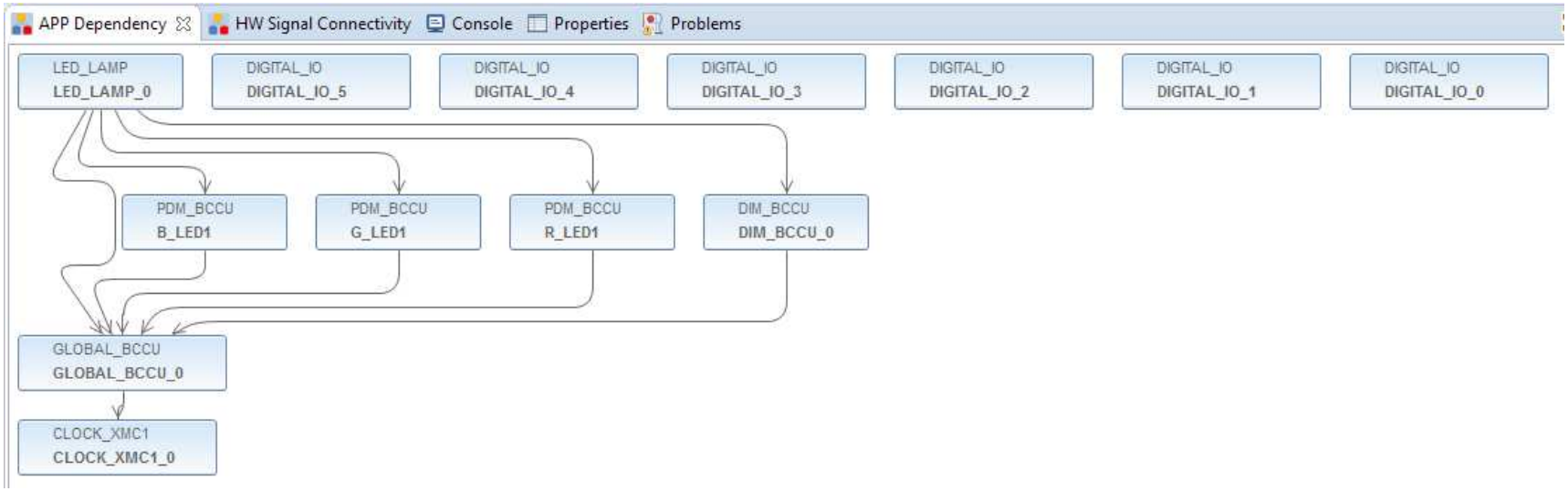


HOT (8/13)
APP configuration (DIGITAL_IO)

- Initialize unused pins to LED2 and LED3
  - Add 6 instances of DIGITAL_IO APPs



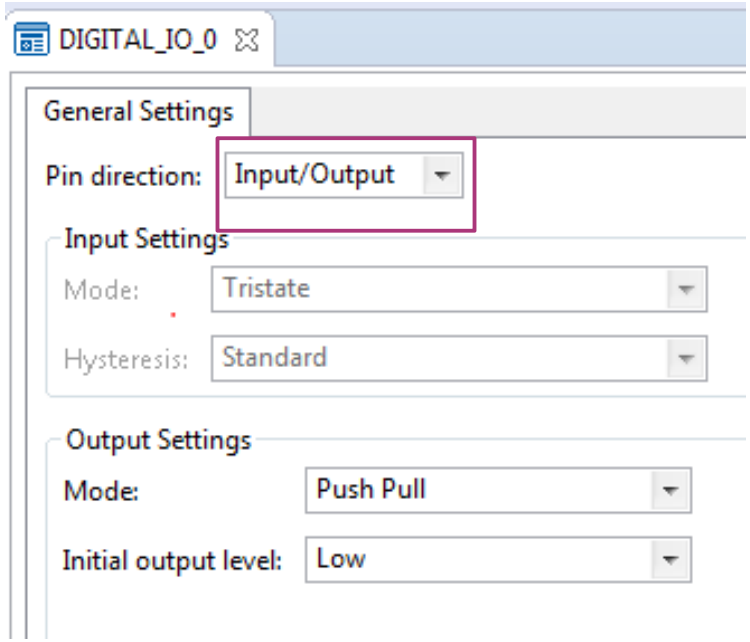
HOT (9/13)
APP configuration (DIGITAL_IO)



› Double-click a DIGITAL_IO APP instance to configure

HOT (10/13)
APP configuration (DIGITAL_IO)

› Configure the pin direction as “Input/Output”



› Repeat this configuration step for other 5 DIGITAL_IO APP instances

HOT (12/13)
Manual pin assignment (all)

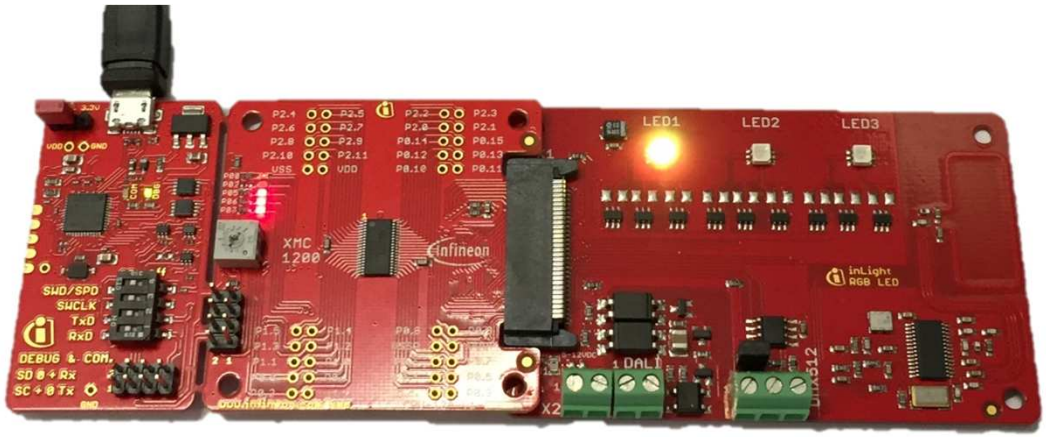
- Assign pins to LED1 and the unused LED2 and LED3
  - Click 
  - Assign pin as follows:

App Instance Name	App Pin Name	Pin Number (Port)
B_LED1		
	pin	#18 (P0.1)
DIGITAL_IO_0		
	pin	#22 (P0.5)
DIGITAL_IO_1		
	pin	#23 (P0.6)
DIGITAL_IO_2		
	pin	#24 (P0.7)
DIGITAL_IO_3		
	pin	#27 (P0.8)
DIGITAL_IO_4		
	pin	#28 (P0.9)
DIGITAL_IO_5		
	pin	#29 (P0.10)
G_LED1		
	pin	#30 (P0.11)
R_LED1		
	pin	#21 (P0.4)

  - Click “Solve and Save”

HOT (13/13)
Compile the project and observe LED1

- Generate code 
- Build project 
- Download code 
- Start code 
- Observe LED1 on Color LED card showing yellow light



HOT:
Changing color to blue

Objectives

- To demonstrate using the DAVE™ APPs to change the color of the LED to blue
  - This training is an extension of the previous HOT (yellow light)
  - LED will change color at start from yellow to blue in 7 s

Things you need for the training

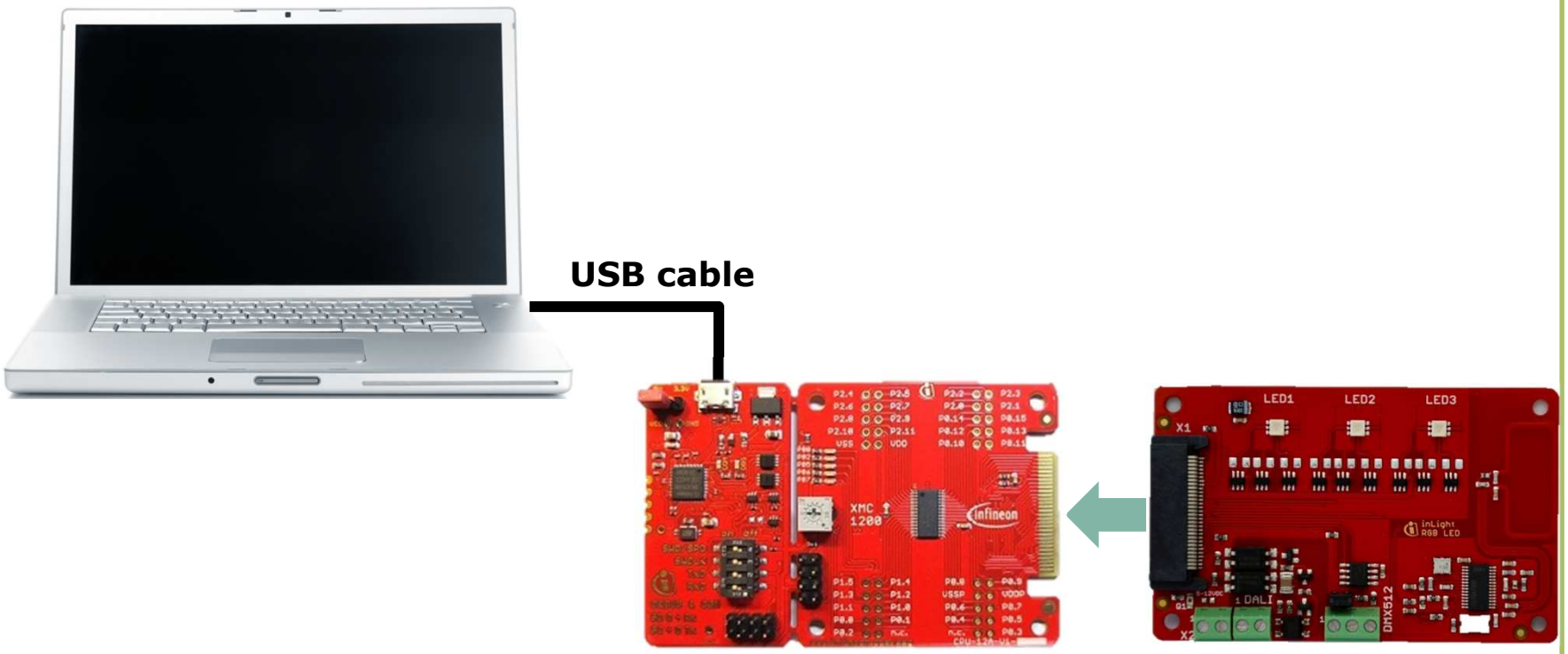
- Hardware
  - XMC1200 boot kit
  - Color LED card
    - Only LED1 will be used
  - PC
  - USB cable

- Software/Tools
  - DAVE™ Version 4
  - Previous HOT project (yellow light)

- **Things you must know before starting on this training**
  - We assume that you are equipped with the basic knowledge to create a DAVE™ Version 4 project, compile a software and enter into a debug environment

For more information on the kit and tool, please refer to General Information.

