# Application - digital multiplex (DMX512) receiving device

XMC™ microcontrollers
September 2016

# Agenda

# Agenda

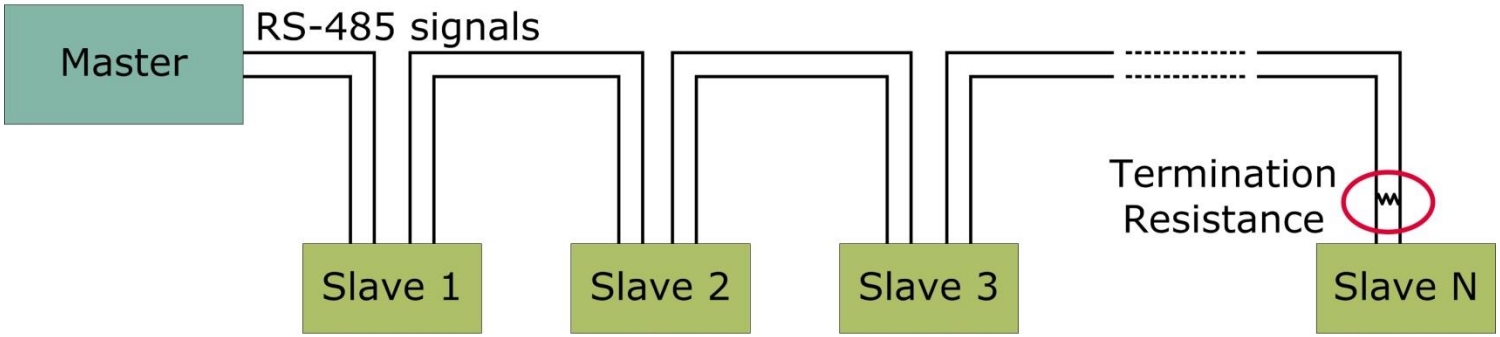
# DMX512 receiving device – Key features

Target application

- DMX512 receiving device

Key features

- RS-485
- Daisy chain
  - Single master (transmitting device)
  - Up to 32 slaves per branch (receiving device)



# Agenda

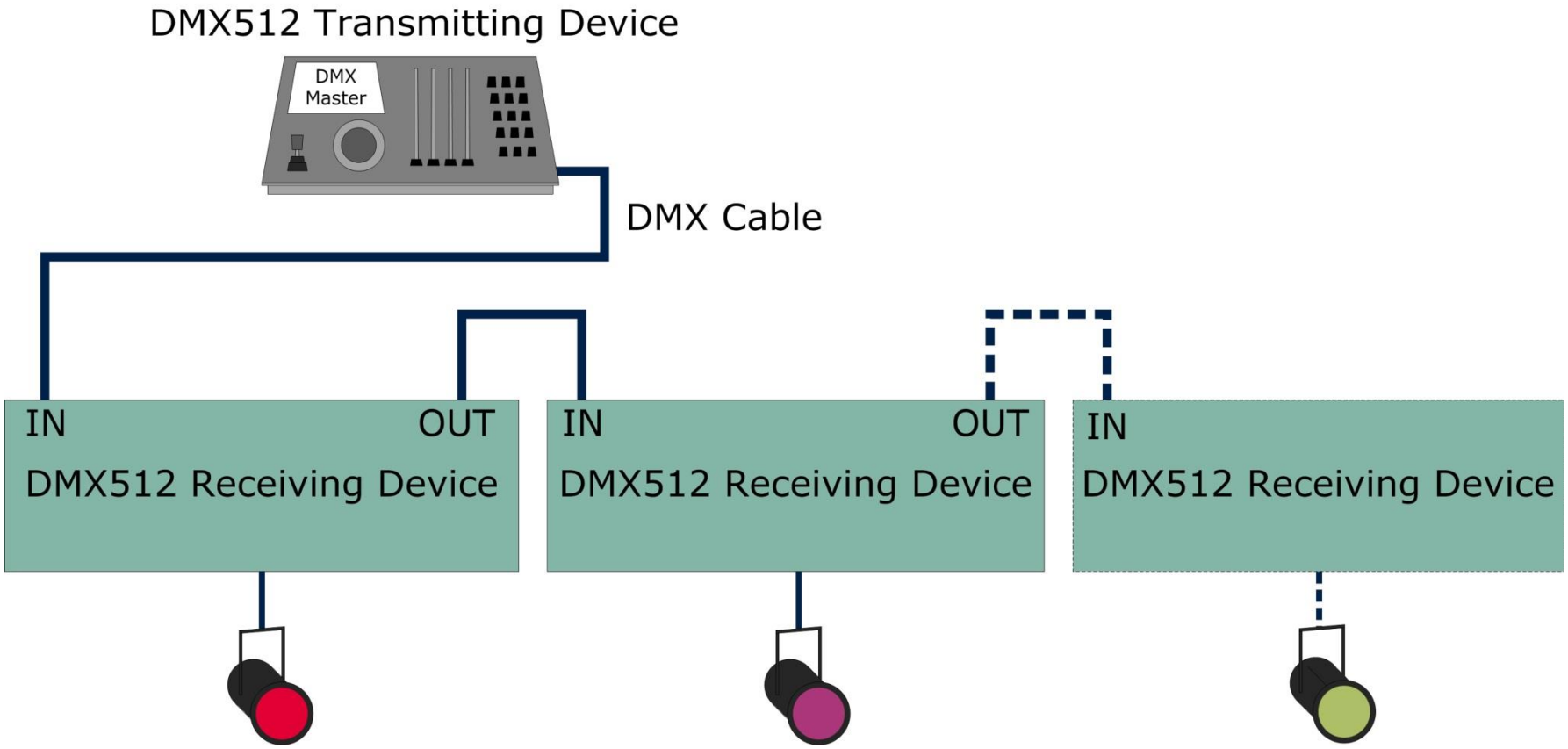
# DMX512 receiving device – Specification

Specifications

- Connectors: 5-pin XLR
  - Often 3-pin XLR is used
- Cable: twisted-pair, shielded, low-capacitance data cable
- Data: 250 kHz transmission

