ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Programing of embedded systems

6. Neopixels driver

Lead Partner: Warsaw University of Technology

Authors: Daniel Krol

University of Applied Sciences in Tarnow

Erasmus+

GRANT NUMBER: 2020-1-PL01-KA226-HE-095653

Declaration

This report has been prepared in the context of the ENGINE project. Where other published and unpublished source materials have been used, these have been acknowledged.

Copyright

Funding Disclaimer

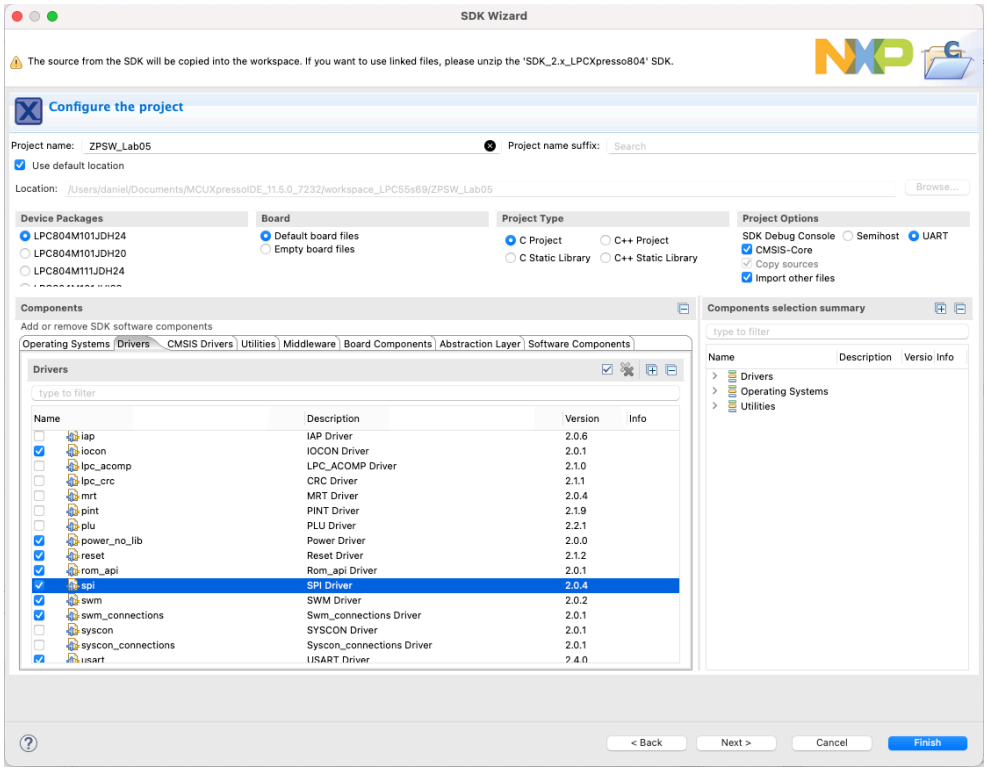
This project has been funded with support from the European Commission. This report reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Programing of embedded systems