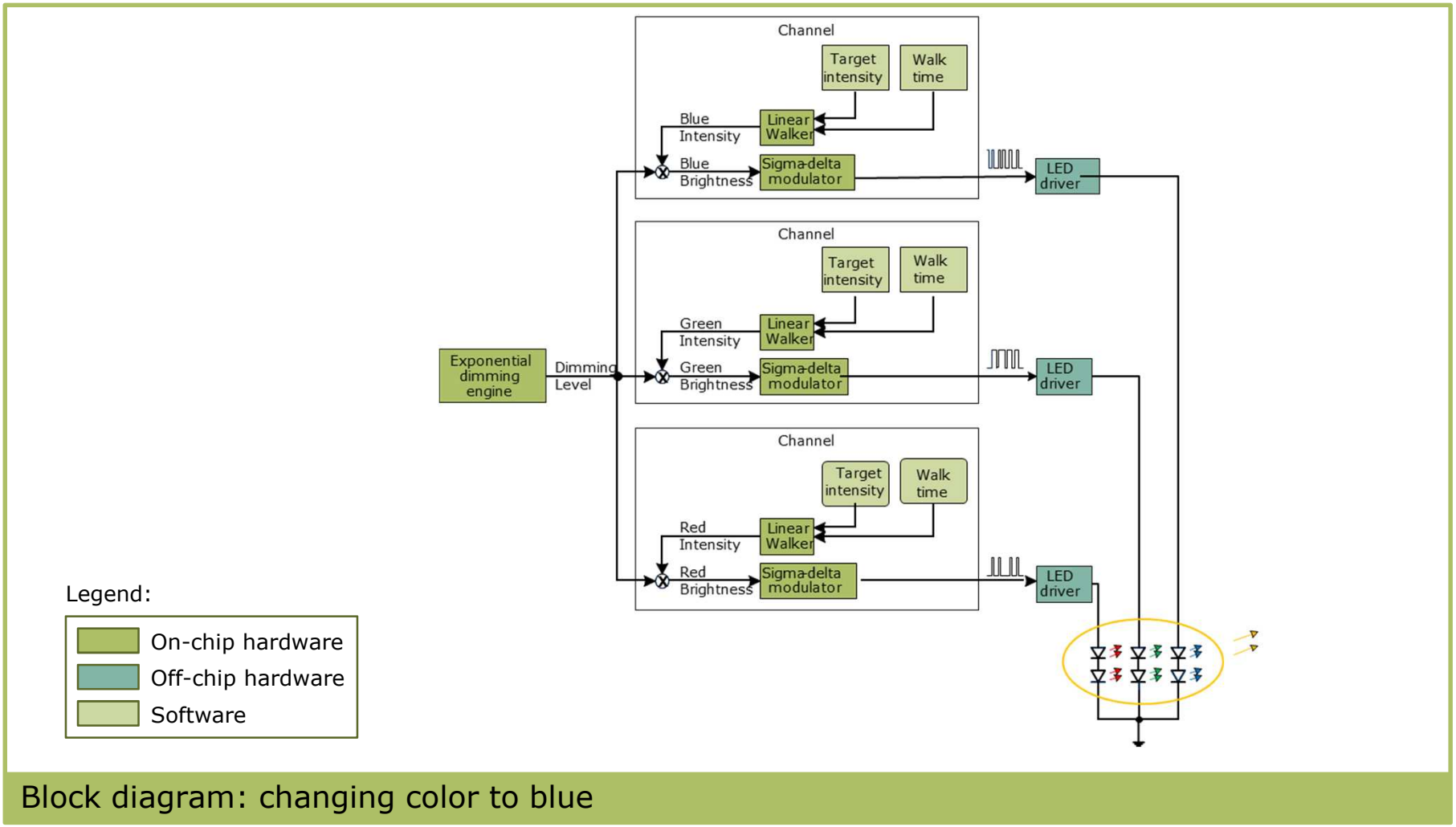
Hardware setup



1. Plug the Color LED card onto the XMC1200 boot kit.
2. Connect the debug USB of the XMC1200 boot kit to the PC using the USB cable.

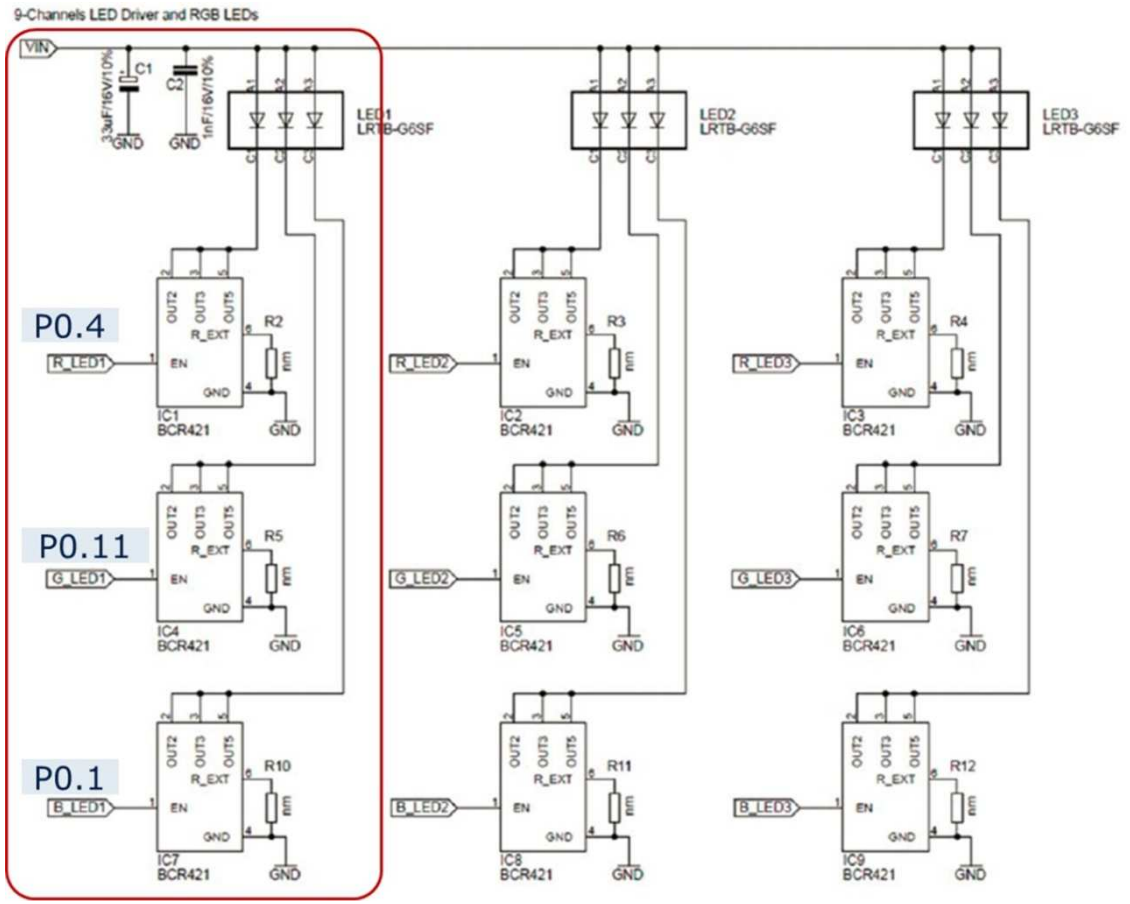
Hardware Setup: Changing Color to Blue

Peripheral configuration



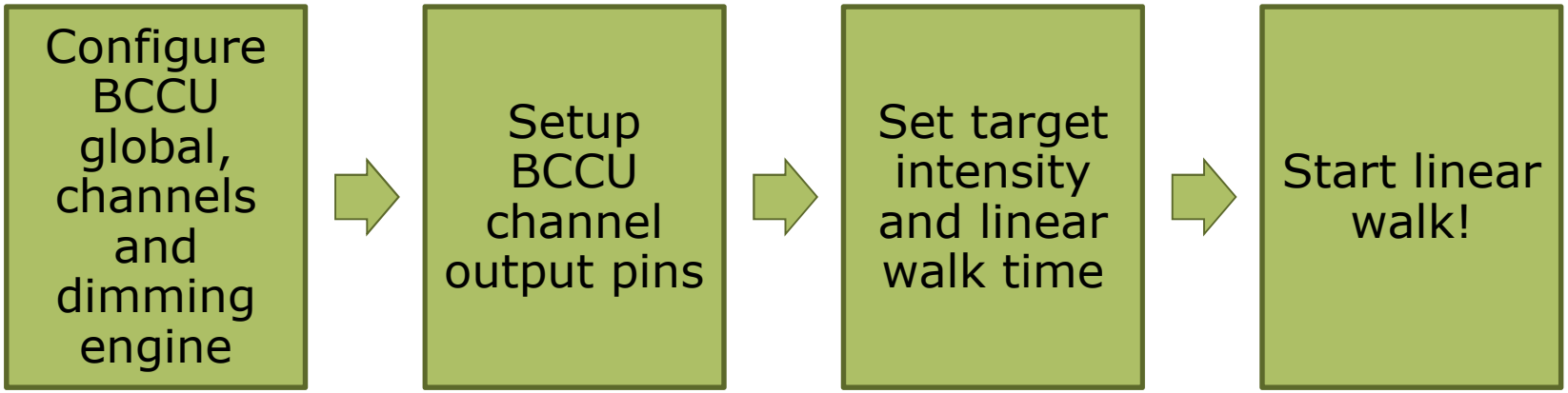
Block diagram: changing color to blue

Board schematics



Schematic: changing color to blue

Software overview



Flow chart: changing color to blue

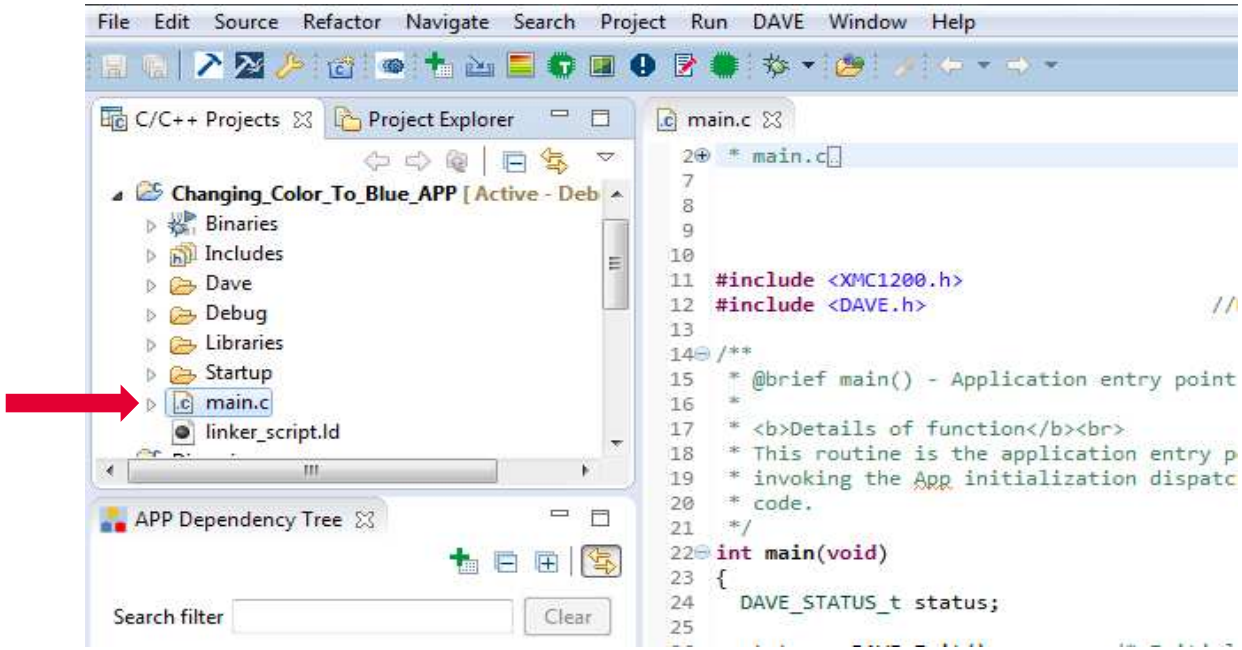
Software, list of DAVE™ APPs

- LED_LAMP
  - Configures the system and peripheral clocks
  - Configures BCCU global, channels and dimming engine
  - Provides control for channels and dimming engine

- DIGITAL_IO
  - Initializes IO pins that are connected to the RGB LEDs on board

HOT (1/7)
Open previous project

- Open project created in previous HOT (yellow light)
- Copy and paste project to same workspace
- Rename project
- Set project as Active
- Open main.c



HOT (2/7)
Software coding

- Add code to main.c
- Steps:
  1. Set channel intensities for blue light
  2. Determine linear walk time
  3. Start new linear walk

HOT (3/7)
Software coding

- LED_LAMP APP generates a configuration data structure "***config***"
  - Can be found in led_lamp_conf.c

led_lamp_conf.c

```
71  ************************************
72  #include "led_lamp.h"
73
74  LED_LAMP_CONFIG_t LED_LAMP_0_config =
75  {
76   .led_intensity = {2048,2048,0,
77                     0,0,0,
78                     0,0,0},
79
80  .dim_level = 4095
81
82  };
83
```

- "***config***" has 2 elements
  - led_intensity
    - An array holding configuration data for channel intensity
    - Each array element is for each of the 9 BCCU channels
      - led_intensity[0] for LED channel0, led_intensity[1] for LED channel1 etc.
  - dim_level
    - holds configuration data for dimming level