# Agenda

# DMX512 receiving device – System block diagram




System block diagram: DMX system

# Agenda

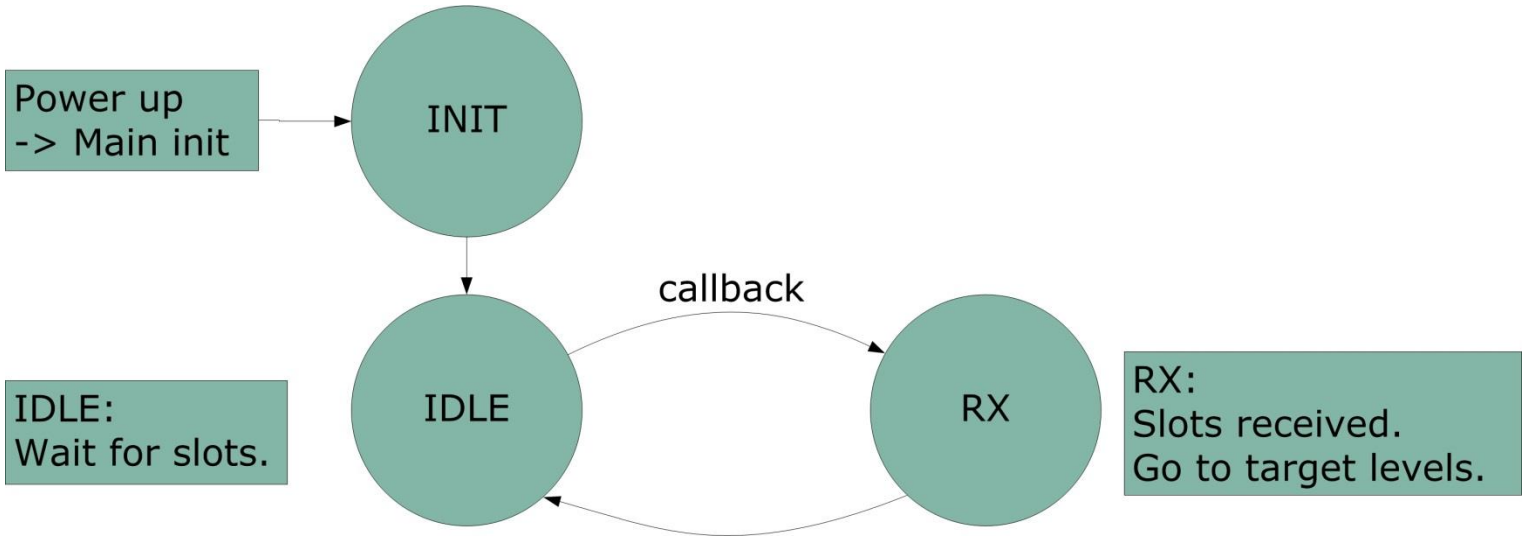
# DMX512 receiving device – Hardware overview

- XMC1000 LED Lighting Application Kit comprising of
  - XMC1200 Boot Kit
  - Colour LED card


- Kit schematics, documentation
  - http://www.infineon.com/cms/en/product/evaluation-boards/KIT_XMC1X_AK_LED_001/productType.html?productType=db3a30443ba77cfd013baec9c7880ca9

# Agenda

# DMX512 receiving device – Software overview




Flow chart: DMX512 receiving device – Software overview

# Agenda

# DMX512 receiving device – Highlight MCU features

- BCCU
  - 12-bit intensities
  - Up to 9 channels: convenient for driving multi-channel lamps
  - Separate dimming and color control: dimming level can be adjusted while preserving color output naturally, vice versa
  - 12-bit dimming level

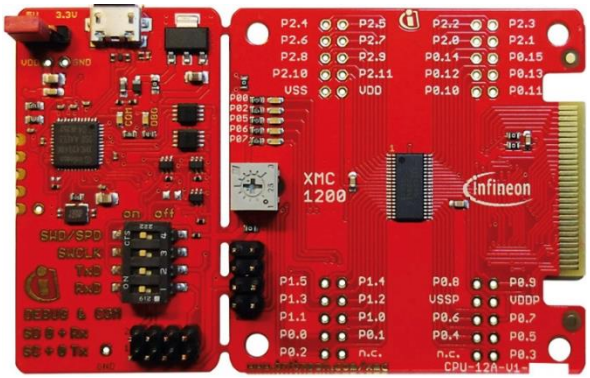
- USIC
  - USIC channel detects DMX512 packet and break automatically

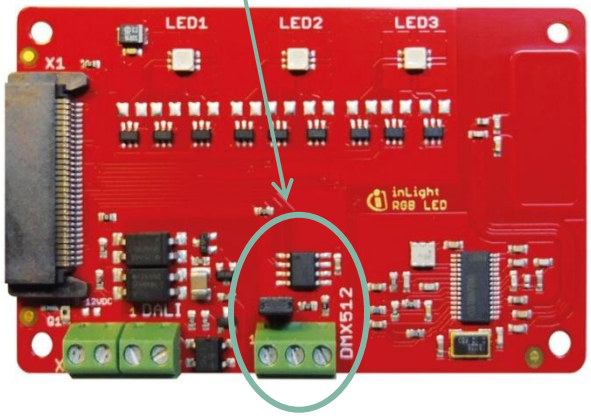
- [Optional] CCU4
  - Capture mode and timer automatically and accurately detects break

# Agenda

# DMX512 receiving device
# Hands-on training

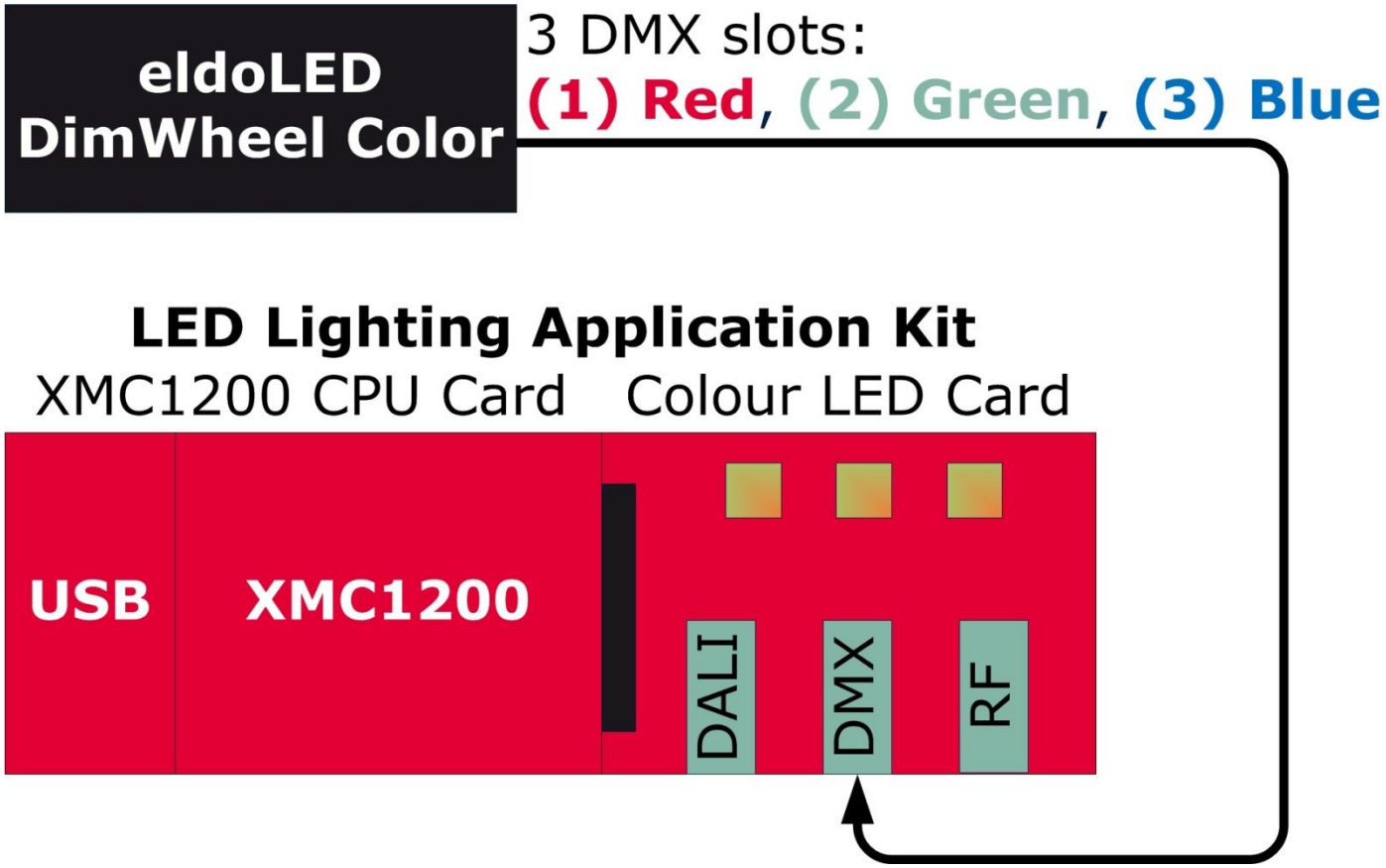
- Receiving device
  - XMC1200 Boot Kit + Colour LED card