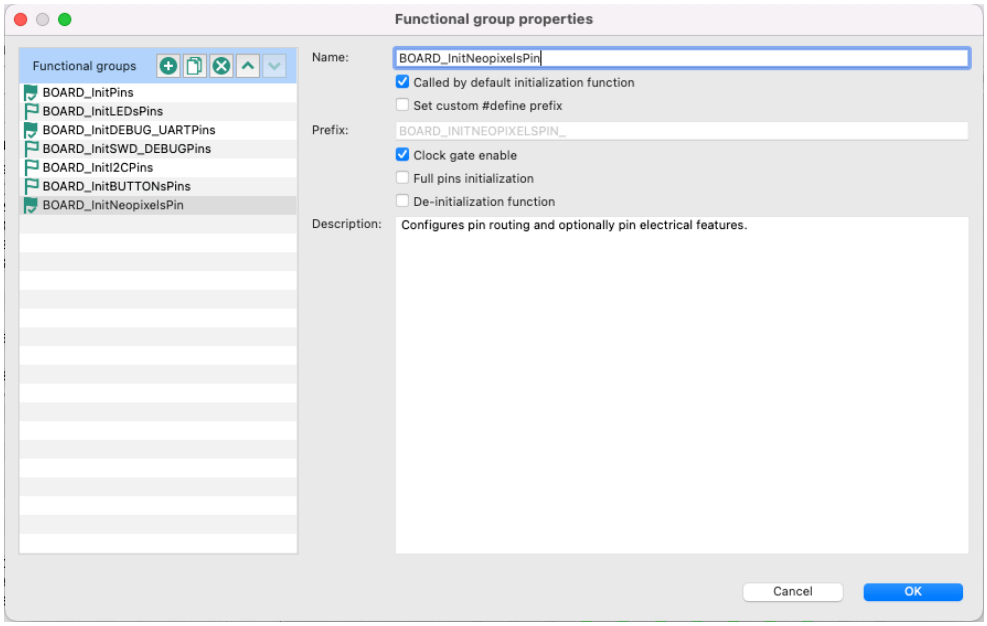
6. Neopixels driver

I. Using the SPI interface

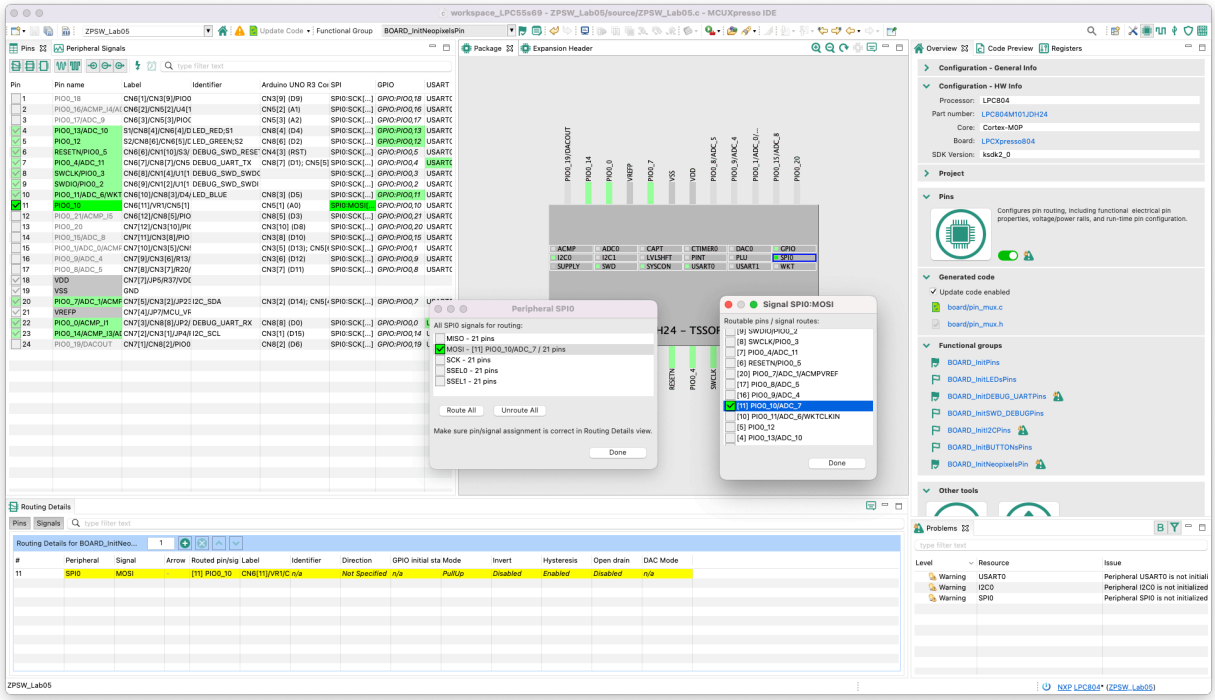
1. Create a new project for the *LPCXpresso804* board and name it eg *Lab06*.
2. Add *SPI* driver:



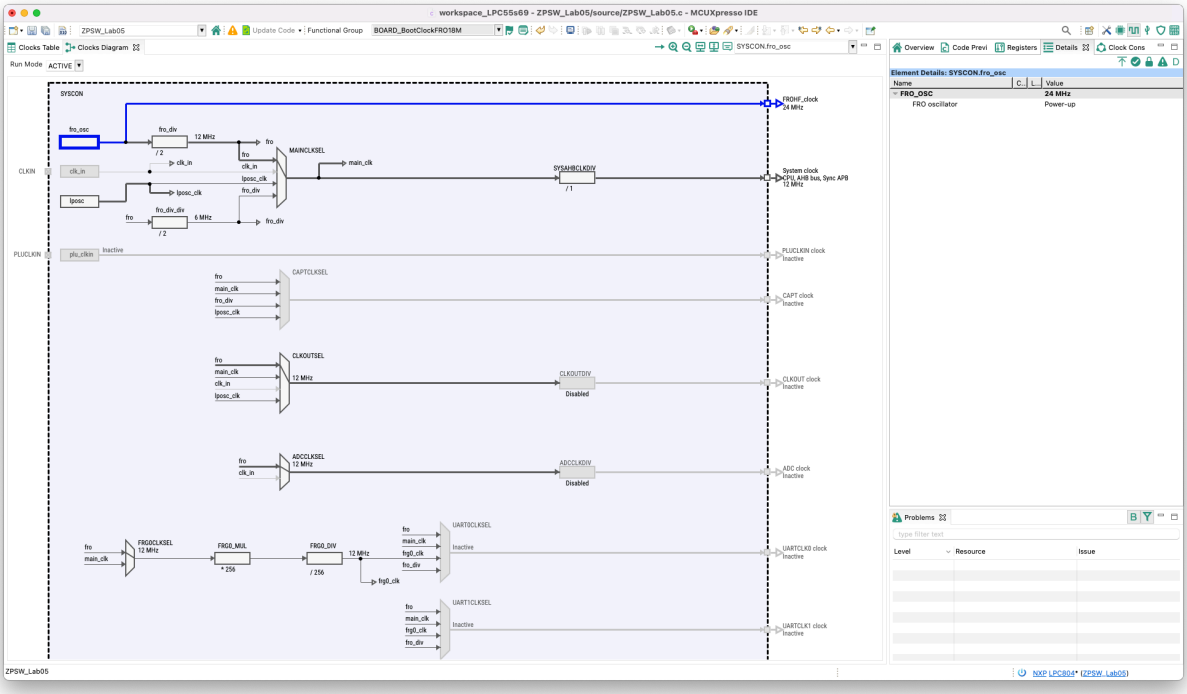
3. Go to *Config Tool -> Pins* and then in the *Functional group* create a new preset *BOARD_InitNeopixelsPin*:



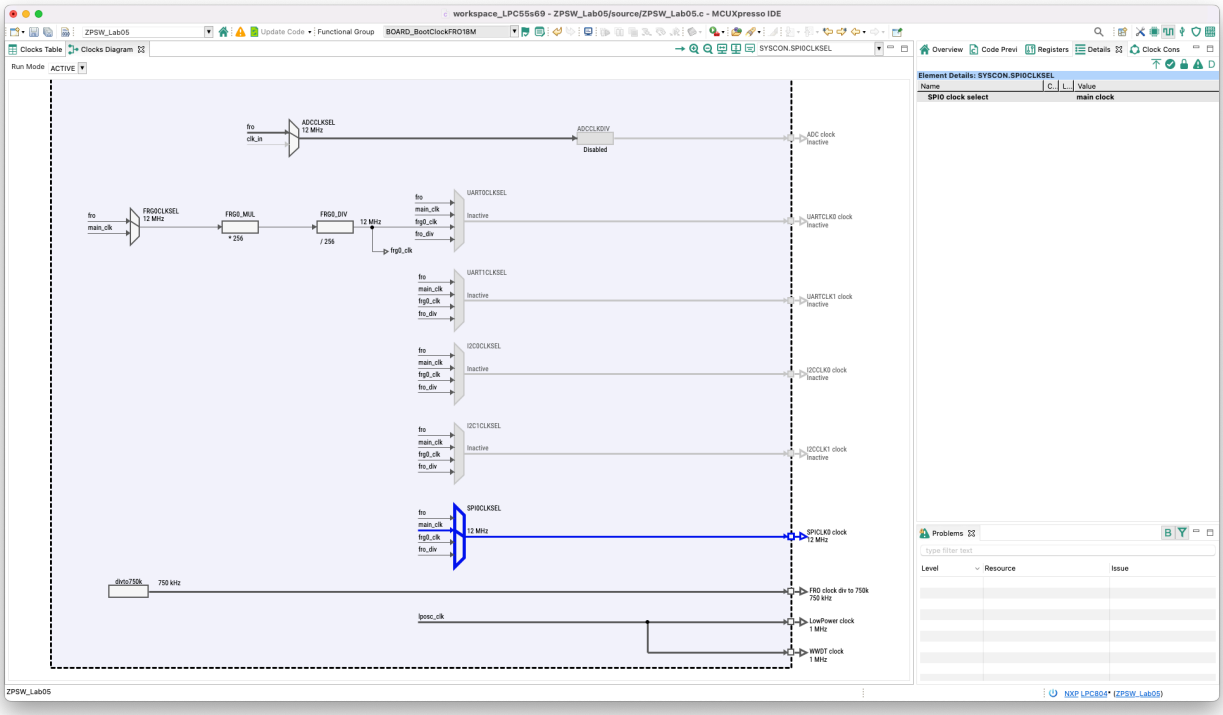
4. Connect the *MOSI* line of the *SPI0* interface to the *PIO_010* pin of the microcontroller:



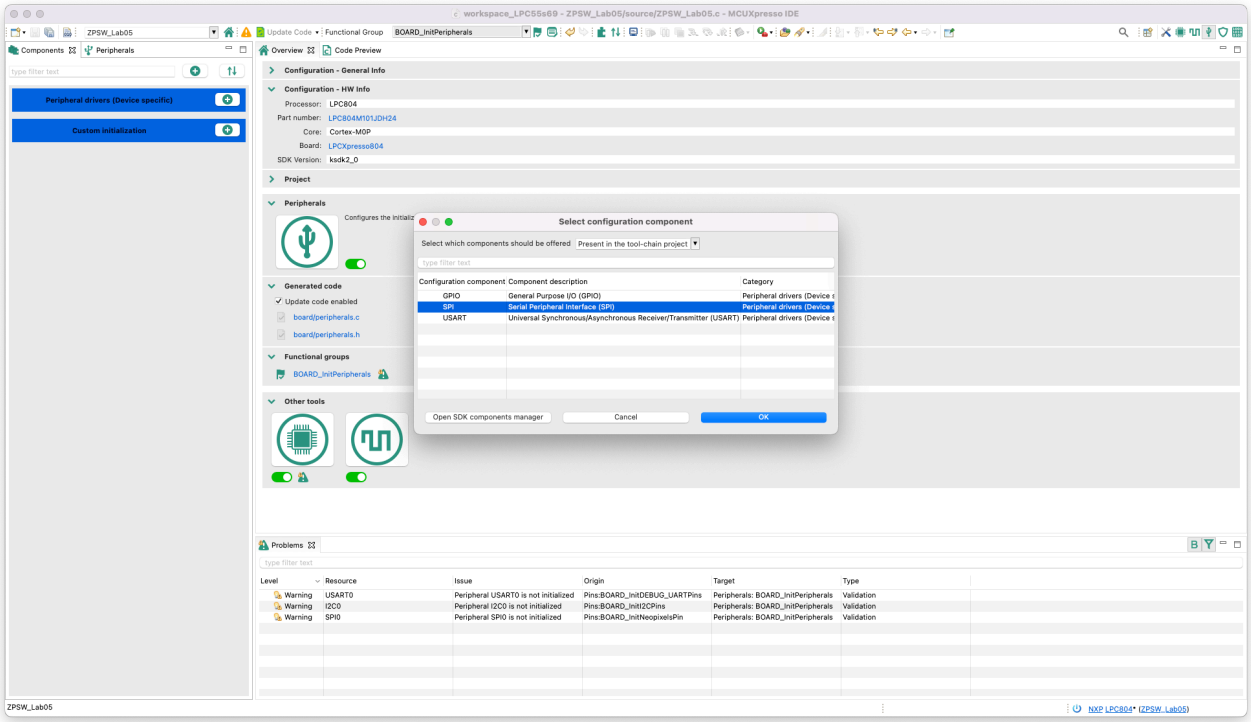
5. In the Clocks tab, click on the *FRO_OSC* oscillator block and change its frequency to 24 MHz:



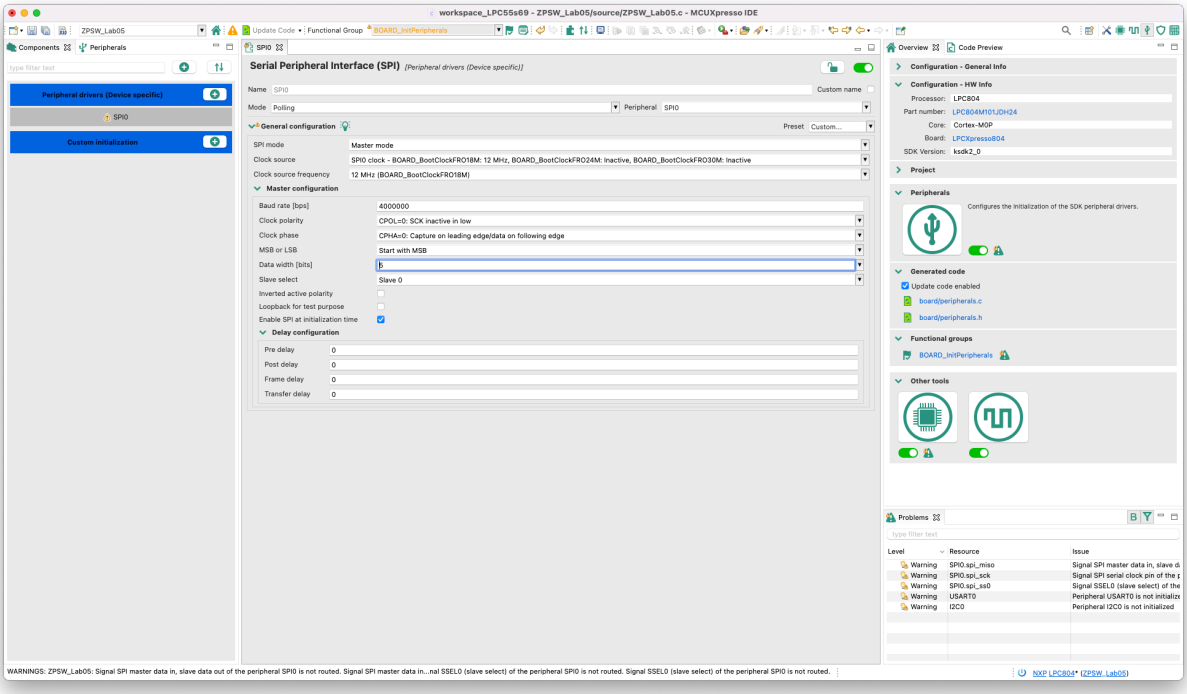
6. Then bring the 12 MHz clock signal to the *SPI0* interface by selecting *main_clk* in the *SPI0CLKSEL* block:



7. In the *Peripherals* tab, select the *SPI driver*:



8. Configure *SPI* transmission parameters in *Polling mode* for the *SPI0* interface:



9. Go to the main project file and modify the code as below:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"

#define LEDS 37
#define GET_BIT(k, n)     (k &   (1 << (n)))
#define SET_BIT(k, n)     (k |=  (1 << (n)))
#define CLR_BIT(k, n)     (k &= ~(1 << (n)))

#define CODE_0                                  0b10000
#define CODE_1                                  0b11100

uint32_t colors[LEDS]={0};

void Neopixels_Send(SPI_Type *base, uint32_t n, uint32_t *value) {
          uint16_t LED_data=0;


          for(int j=0;j<n;j++) {
                    for(int i=23;i>=0;i--) {
                              LED_data = GET_BIT(value[j], i) ? CODE_1 : CODE_0;