HOT (4/7)
Software coding

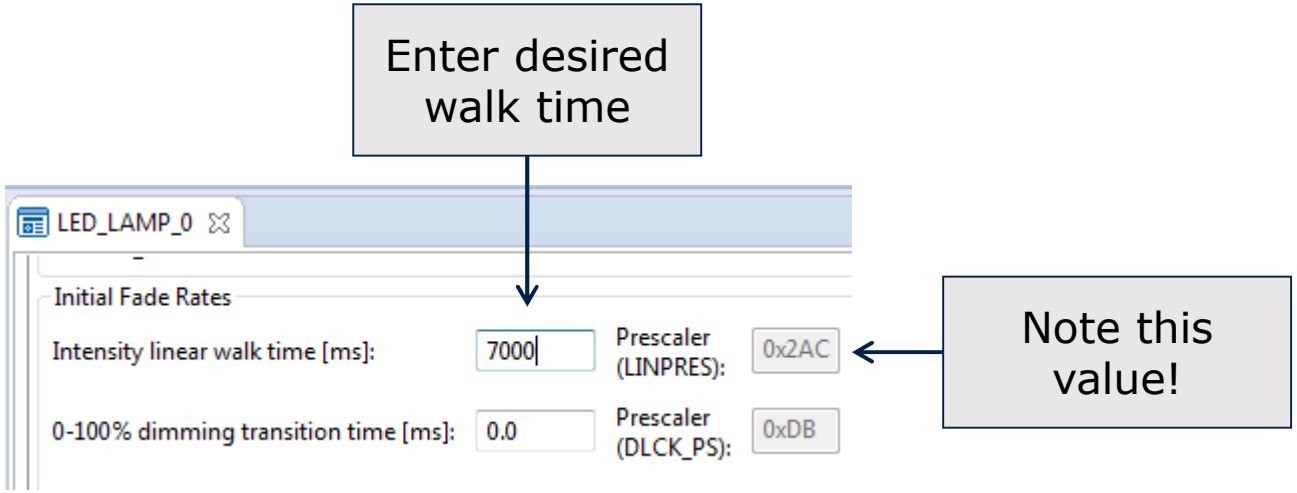
1. Set channel intensities for blue light

Desired color	RED channel intensity	GREEN channel intensity	BLUE channel intensity
Red	4095	0	0
Green	0	4095	0
Blue	0	0	4095
White	1365	1365	1365
Yellow	2048	2048	0

```
LED_LAMP_0.config->led_intensity[0] = 0;
LED_LAMP_0.config->led_intensity[1] = 0;
LED_LAMP_0.config->led_intensity[2] = 4095;
```

HOT (5/7)
Software coding

2. Determine linear walk time
   - Let's use a walk time of 7 s
   - To determine the linear walk time prescaler (LINPRES), open LED_LAMP APP UI
     - "Dimming and Intensities Settings" tab



Note: Set the linear walk time to its original value after you have obtained the LINPRES value.

HOT (6/7)
Software coding

3. Start linear walk
   - Use the obtained LINPRES value in code

```
LED_LAMP_SetColorAdv(&LED_LAMP_0, 0x2AC); // walk time = 7s
```

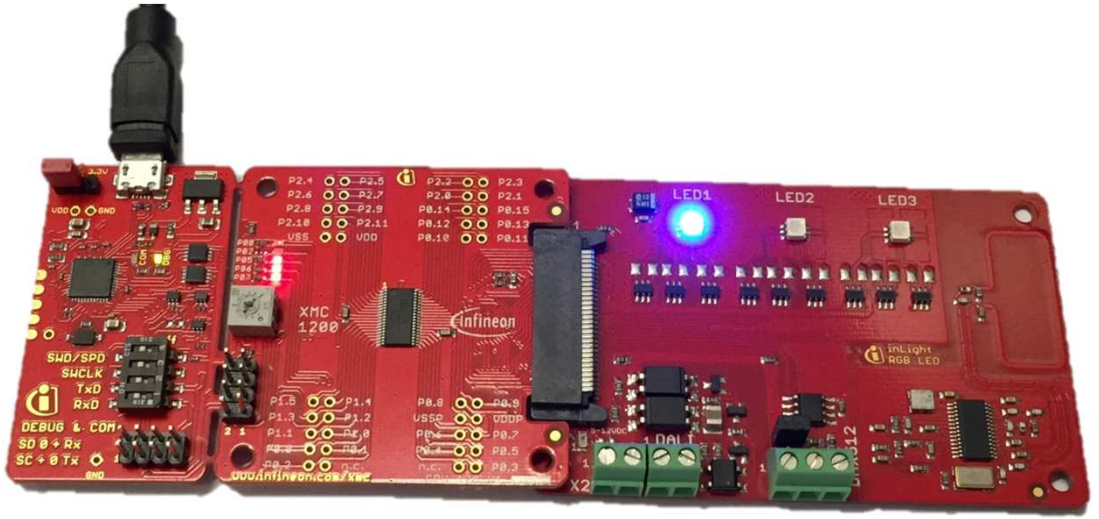
› There is also another API for starting linear walk

```
LED_LAMP_SetColor(&LED_LAMP_0);
```

Difference: this uses the linear walk time set in the LED_LAMP APP UI

HOT (7/7)
Compile the project and observe LED1

- Build project 
- Download code 
- Start code 
- Observe LED1 on Color LED card changing to blue



HOT:
Slow color-change sequence

Objectives

- To demonstrate using the BCCU with System Timer (SysTick) via DAVE™ APPs for a color-change sequence
  - This training is an extension of the previous HOT (changing color to blue)
  - Using SysTick to sequentially cycle LED color from white to magenta to cyan

Things you need for the training

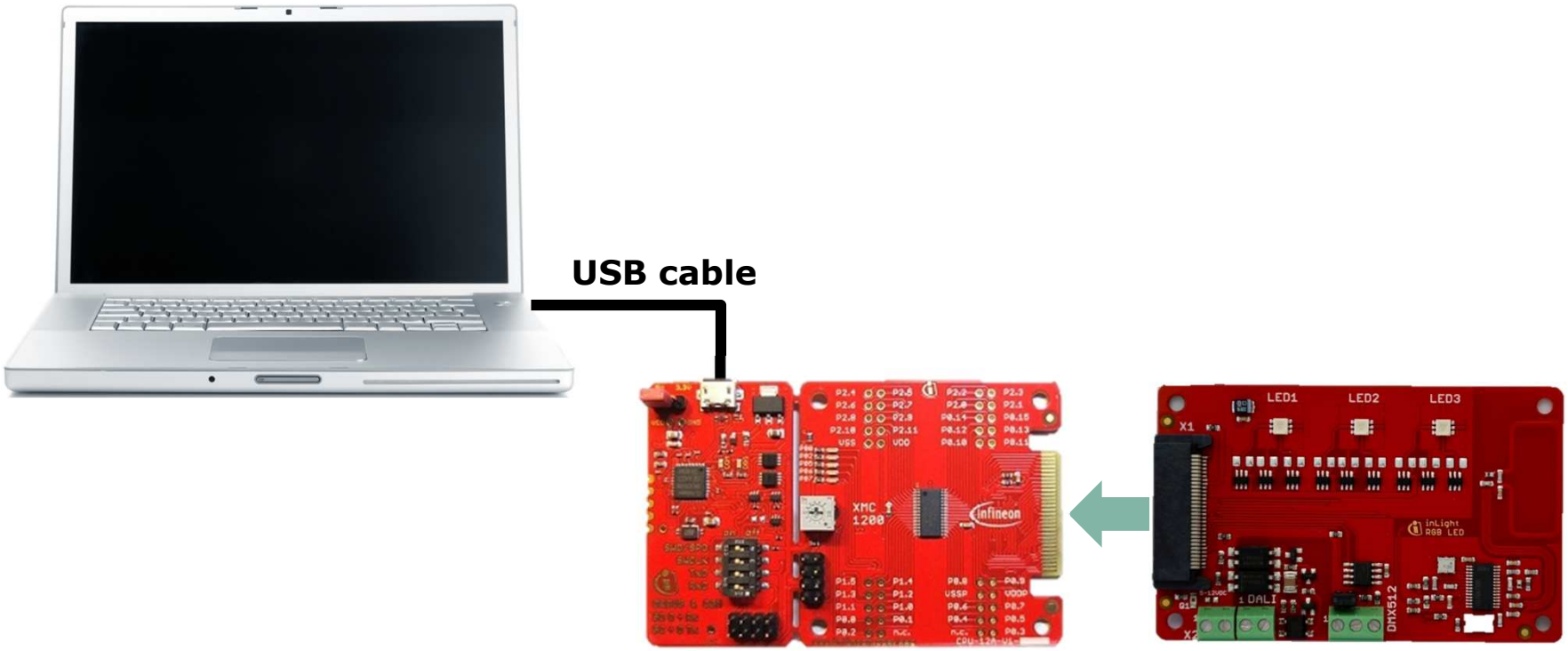
- Hardware
  - XMC1200 boot kit
  - Color LED Card
    - Only LED1 will be used
  - PC
  - USB cable

- Software/Tools
  - DAVE™ Version 4
  - Previous HOT project (changing color to blue)

- **Things you must know before starting on this training**
  - We assume that you are equipped with the basic knowledge to create a DAVE™ Version 4 project, compile a software and enter into a debug environment

For more information on the kit and tool, please refer to General Information.

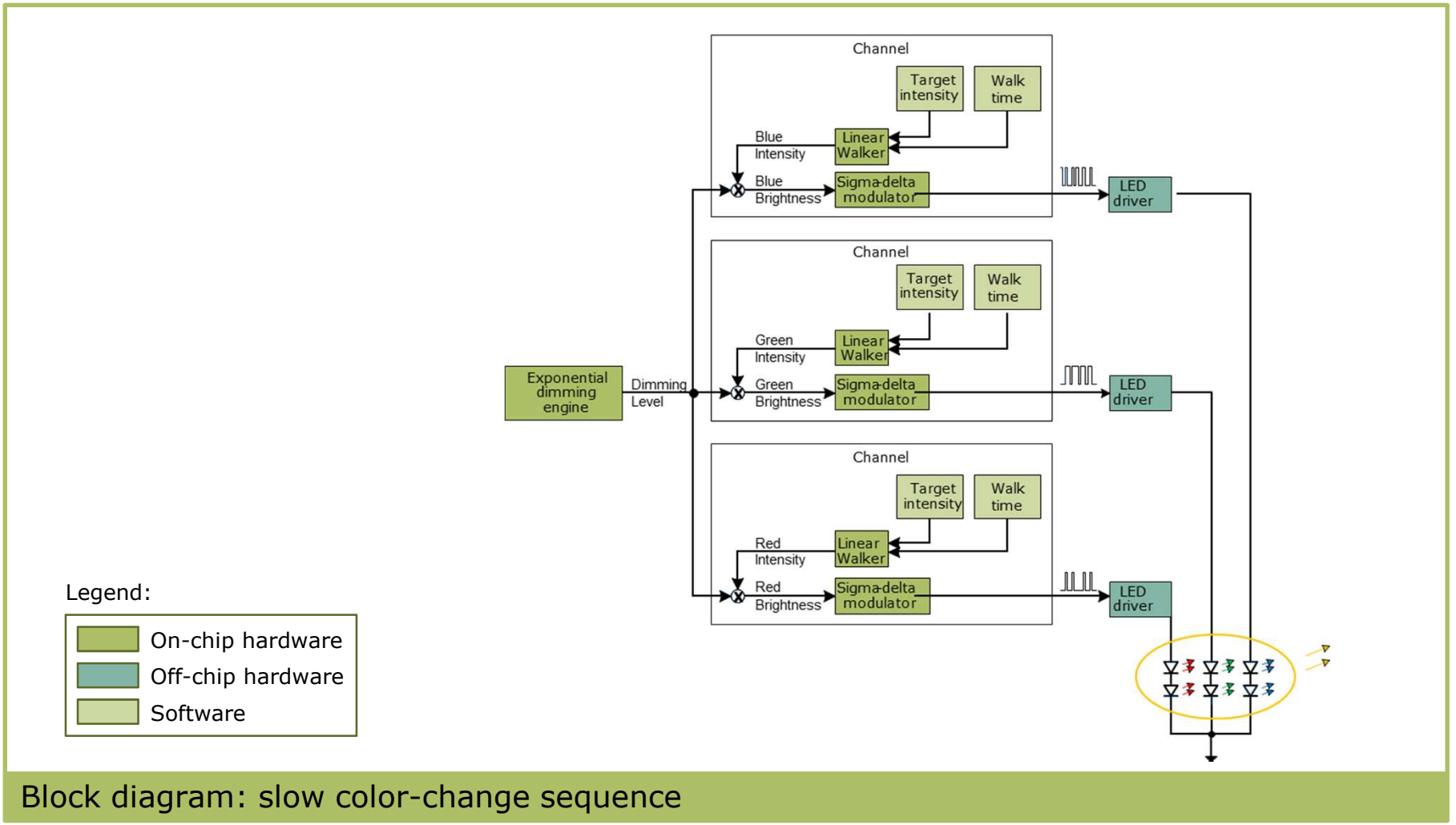
Hardware setup



1. Plug the Color LED Card onto the XMC1200 boot kit.
2. Connect the debug USB of the XMC1200 boot kit to the PC using the USB cable.