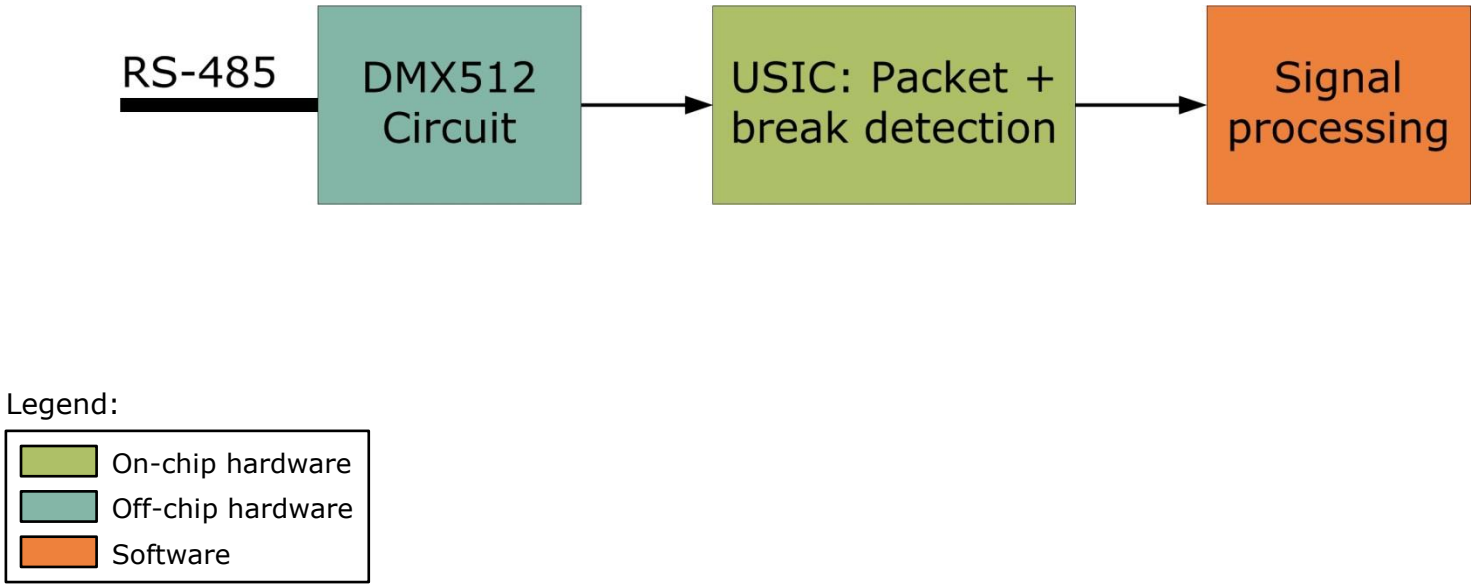




- Transmitting device
  - eldoLED DimWheel Colour EU DMX Controller

# DMX512 receiving device
# Hands-on: hardware setup

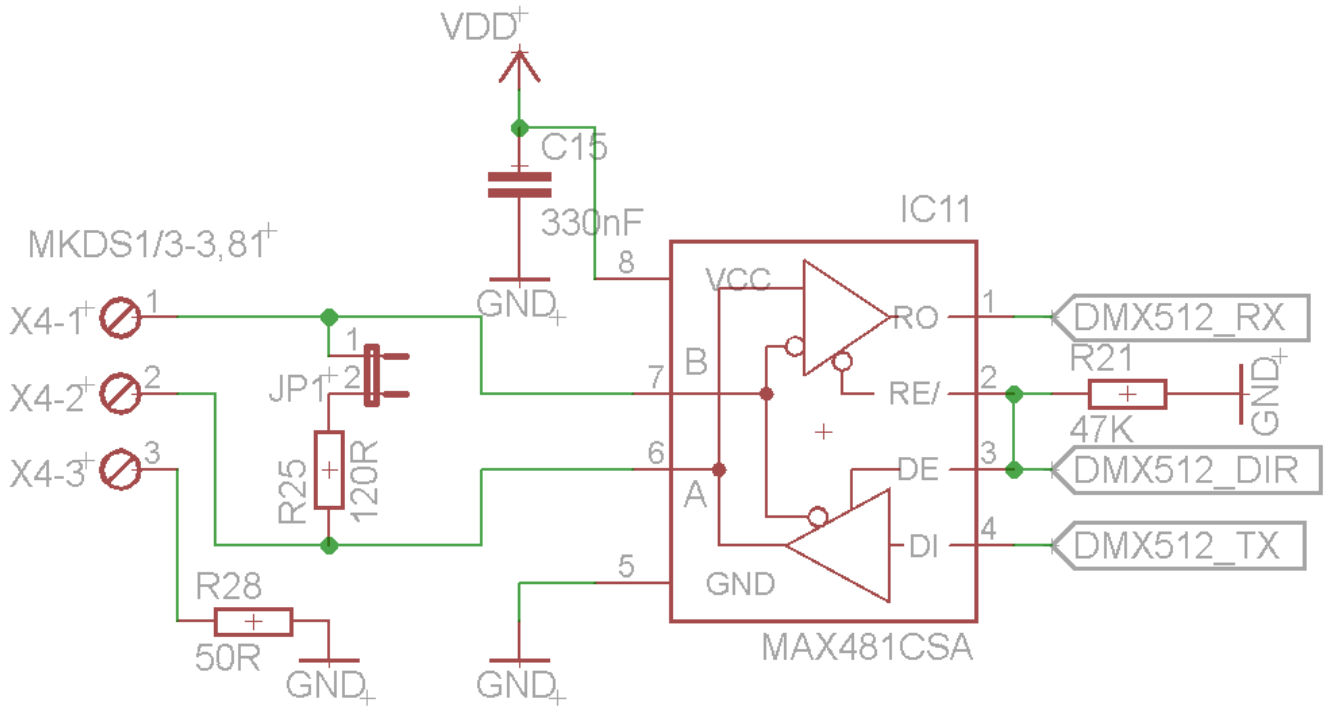


# DMX512 receiving device
# Hands-on: block diagram



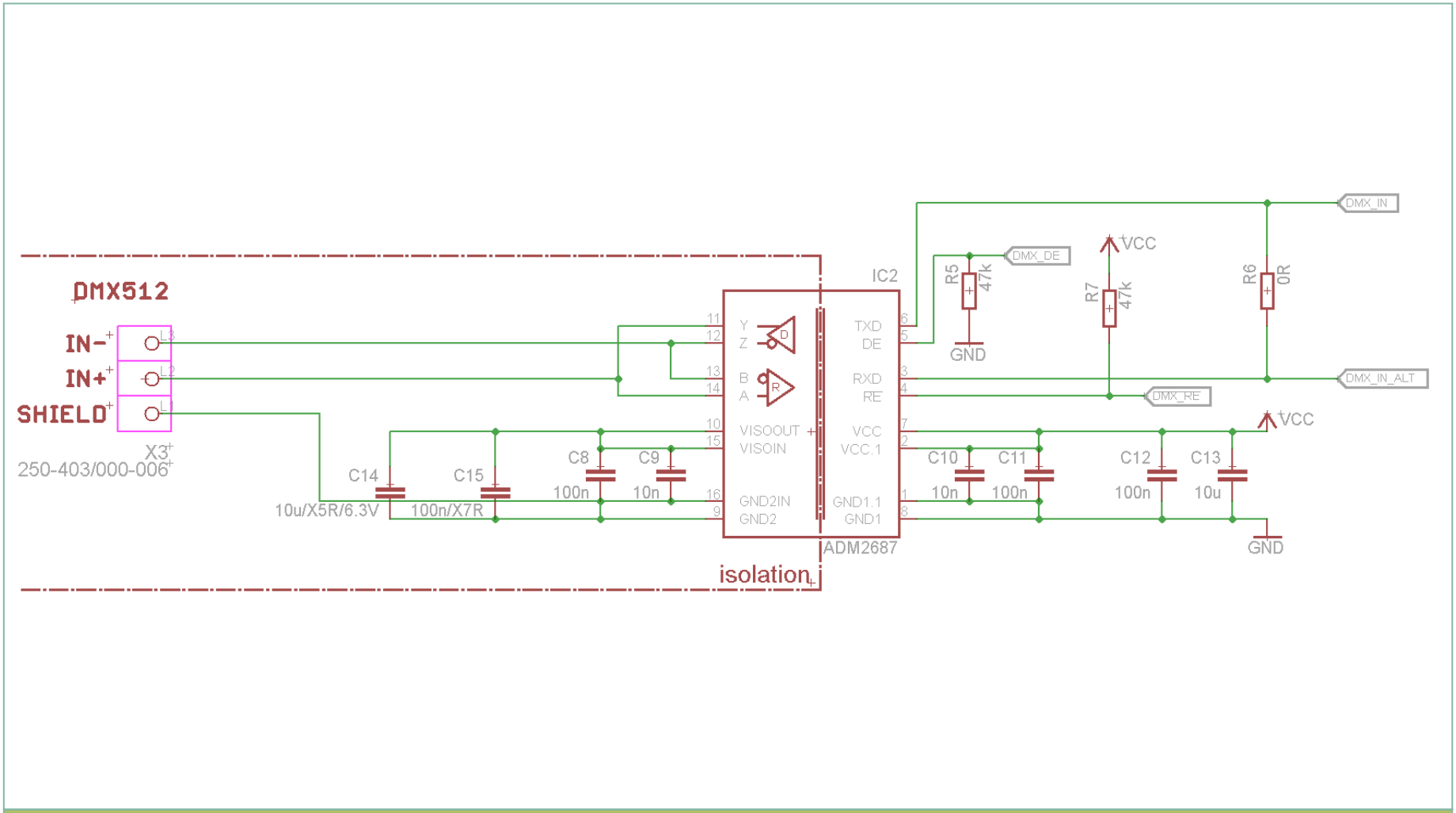

Block diagram: DMX512 receiving device demo

# DMX512 receiving device
# Hands-on: board schematic




Schematic: Simple non-isolated DMX512 interface

# DMX512 receiving device
# Hands-on: board schematic



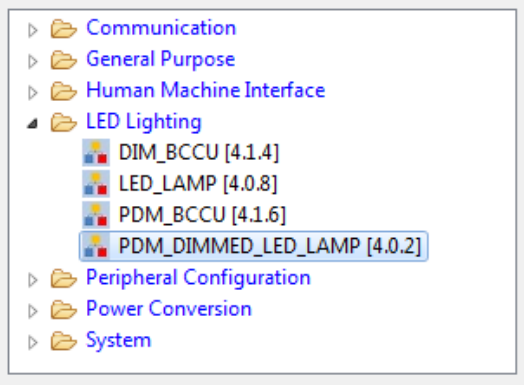

Schematic: Isolated DMX512 interface

# DMX512 receiving device
# Hands-on: software, list of DAVE™ APPs

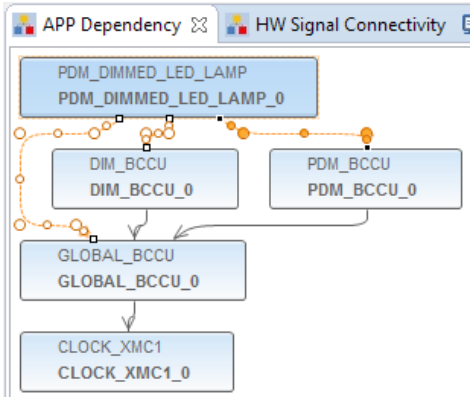
- PDM_DIMMED_LED_LAMP
  - Aggregates GLOBAL_BCCU and PDM_BCCU APPs
  - Provides configurations, color and dimming control for RGB LED lamp
- DMX512_RD
  - DMX512 application stack
- DIGITAL_IO
  - Initializes multiple IO pins that are connected to the other 2 unused RGB LEDs on board
  - Also initializes DMX input pin
- [Optional] EVENT_DETECTOR, EVENT_GENERATOR
  - For accurate break detection