                              while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                              base->TXDAT = LED_data ;
                    }
          }
          // Reset >= 50 us
          LED_data=0;
          for(int j=0;j<50;j++) {
                    while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                    base->TXDAT = LED_data ;
          }
}

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif
```

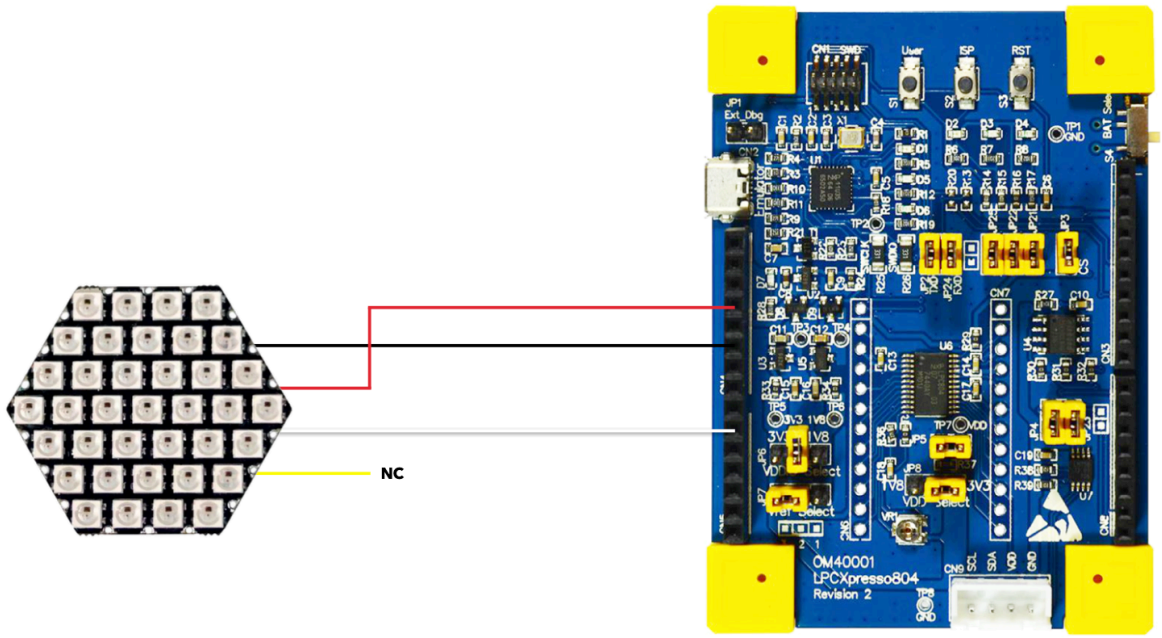
```
        SPI_WriteConfigFlags(SPI0_PERIPHERAL,  kSPI_ReceiveIgnore);

        while(1) {

                colors[18] = 0x00000F;
                Neopixels_Send(SPI0_PERIPHERAL, LEDS, colors);

        }
        return 0 ;
}
```

10. Connect the LED module to the prototype board according to the diagram below:



11. Build the project in the ***Release*** configuration, program the microcontroller and check the program operation.

ATTENTION! Do not turn on all the LEDs with full brightness without additional power supply to the module - it may damage the on board stabilizer. The prototype board is able to power the entire LED module with approx. 1/16 brightness - the maximum value for the white color: **HEX 0x0F0F0F.**
DEC R: 15, G: 15, B: 15

II. Function *setRGB*

1. Write a function that sets the color in RGB 888 format.
2. Add a system timer and check the brightness adjustment of individual RGB components:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"

#define LEDS 37
#define GET_BIT(k, n)     (k &   (1 << (n)))
#define SET_BIT(k, n)     (k |=  (1 << (n)))
#define CLR_BIT(k, n)     (k &= ~(1 << (n)))

#define CODE_0                                  0b10000
#define CODE_1                                  0b11100

uint32_t colors[LEDS]={0};
uint8_t k=0;


void Neopixels_Send(SPI_Type *base, uint32_t n, uint32_t *value) {
          uint16_t LED_data=0;


          for(int j=0;j<n;j++) {
                    for(int i=23;i>=0;i--) {
                              LED_data = GET_BIT(value[j], i) ? CODE_1 : CODE_0;

                              while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                              base->TXDAT = LED_data ;
                    }
          }
          // Reset >= 50 us
          LED_data=0;
          for(int j=0;j<50;j++) {
                    while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                    base->TXDAT = LED_data ;
          }
}

inline uint32_t setRGB(uint8_t r, uint8_t g, uint8_t b) {

          return ((g<<16) | (r<<8) | b);
}


void SysTick_Handler(void) {

          colors[18] = setRGB(k++, 0,   0);
          //colors[18] = setRGB(0,   k++, 0);
          //colors[18] = setRGB(0,   0,   k++);
          Neopixels_Send(SPI0_PERIPHERAL, LEDS, colors);
}

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif

          SPI_WriteConfigFlags(SPI0_PERIPHERAL,  kSPI_ReceiveIgnore);

          SysTick_Config(SystemCoreClock / 100); // 100 Hz

          while(1) {

          }
          return 0 ;
}
```