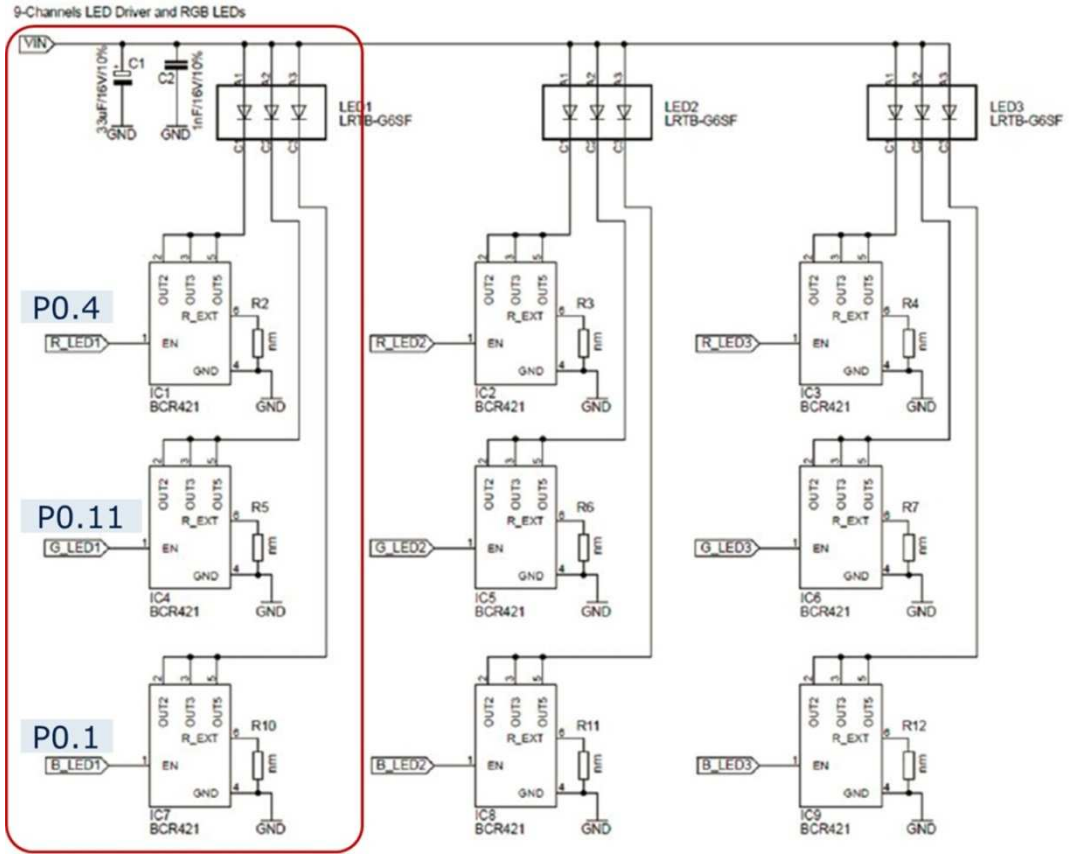
Hardware setup: slow color-change sequence

Peripheral configuration



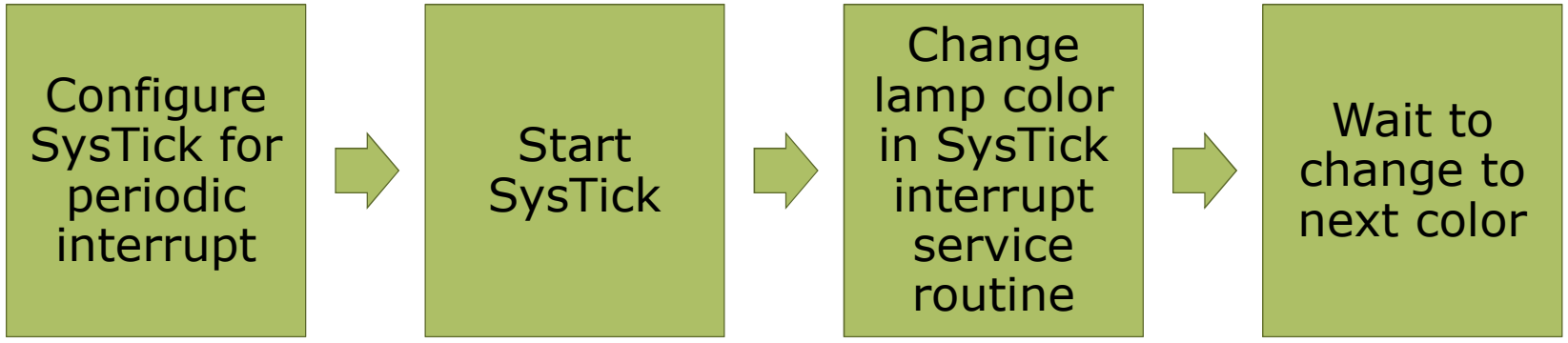
Block diagram: slow color-change sequence

Board schematics



Schematic: slow color-change sequence

Software overview



Flow chart: slow color-change sequence

Software, list of DAVE™ APPs

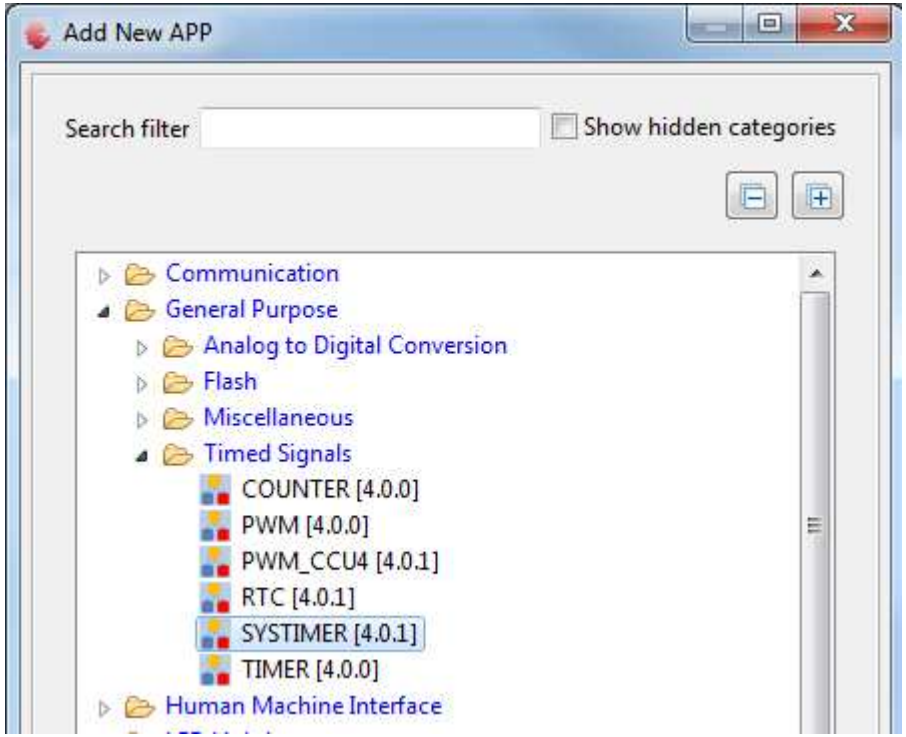
- LED_LAMP
  - Configures the system and peripheral clocks
  - Configures BCCU global, channels and dimming engine
  - Provides control for channels and dimming engine
- DIGITAL_IO
  - Initializes IO pins that are connected to the RGB LEDs on board
- SYSTIMER
  - Uses the SysTick interrupt to call user functions periodically at a specified rate

HOT (1/8)
Open previous project

- Open project created in previous HOT (changing color to blue)
- Copy and paste project to same workspace
- Rename project
- Set project as Active

HOT (2/8)
APP configuration (SYSTIMER)

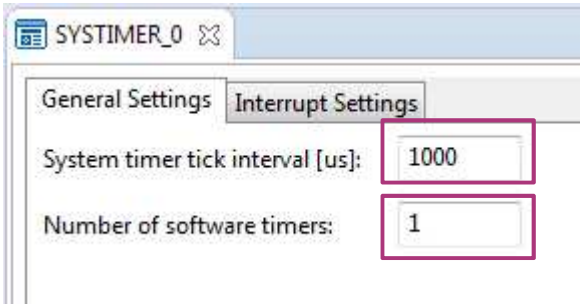
- Add SYSTIMER APP to project



- Double-click SYSTIMER APP to open the UI

HOT (3/8)
APP configuration (SYSTIMER)

- Set tick interval and number of software timers



- Generate code 

HOT (4/8)
Software coding

- Open main.c
- Steps:
  1. Configure software timer for 1s periodic interrupt
  2. Define timer interrupt handler to change LED color

HOT (5/8)
Software coding

1. Configure software timer for 1s periodic interrupt
   a. Create software timer

```
uint32_t timer_id;
timer_id = SYSTIMER_CreateTimer(1000000,SYSTIMER_MODE_PERIODIC,(void*)One_Sec_Intr,NULL);
```

   b. Start software timer

```
status = SYSTIMER_StartTimer(timer_id);
```

HOT (6/8)
Software coding

2. Define timer interrupt handler to change LED color
   a. Give the timer interrupt handler a name

      e.g. `void One_Sec_Intr(void)`

   b. Use a variable as a counter in the interrupt handler e.g. `uint8_t step`
      - `step` will increment every second
      - Assuming linear walk time of 7 s, we will change LED color every 8th second (i.e. `step = 8`)

   c. Code to change LED color is the same as in previous HOT

- See next slide for interrupt handler definition

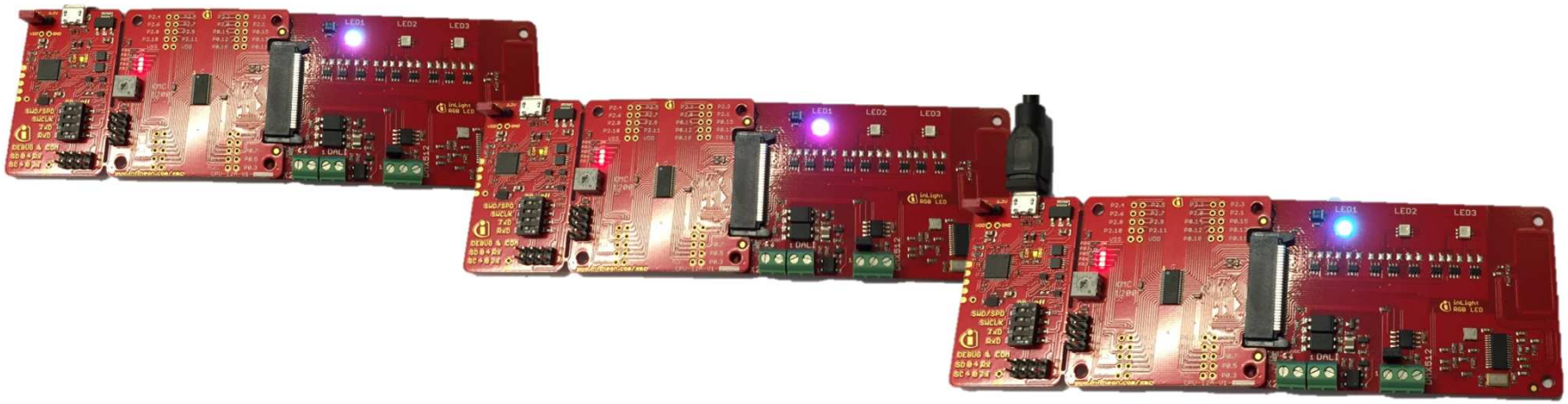
HOT (7/8)
Software coding

```
void One_Sec_Intr(void)
{
  static uint8_t step = 0;
  if(++step == 8)
  {
    LED_LAMP_0.config->led_intensity[0] = 1365;
    LED_LAMP_0.config->led_intensity[1] = 1365;
    LED_LAMP_0.config->led_intensity[2] = 1365;
    LED_LAMP_SetColorAdv(&LED_LAMP_0, 0x2AC);
  }
  if(step == 16)
  {
    LED_LAMP_0.config->led_intensity[0] = 2048;
    LED_LAMP_0.config->led_intensity[1] = 0;
    LED_LAMP_0.config->led_intensity[2] = 2048;
    LED_LAMP_SetColorAdv(&LED_LAMP_0, 0x2AC);
  }
  if(step == 24)
  {
    LED_LAMP_0.config->led_intensity[0] = 0;
    LED_LAMP_0.config->led_intensity[1] = 2048;
    LED_LAMP_0.config->led_intensity[2] = 2048;
    LED_LAMP_SetColorAdv(&LED_LAMP_0, 0x2AC);
    step = 0;
  }
}
```

Desired color	RED channel intensity	GREEN channel intensity	BLUE channel intensity
White	1365	1365	1365
Magenta	2048	0	2048
Cyan	0	2048	2048

HOT (8/8)
Compile the project and observe LED1

- Build project 
- Download code 
- Start code 
- Observe LED1 on Color LED card cycling from white to magenta to cyan