# DMX512 receiving device
# Hands-on: HOT (1/14)

- Add one instance of PDM_DIMMED_LED_LAMP APP to the project

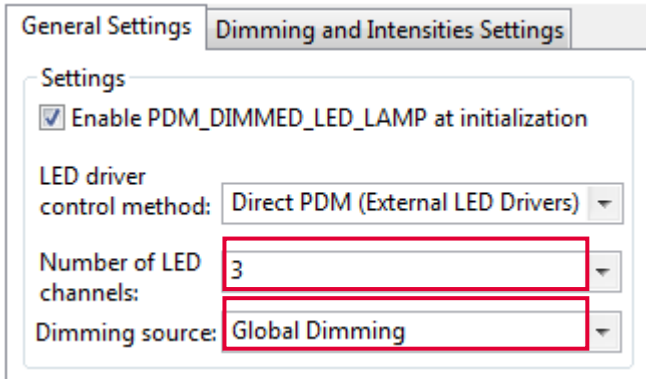



- PDM_DIMMED_LED_LAMP aggregates DIM_BCCU, PDM_BCCU, GLOBAL_BCCU and CLOCK_XMC1. Double-click PDM_DIMMED_LED_LAMP in APP Dependency View to open the UI Editor

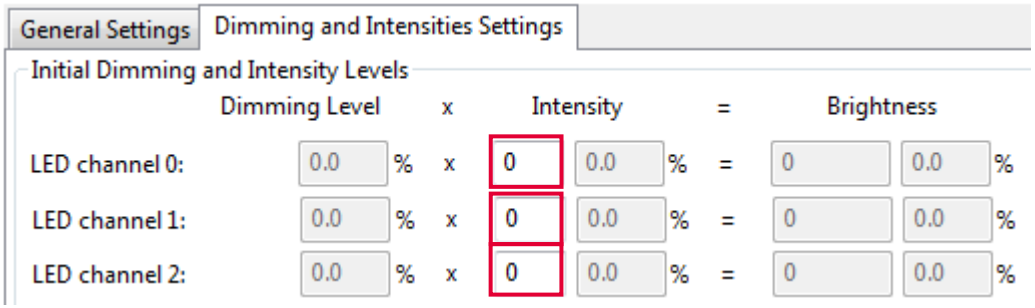


# DMX512 receiving device
# Hands-on: HOT (2/14)

- Under *General Settings* tab,
  - Select 3 LED channels
  - Select *Global Dimming* as dimming source

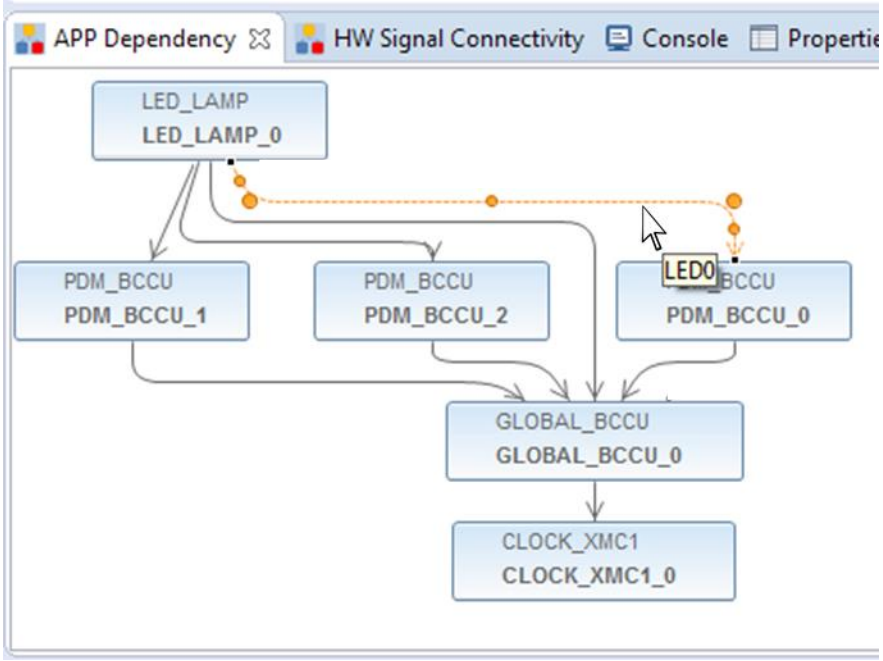



- Under *Dimming and Intensities Settings* tab,
  - Set intensities to 0



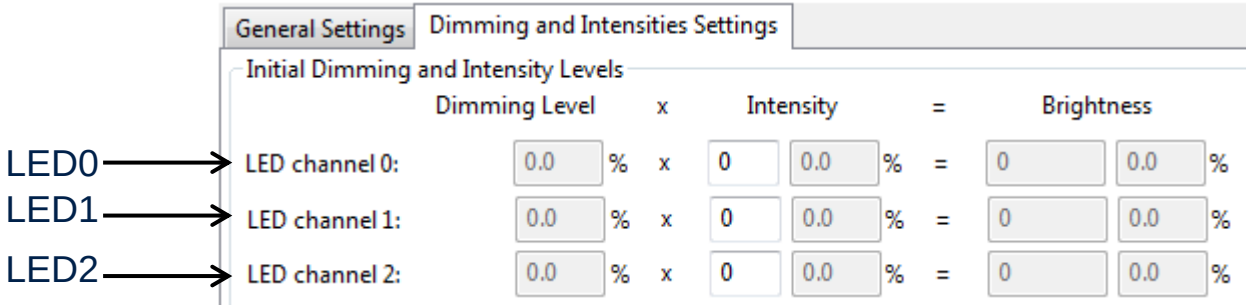
# DMX512 receiving device
# Hands-on: HOT (3/14)