HOT:
Dimming

Objectives

- To demonstrate using DAVE™ APPs to configure the brightness of an LED
  - This training is an extension of the previous HOT (yellow light)
  - LED will dim up at start from 0 % to 100 % brightness level in 7 s

Things you need for the training

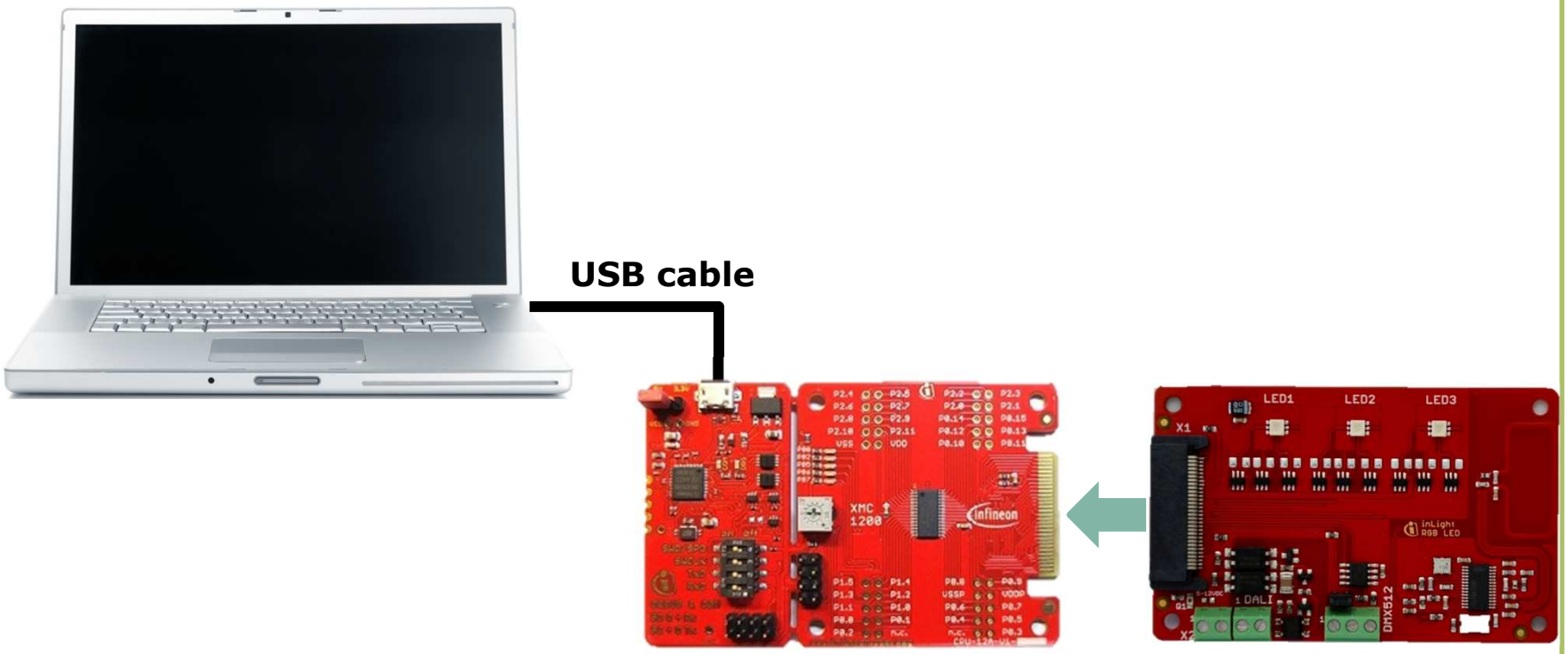
- Hardware
  - XMC1200 boot kit
  - Color LED card
    - Only LED1 will be used
  - PC
  - USB cable

- Software/Tools
  - DAVE™ Version 4
  - Previous HOT project (yellow light)

- **Things you must know before starting on this training**
  - We assume that you are equipped with the basic knowledge to create a DAVE™ Version 4 project, compile a software and enter into a debug environment

For more information on the kit and tool, please refer to General Information.

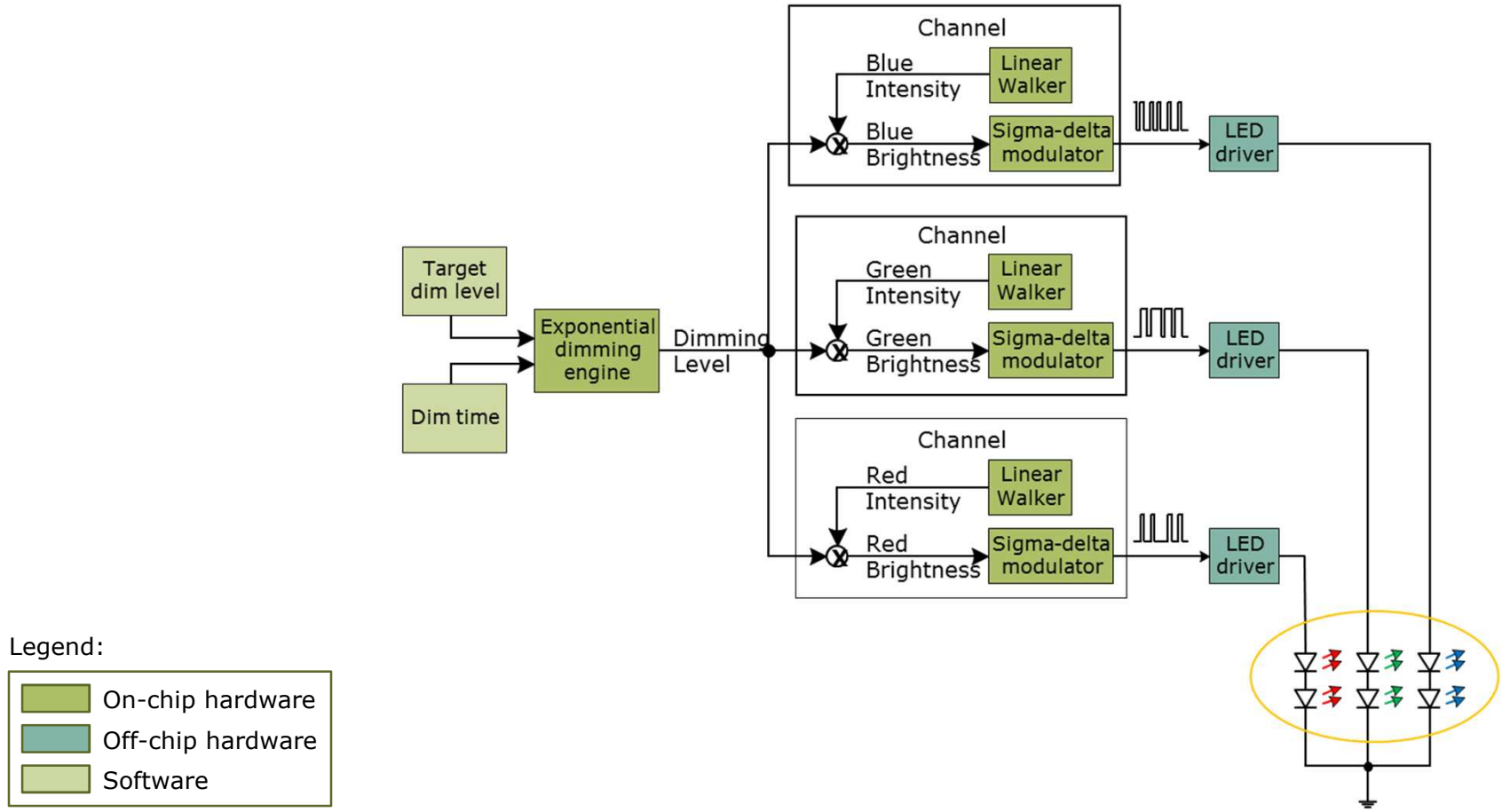
Hardware setup



1. Plug the Color LED Card onto the XMC1200 boot kit.
2. Connect the debug USB of the XMC1200 boot kit to the PC using the USB cable.

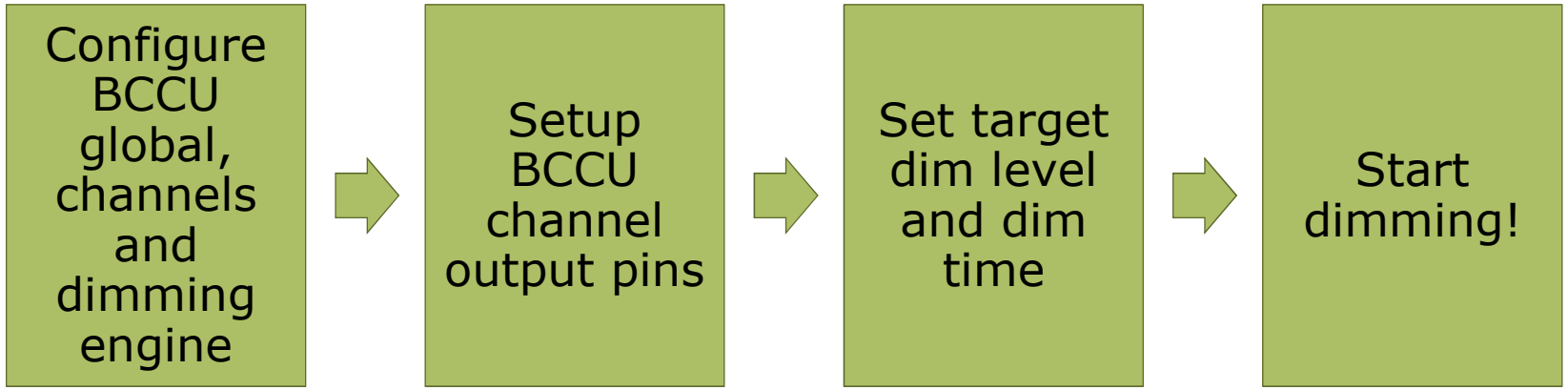
Hardware setup: dimming

Peripheral configuration



Block diagram: dimming

Software overview



Flow chart: dimming

Software, list of DAVE™ APPs

- LED_LAMP
  - Configures the system and peripheral clocks
  - Configures BCCU global, channels and dimming engine
  - Provides control for channels and dimming engine

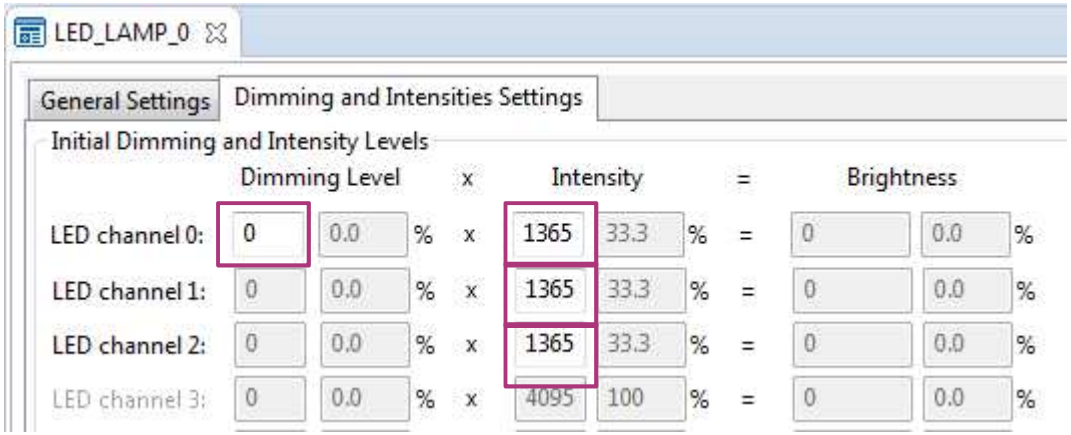
- DIGITAL_IO
  - Initializes IO pins that are connected to the RGB LEDs on board

HOT (1/7)
Open previous project

- Open project created in previous HOT (yellow light)
- Copy and paste project to same workspace
- Rename project
- Set project as Active

HOT (2/7)
APP configuration (LED_LAMP)

- Double-click LED_LAMP to open configuration UI
- Under "Dimming and Intensities Settings" tab
  - Set dimming level to 0
  - Set intensities of Channel 0, 1 and 2 to 1365
    - This will set the LED color to white



- Generate code 

HOT (3/7)
Software coding

- Open main.c
- Steps:
  1. Set desired dimming level
  2. Determine dimming clock prescaler (DCLK_PS) and divider (DIMDIV)
  3. Start dimming process

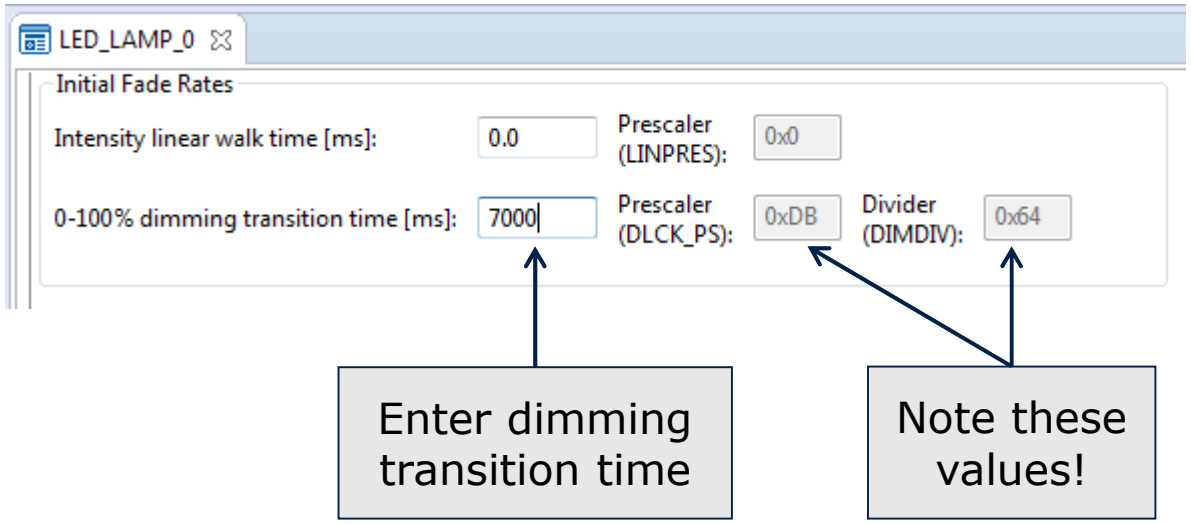
HOT (4/7)
Software coding

1. Set desired dimming level
   - **RECALL:** LED_LAMP generates a configuration data structure, "***config***"
     - One of its element is dim_level
       - Holds configuration data for dimming level of LED_LAMP
   - Let's set dimming level to 100 % (max)

```
LED_LAMP_0_config.dim_level = 4095;
```

HOT (5/7)
Software coding

2. Determine dimming clock prescaler (DCLK_PS) and divider (DIMDIV)
   - Let's use dimming transition time = 7 s
   - To determine DCLK_PS and DIMDIV, open LED_LAMP APP UI
     - "Dimming and Intensities Settings" tab



Note: Set the dimming transition time to its original value after you have obtained the DCLK_PS and DIMDIV values.

HOT (6/7)
Software coding

3. Start dimming process
   - Use the obtained DCLK_PS and DIMDIV values

```
LED_LAMP_SetDimLevelExponentialAdv(&LED_LAMP_0, 0x64, 0xDB);
```

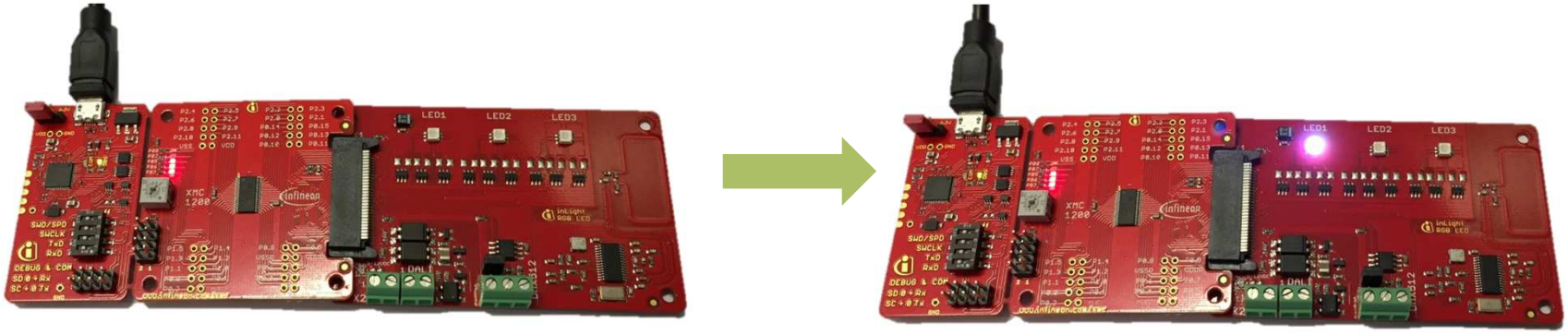
   - Dimming process can also be started using another API

```
LED_LAMP_SetDimLevelExponential(&LED_LAMP_0);
```

Difference: This API will use the dimming transition time configured in the LED_LAMP APP UI.

HOT (7/7)
Compile the project and observe LED1

- Build project 
- Download code 
- Start code 
- Observe LED1 on Color LED card dim up from 0 % to 100 % in 7 s



HOT:
Flicker!

Objectives

- To demonstrate how the BCCU can eliminate flickering from LED light
  - This training is an extension of previous HOT (yellow light)
  - This training will be broken down to 3 small parts:
    1. Demonstrate visible flickering
    2. Demonstrate the function of flicker watchdog to reduce flickering
    3. Obtain flicker-free light, even when viewed through a camera

Things you need for the training

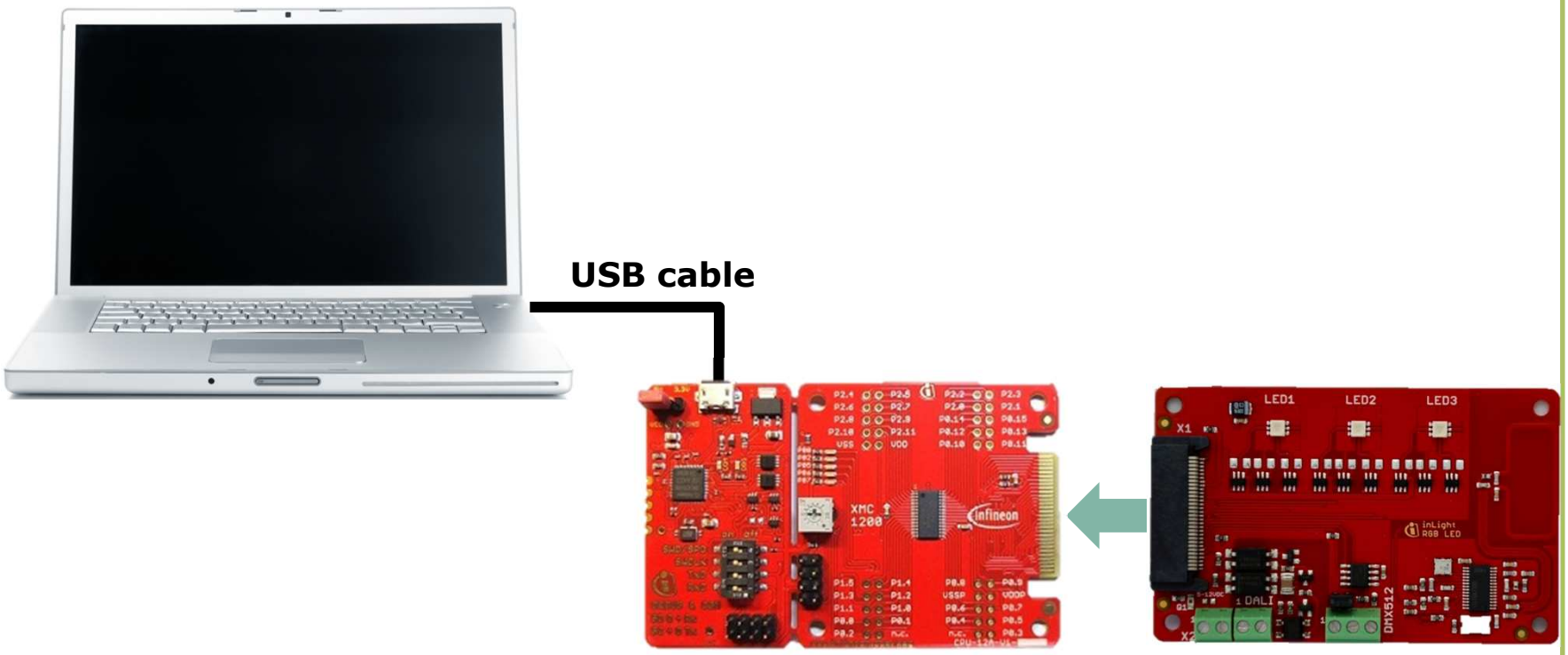
- Hardware
  - XMC1200 boot kit
  - Color LED card
    - Only LED1 will be used
  - PC
  - USB cable

- Software/Tools
  - DAVE™ Version 4
  - Previous HOT project (yellow light)

- **Things you must know before starting on this training**
  - We assume that you are equipped with the basic knowledge to create a DAVE™ Version 4 project, compile a software and enter into a debug environment

For more information on the kit and tool, please refer to General Information.

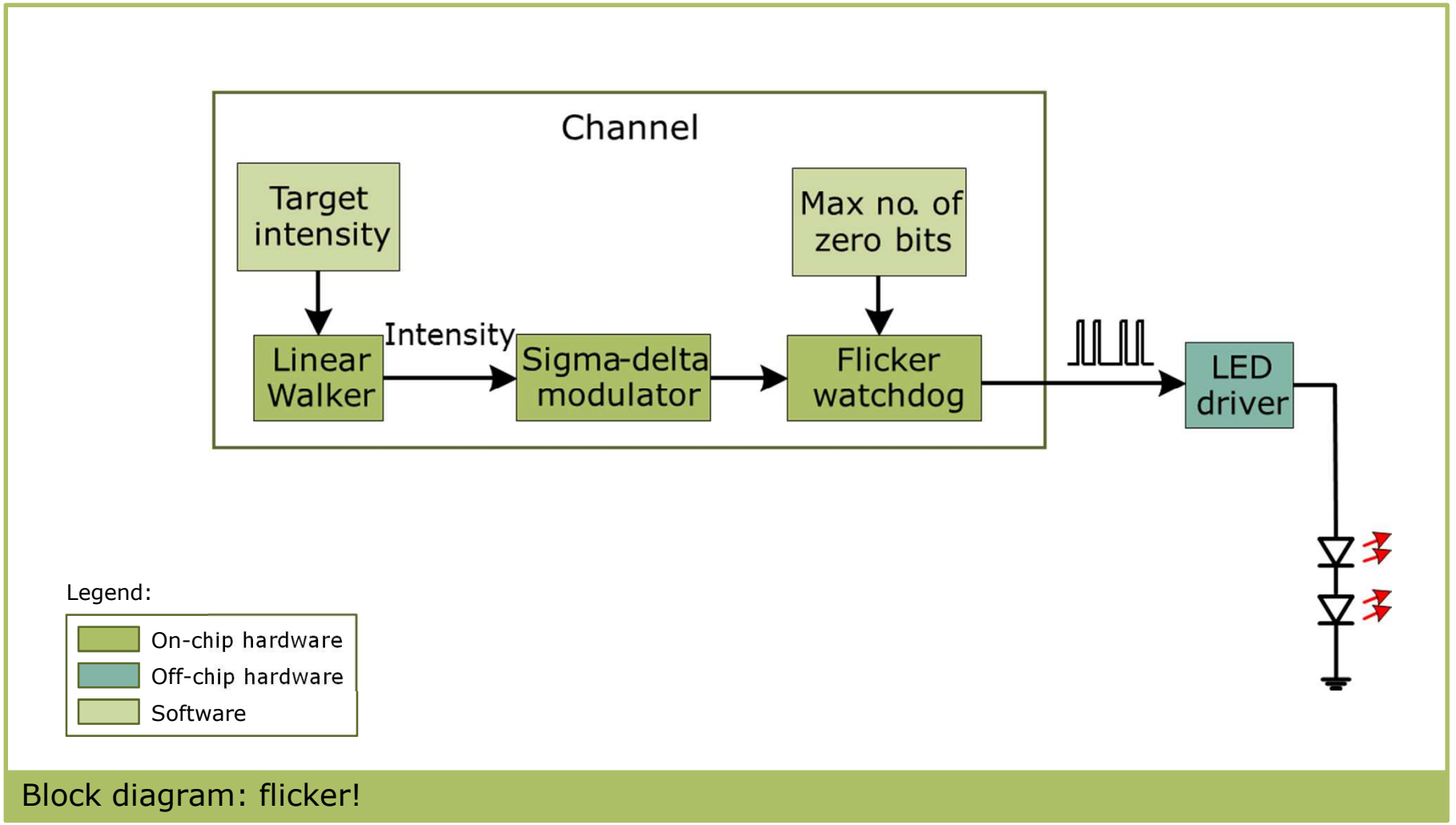
Hardware setup



1. Plug the Color LED card onto the XMC1200 boot kit.
2. Connect the debug USB of the XMC1200 boot kit to the PC using the USB cable.