- 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



# DMX512 receiving device
# Hands-on: HOT (4/14)

› The labels correspond to the LED channels in the UI




› Rename the PDM_BCCU instance label according to the table below

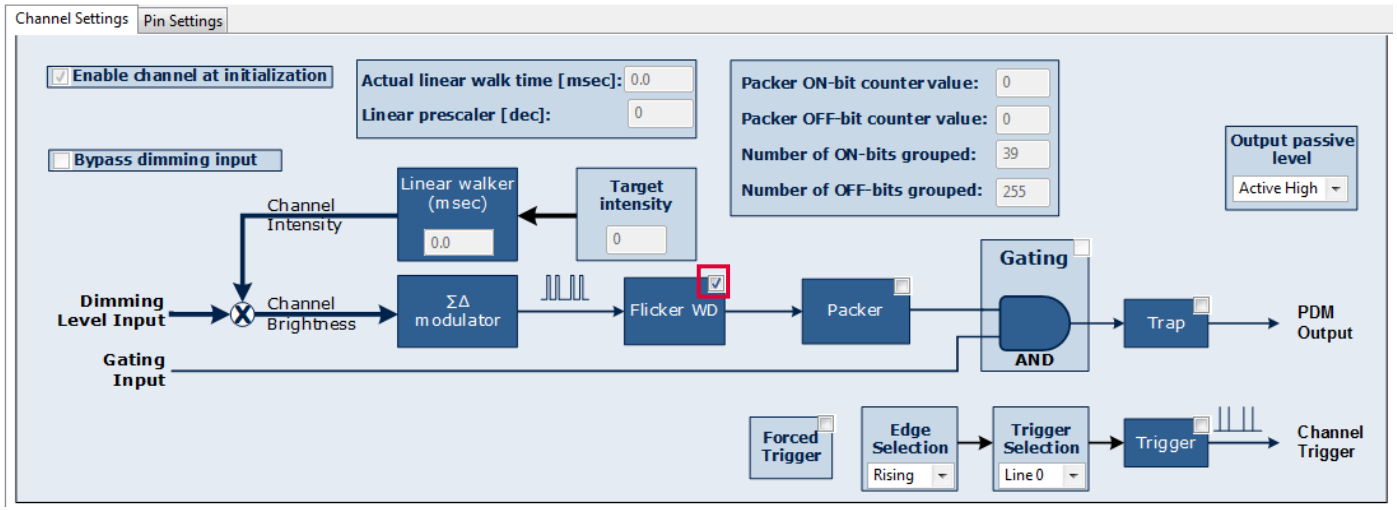
- Right-click PDM_BCCU APP
- Select “Rename Instance Label”

| Label | New Label |
|---|---|
| LED0 | RED |
| LED1 | GREEN |
| LED2 | BLUE |

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

# DMX512 receiving device
# Hands-on: HOT (5/14)

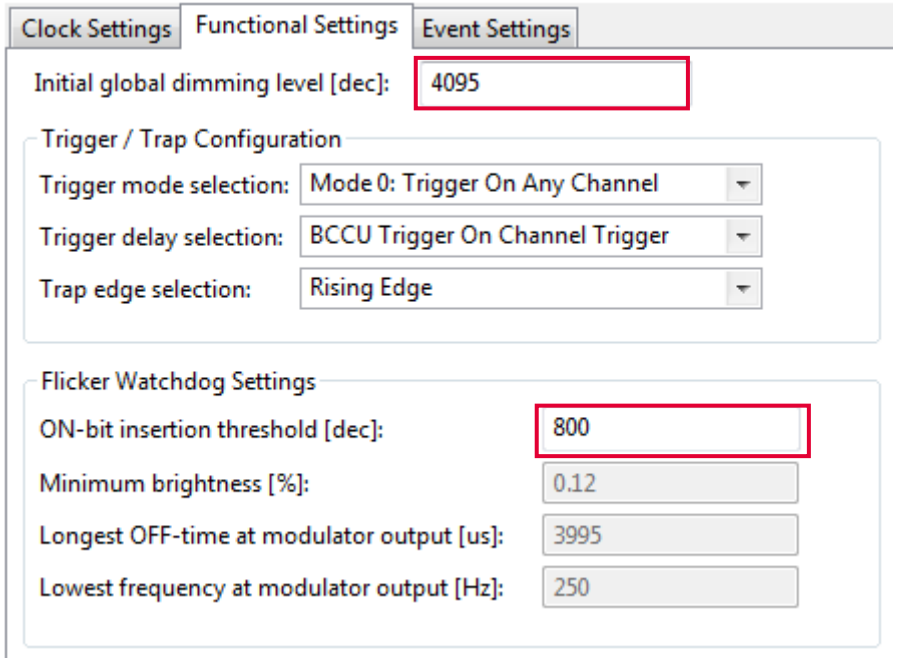
- Open UI of a PDM_BCCU APP
- Enable *Flicker Watchdog*




- Repeat for other 2 PDM_BCCU instances

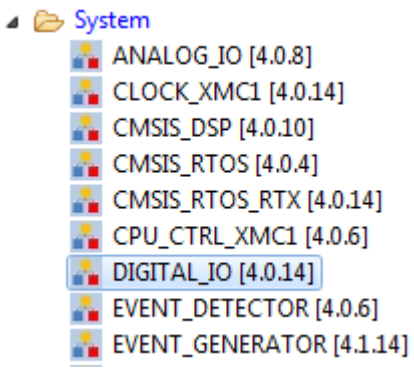
# DMX512 receiving device
# Hands-on: HOT (6/14)

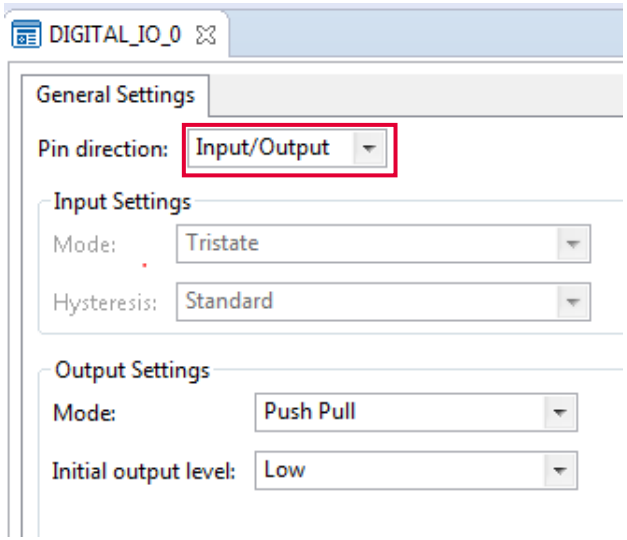
- Open UI of GLOBAL_BCCU APP
- Under *Function Settings* tab,
  - Set *initial global dimming level* to 4095
  - Set *Flicker Watchdog threshold* to 800



# DMX512 receiving device
# Hands-on: HOT (7/14)

- Add 6 instances of DIGITAL_IO APPs to the project for the pins to the unused LEDs
- Open the UI of a DIGITAL_IO APP
- Configure the pin direction as “Input/Output”

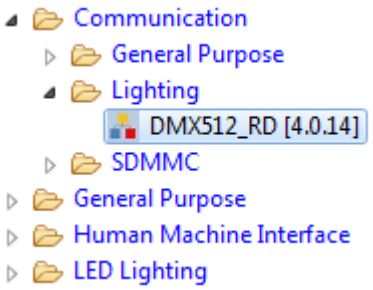


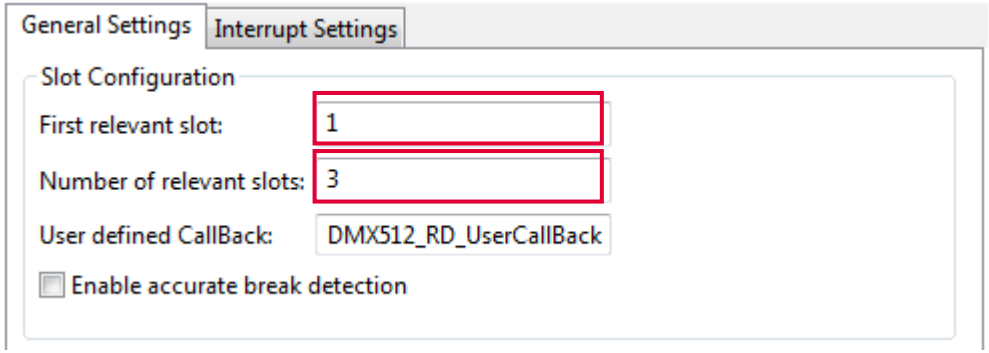




- Repeat for other 5 DIGITAL_IO instances
- Rename instance label for all 6 DIGITAL_IO instances as “UNUSED_LEDx” where x is 1 to 6

# DMX512 receiving device
# Hands-on: HOT (8/14)

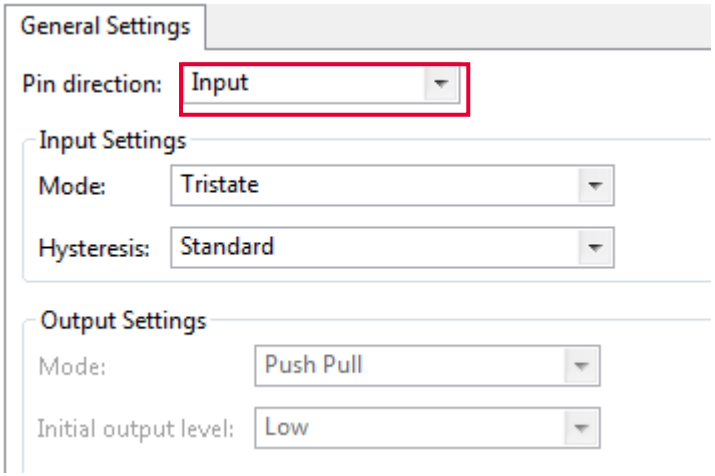
- Add an instance of DMX512_RD to the project
- Open UI of DMX512_RD APP
- Under *General Settings* tab,
  - Configure *First relevant slot* to 1
  - Configure *Number of relevant slots* to 3



# DMX512 receiving device
# Hands-on: HOT (9/14)

- Add one instance of DIGITAL_IO APP for configuring DMX input pin
- Open UI of DIGITAL_IO APP
- Configure pin direction as “Input”




- Rename instance label as “DMX_INPUT”

# DMX512 receiving device
# Hands-on: HOT (10/14)

- Connect DMX_INPUT to DMX512_RD APP
  - Right-click DMX_INPUT
    - Select “HW Signal Connections”
    - Configure as follows:

| Source APP Instance Name | Source Signal | Connect To | Target APP Instance Name | Target Signal |
|---|---|---|---|---|
| DMX_INPUT | | | | |
| | pin | ----> | DMX512_RD_0 | dmx512_input |
| | Not Selected | ----> | Not Selected | Not Selected |
| | | | | |

    - Click “Solve and Save”
    - Click “Close”

# DMX512 receiving device
# Hands-on: HOT (11/14)

- Open “Manual Pin Assignment” window by clicking the shortcut button
- Assign the pins as follows:

| APP Instance Name | APP Pin Name | Pin Number (Port) |
|---|---|---|
| BLUE | | |
| | PDM Output pin | #18 ( P0.1 ) |
| DMX_INPUT | | |
| | pin | #36 ( P2.1 ) |
| GREEN | | |
| | PDM Output pin | #30 ( P0.11 ) |
| RED | | |
| | PDM Output pin | #21 ( P0.4 ) |
| UNUSED_LED1 | | |
| | pin | #22 ( P0.5 ) |
| UNUSED_LED2 | | |
| | pin | #23 ( P0.6 ) |
| UNUSED_LED3 | | |
| | pin | #24 ( P0.7 ) |
| UNUSED_LED4 | | |
| | pin | #27 ( P0.8 ) |
| UNUSED_LED5 | | |
| | pin | #28 ( P0.9 ) |
| UNUSED_LED6 | | |
| | pin | #29 ( P0.10 ) |

- Click “Solve and Save”
- Click “Close”

# DMX512 receiving device
# Hands-on: HOT (12/14)

- Generate code 
- In Main.c, define the DMX512_RD callback function:

```
void DMX512_RD_UserCallBack(void)
{
  /* Extract 8-bit information for Red color */
  PDM_DIMMED_LED_LAMP.config->led_intensity[0] = DMX512_RD_0_rx_array[0] << 4U;
  /* Extract 8-bit information for Green color */
  PDM_DIMMED_LED_LAMP.config->led_intensity[1] = DMX512_RD_0_rx_array[1] << 4U;
  /* Extract 8-bit information for Blue color */
  PDM_DIMMED_LED_LAMP.config->led_intensity[2] = DMX512_RD_0_rx_array[2] << 4U;
  /* Change lamp color */
  PDM_DIMMED_LED_LAMP_SetColor(&PDM_DIMMED_LED_LAMP_0);
}
```

# DMX512 receiving device
# Hands-on: HOT (13/14)

- DMX512_RD callback function
  - It is called after slot data detection
  - Typically, the LED channel intensities are updated here to achieve what the DMX512 master is requesting
  - 8-bit intensity information in this example

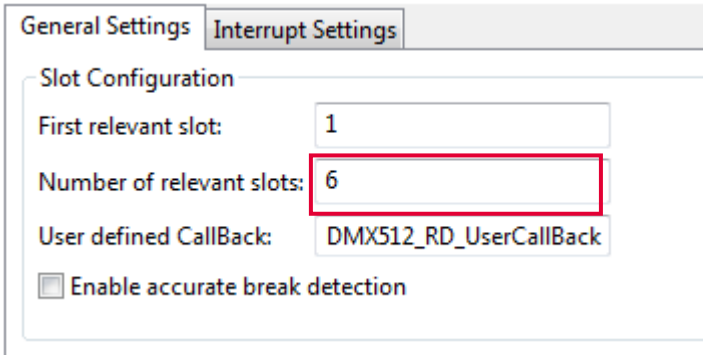
# DMX512 receiving device
# Hands-on: HOT (14/14)

- Build project
- Connect XMC1200 Boot Kit to PC
- Download code
- Start code
- Dial knob on eldoLED DimWheel Colour
- Observe LEDs on Colour LED card

# Agenda

# DMX512 receiving device
# Extras: 16-bit slots

- If DMX512 transmitting device transmits slots of 16 bits,
  - In DMX512_RD UI Editor:
    - Set *no. of relevant slots* to 6