Hardware setup: flicker!

Peripheral configuration



Block diagram: flicker!

Software, list of DAVE™ APPs

- LED_LAMP
  - Configures the system and peripheral clocks
  - Configures BCCU global, channels and dimming engine
  - Provides control for channels and dimming engine

- DIGITAL_IO
  - Initializes IO pins that are connected to the RGB LEDs on board

HOT (1/15)
Open previous project

- Open project created in previous HOT (yellow light)
- Copy and paste project to same workspace
- Rename project
- Set project as Active

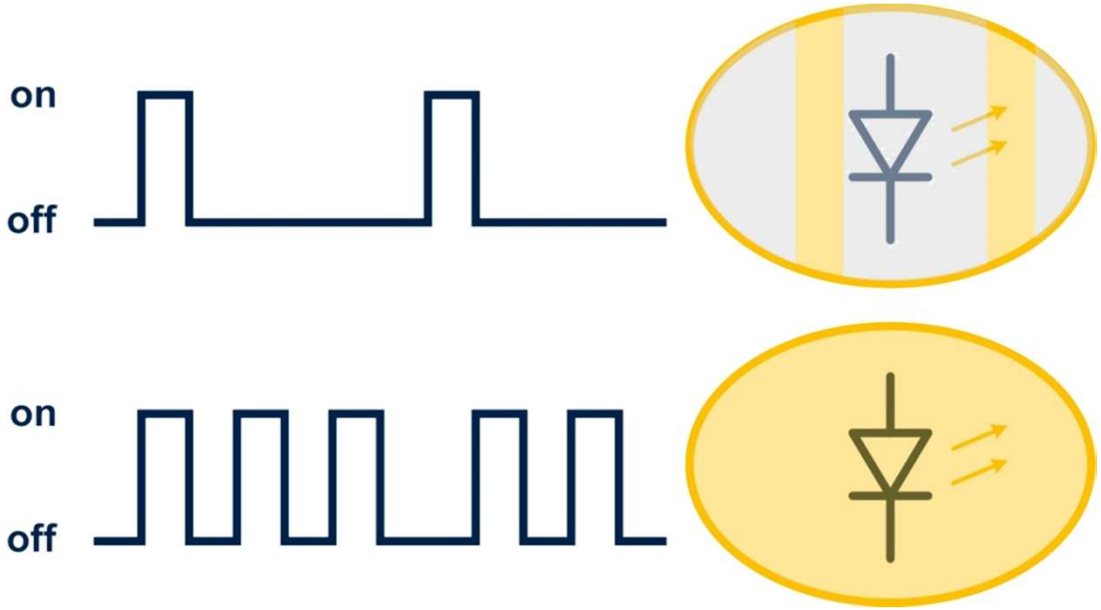
HOT (2/15)
Part 1: demonstrate visible flickering

1. Demonstrate visible flickering
   - Reconfigure GLOBAL_BCCU and LED_LAMP APP
   - Steps:
     a) Reduce bit clock (BCCU_bclk) frequency
     b) Change initial dimming level
     c) Change initial light color to white

HOT (3/15)
BCCU bit clock

a) Reduce BCCU_bclk frequency

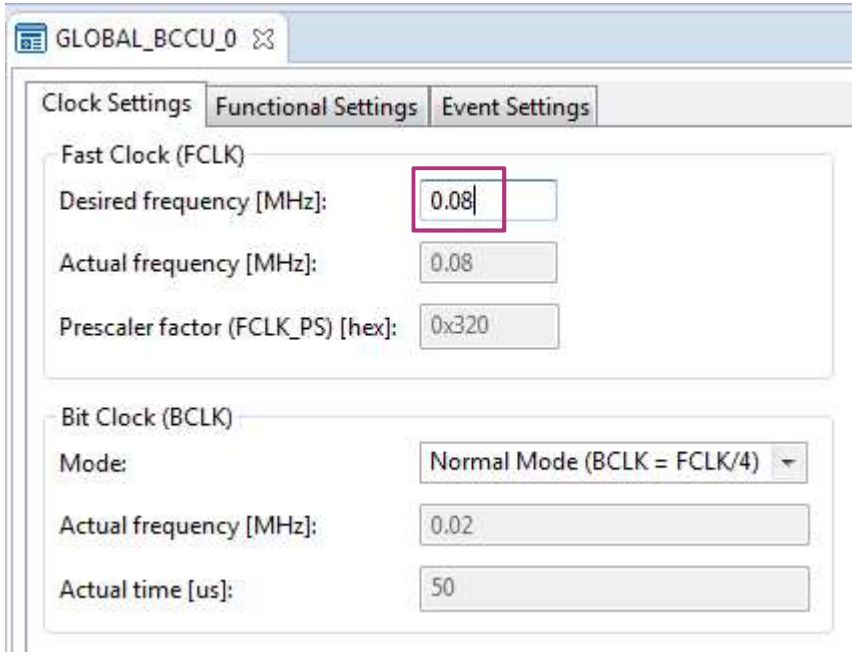
- BCCU_bclk determines the frequency of the modulation signal generated
- Slower frequency = longer off time → visible flickering



HOT (4/15)
Reconfigure APP (GLOBAL_BCCU)

a) Reduce BCCU_blck frequency (continued)

- Frequency of BCCU_bclk can be reduced by reducing the frequency of BCCU_fclk
- Change BCCU_fclk to 80 kHz → this changes BCCU_bclk to 20 kHz (bit time = 50 µs)
- Open GLOBAL_BCCU APP UI
  - "Clock Settings" tab
- Change Fast Clock frequency to 80 kHz

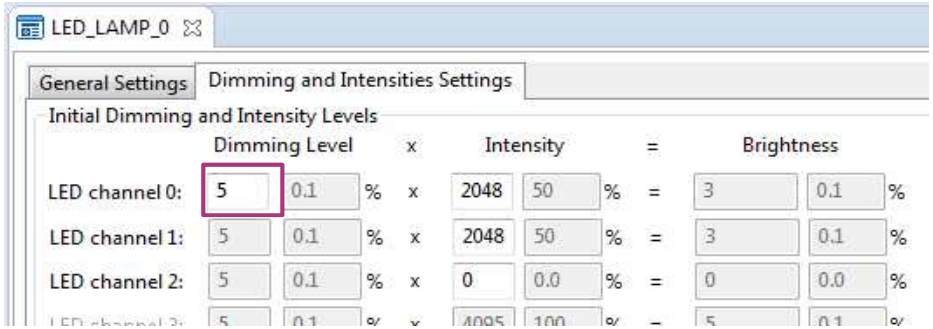


HOT (5/15)
Reconfigure APP (LED_LAMP)

b) Change initial dimming level

- At low dimming levels, the frequency of the modulation signal is even lower → higher chance of flickering
- In this training, we will use dimming level of 5

- Open LED_LAMP APP UI
  - "Dimming and Intensities Settings" tab
- Set dimming level to 5

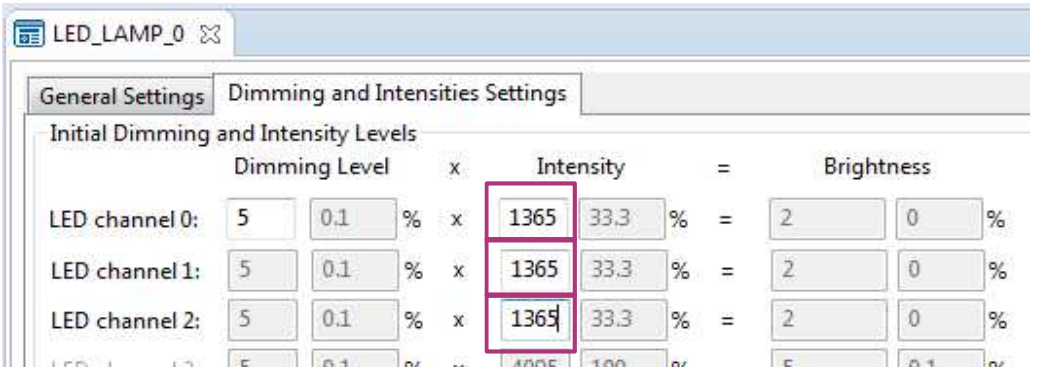


HOT (6/15)
Reconfigure APP (LED_LAMP)

c) Change initial light color to white

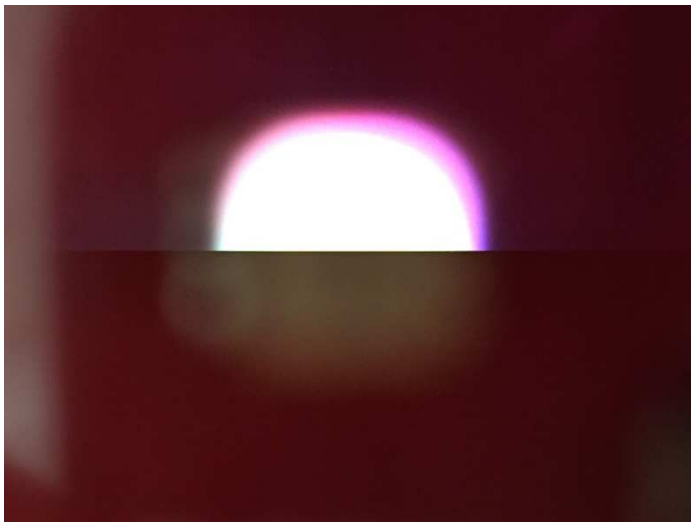
Desired color	RED channel intensity	GREEN channel intensity	BLUE channel intensity
Red	4095	0	0
White	1365	1365	1365
Yellow	2048	2048	0

– Set channel intensities to 1365



HOT (7/15)
Compile the project and observe LED1

- Generate code 
- Build project 
- Download code 
- Start code 
- Observe LED1 on Color LED card flickering



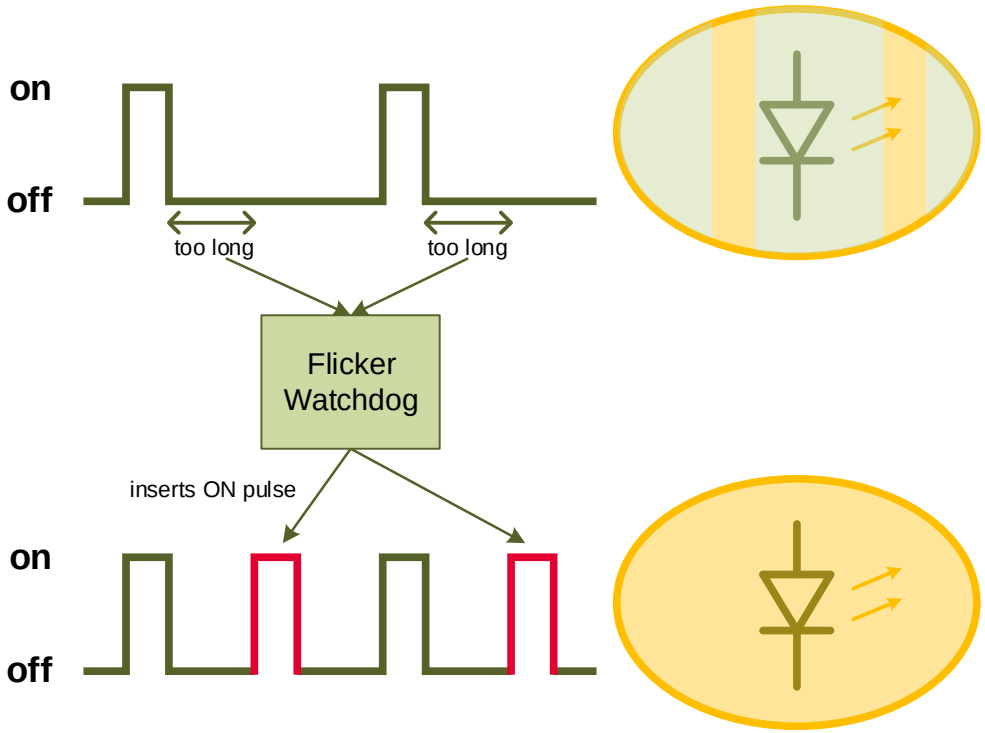
HOT (8/15)
Part 2: demonstrate flicker watchdog

2. Demonstrate the function of flicker watchdog to reduce flickering
   - Steps:
     a) Enable flicker watchdog in each channel
     b) Configure the threshold for the flicker watchdog