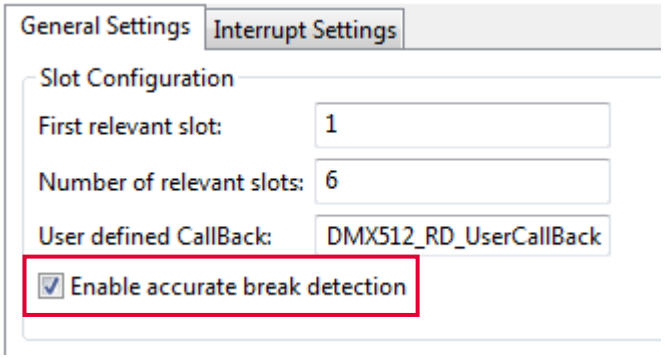

  - Re-generate code 
  - Adjust code as follows:

```
void DMX512_RD_UserCallBack(void)
{
  RGB_LAMP.config->led_intensity[0] = (uint16_t)((DMX512_RD_0_rx_array[0] << 4U) + (DMX512_RD_0_rx_array[1] >> 4U)); /* 16-bit information for Red color */
  RGB_LAMP.config->led_intensity[1] = (uint16_t)((DMX512_RD_0_rx_array[2] << 4U) + (DMX512_RD_0_rx_array[3] >> 4U)); /* 16-bit information for Green color */
  RGB_LAMP.config->led_intensity[2] = (uint16_t)((DMX512_RD_0_rx_array[4] << 4U) + (DMX512_RD_0_rx_array[5] >> 4U)); /* 16-bit information for Blue color */
  PDM_DIMMED_LED_LAMP_SetColor(&RGB_LAMP);
}
```

  - Rebuild project 
  - Download code 

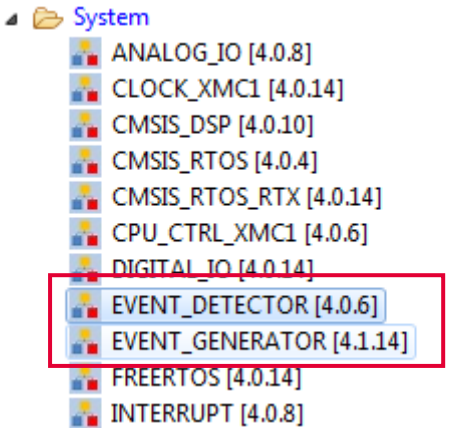
# DMX512 receiving device
# Extras: accurate break detection (1/6)

- DMX512 defines break as low signal for minimum duration of 92µs
- By default, DMX512_RD uses USIC Sync break for break detection
  - Functional but does not confirm the minimum duration
- Accurate detection can be achieved by using a CCU4 slice
  - In DMX512_RD UI Editor:
    - Enable accurate break detection



# DMX512 receiving device
# Extras: accurate break detection (2/6)

- Add EVENT_DETECTOR and EVENT_GENERATOR APPs (one instance each) to project
  - This is necessary because the input pin (P2.1) is not connected to any CCU4 slice so the input signal has to be rerouted via ERU




- Open the UI of EVENT_DETECTOR

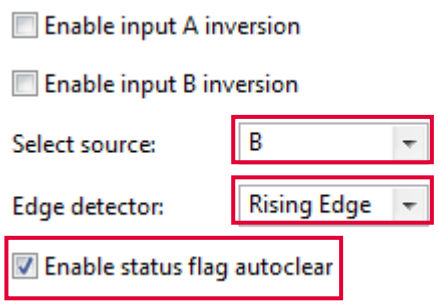
# DMX512 receiving device
# Extras: accurate break detection (3/6)

- Select “B” as request source as P2.1 is connected to input source B of the ERU slice

**Table 6-3 ERU0 Pin Connections**

| Global Inputs/Outputs | Connected To | I/O | Description |
|---|---|---|---|
| ERU0.0B2 | ORC0.OUT | I | |
| ERU0.0B3 | VADC0.G1BFLOUT0 | I | from ADC boundary flag |
| ERU0.1A0 | ACMP1.OUT | I | |
| ERU0.1A1 | P2.5 | I | |
| ERU0.1A2 | ORC3.OUT | I | |
| ERU0.1A3 | VADC0.G0BFLOUT1 | I | from ADC boundary flag |
| ERU0.1B0 | P2.1 | I | |
| ERU0.1B1 | P2.3 | I | |

- Select “Rising edge” detection
- Enable status flag autoclear



# DMX512 receiving device
# Extras: accurate break detection (4/6)

- Open the UI of EVENT_GENERATOR
- Enable pattern detection


- Connect DMX_INPUT pin to EVENT_DETECTOR
  - Right-click DMX_INPUT
  - Select “HW Signal Connections”
  - Configure as follows:

| | | | | |
|---|---|---|---|---|
| DMX_INPUT | | | | |
| | pin | ----> | DMX512_RD_0 | dmx512_input |
| | pin | ----> | EVENT_DETECTOR_0 | signal_b |

  - Click “Save”
  - Click “Close”

# DMX512 receiving device
# Extras: accurate break detection (5/6)

- Connect EVENT_DETECTOR status signal to EVENT_GENERATOR
  - Right-click EVENT_DETECTOR
  - Select “HW Signal Connections”
  - Configure as follows:

| | | | | |
|---|---|---|---|---|
| EVENT_DETECTOR_0 | | | | |
| | status | ----> | EVENT_GENERATOR_0 | pattern |

  - Click “Save”
  - Click “Close”

# DMX512 receiving device
# Extras: accurate break detection (6/6)

- Connect EVENT_GENERATOR pattern detect signal to DMX512_RD for break detection
  - Right-click EVENT_GENERATOR
  - Select “HW Signal Connections”
  - Configure as follows:

| | | | | |
|---|---|---|---|---|
| EVENT_GENERATOR_0 | | | | |
| | pdout | ----> | DMX512_RD_0 | accurate_break_detection |

  - Click “Save”
  - Click “Close”
- Re-generate code 
- Rebuild project 
- Download code 

# General information

- Where to buy kit?
  - www.infineon.com/cms/en/product/evaluation-boards/KIT_XMC1X_AK_LED_001/productType.html?productType=db3a30443ba77cfd013baec9c7880ca9

- For latest updates, please refer to:
  - www.infineon.com/xmc1000

- For support:
  - www.infineonforums.com

# Resource listing

- DMX512 receiving device DAVE™ project

  http://www.infineon.com/cms/en/product/evaluation-boards/KIT_XMC1X_AK_LED_001/productType.html?productType=db3a30443ba77cfd013baec9c7880ca9

  (look under Documents tab)

- LED Lighting Application Kit documentation

  www.infineon.com/cms/en/product/evaluation-boards/KIT_XMC1X_AK_LED_001/productType.html?productType=db3a30443ba77cfd013baec9c7880ca9

- eldoLED DimWheel Colour EU

  http://www.eldoled.com/led-drivers/accessories/dimwheel-colour/dimwheel-colour-eu/

Part of your life. Part of tomorrow.