HOT (9/15)
Flicker watchdog

a) Enable flicker watchdog in each channel

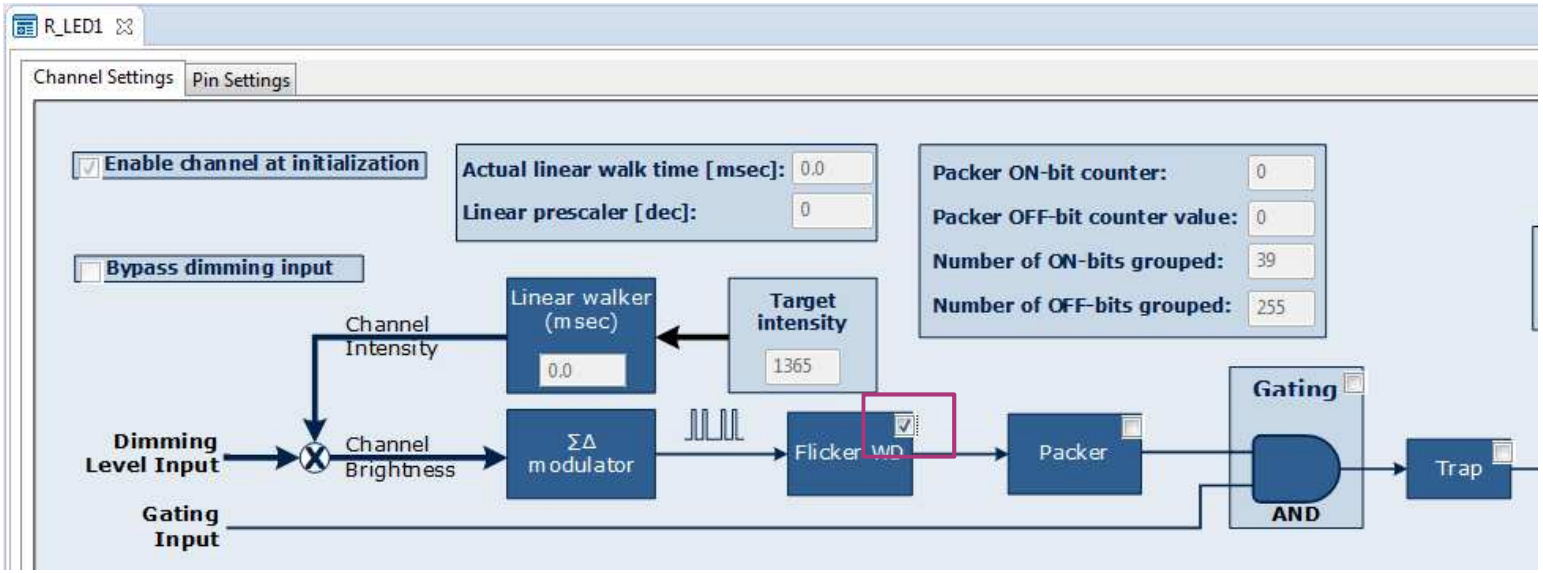
- Actively monitors the generated modulation signal for long periods of OFF
- Inserts an ON pulse in these conditions before the onset of flicker



HOT (10/15)
Reconfigure APP (PDM_BCCU)

a) Enable flicker watchdog in each channel (continued)

- Open the UI to a PDM_BCCU APP instance
  - "Channel Settings" tab
- Enable Flicker WD

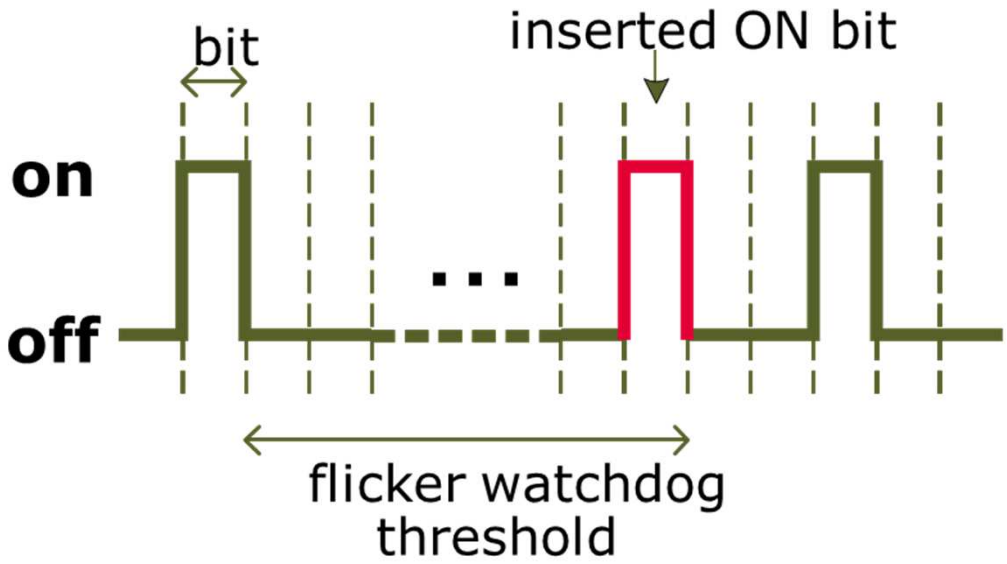


- Repeat for other 2 PDM_BCCU APP instances

HOT (11/15)
Flicker watchdog threshold

b) Configure the threshold for the flicker watchdog

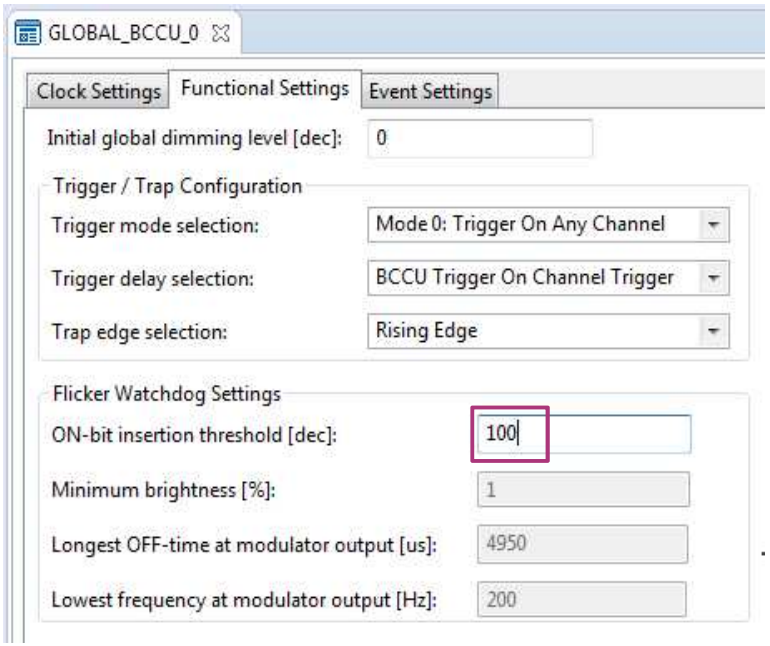
- No. of OFF bits before an ON bit is inserted
- In this example, we shall set this to 100 bits
  - An ON bit inserted for every period of 99 OFF bits
  - Minimum brightness limited to 41 (1 %)



HOT (12/15)
Reconfigure APP (GLOBAL_BCCU)

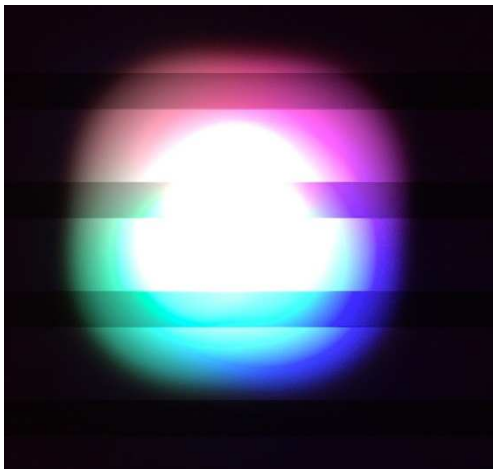
b) Configure the threshold for the flicker watchdog (continued)

- Open GLOBAL_BCCU APP UI
  - "Functional Settings" tab
- Set ON-bit insertion threshold to 100



HOT (13/15)
Compile the project and observe LED1

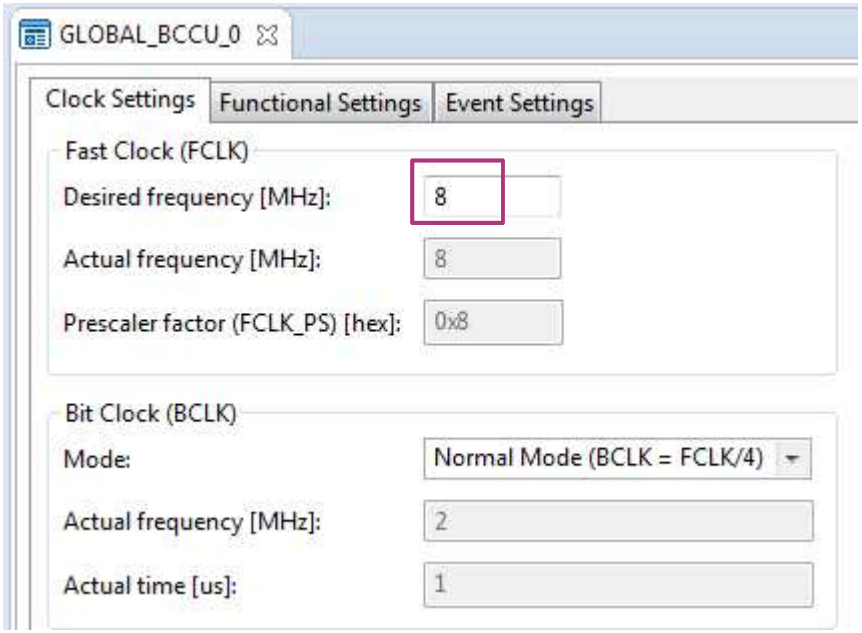
- Generate code 
- Build project 
- Download code 
- Start code 
- Observe LED1 on Color LED card
  - Flickering no longer visible to naked eye
  - Still visible when viewed with a camera



HOT (14/15)
Part 3: Flicker-free light

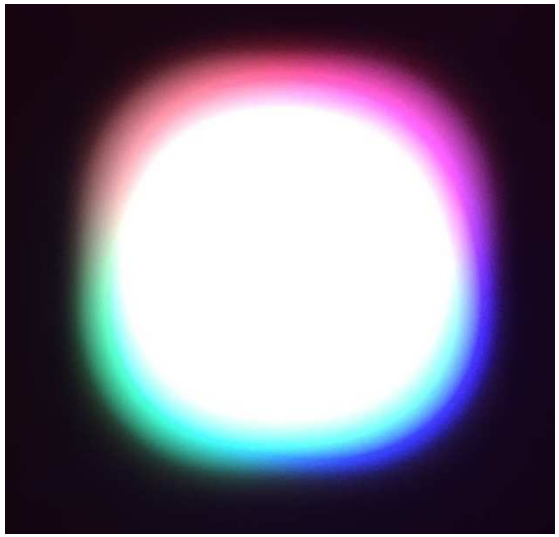
3. Obtain flicker-free light, even when viewed through a camera
   - Increase the frequency of BCCU_bclk
   - In this part, we will increase the frequency to 2 MHz

   - Open GLOBAL_BCCU APP UI
     - "Clock Settings" tab
   - To obtain BCLK = 2 MHz, increase FCLK to 8 MHz



HOT (15/15)
Compile the project and observe LED1

- Generate code 
- Build project 
- Download code 
- Start code 
- Observe LED1 on Color LED card
  - Flickering no longer visible when viewed from camera



General information

- For latest updates, please refer to:

  www.infineon.com/xmc1000

- For support:

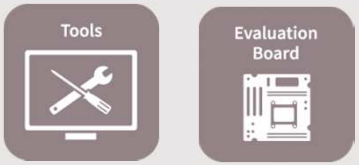
  http://www.infineonforums.com/forums/8-XMC-Forum

Resource listing

- LED lighting application kit

  www.infineon.com/xmc-dev

Support material

Collaterals and Brochures 	› Product Briefs › Selection Guides › Application Brochures › Presentations › Press Releases, Ads	› www.infineon.com/XMC
Technical Material 	› Application Notes › Technical Articles › Simulation Models › Datasheets, MCDS Files › PCB Design Data	› www.infineon.com/XMC › Kits and Boards › DAVE™ › Software and Tool Ecosystem
Videos 	› Technical Videos › Product Information Videos	› Infineon Media Center › XMC Mediathek
Contact 	› Forums › Product Support	› Infineon Forums › Technical Assistance Center (TAC)

Disclaimer

The information given in this training materials 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.

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 training material.

All the images used in the trainings are free for commercial use or free for use with attribution and were designed by Freepik.

Part of your life. Part of tomorrow.

infineon