3. Build a project, program the layout and test the program.

II. Function *setBrightness*

1. Write a function that sets the brightness of a given color:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"

#define LEDS 37
#define GET_BIT(k, n)     (k &   (1 << (n)))
#define SET_BIT(k, n)     (k |=  (1 << (n)))
#define CLR_BIT(k, n)     (k &= ~(1 << (n)))

#define CODE_0                                  0b10000
#define CODE_1                                  0b11100

uint32_t colors[LEDS]={0};
uint8_t k=0;


void Neopixels_Send(SPI_Type *base, uint32_t n, uint32_t *value) {
          uint16_t LED_data=0;


          for(int j=0;j<n;j++) {
                    for(int i=23;i>=0;i--) {
                              LED_data = GET_BIT(value[j], i) ? CODE_1 : CODE_0;

                              while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                              base->TXDAT = LED_data ;
                    }
          }
          // Reset >= 50 us
          LED_data=0;
          for(int j=0;j<50;j++) {
                    while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                    base->TXDAT = LED_data ;
          }
}

inline uint32_t setRGB(uint8_t r, uint8_t g, uint8_t b) {

          return ((g<<16) | (r<<8) | b);
}

uint32_t setBrightness(uint32_t color, uint8_t level) {

          uint8_t b = level *  (color & 0x0000FF)        / 255;
          uint8_t r = level * ((color & 0x00FF00) >> 8)  / 255;
          uint8_t g = level * ((color & 0xFF0000) >> 16) / 255;

          return ((g<<16) | (r<<8) | b);
}

void SysTick_Handler(void) {

          colors[18] = setRGB(255, 0, 128);
          colors[18] = setBrightness(colors[18], k++);

          Neopixels_Send(SPI0_PERIPHERAL, LEDS, colors);
}

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif

          SPI_WriteConfigFlags(SPI0_PERIPHERAL,  kSPI_ReceiveIgnore);

          SysTick_Config(SystemCoreClock / 100); // 100 Hz

          while(1) {

          }
          return 0 ;
}
```

2. Build a project, program the microcontroller and test the program for different RGB colors.

III. Simple animation

1. Write a function that "shifts" the shining point on the matrix:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"

#define LEDS 37
#define GET_BIT(k, n)     (k &   (1 << (n)))
#define SET_BIT(k, n)     (k |=  (1 << (n)))
#define CLR_BIT(k, n)     (k &= ~(1 << (n)))

#define CODE_0                                  0b10000
#define CODE_1                                  0b11100

uint32_t colors[LEDS]={0};
uint8_t k=0;
uint32_t rgbColor=0;

void Neopixels_Send(SPI_Type *base, uint32_t n, uint32_t *value) {
          uint16_t LED_data=0;


          for(int j=0;j<n;j++) {
                    for(int i=23;i>=0;i--) {
                              LED_data = GET_BIT(value[j], i) ? CODE_1 : CODE_0;

                              while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                              base->TXDAT = LED_data ;
                    }
          }
          // Reset >= 50 us
          LED_data=0;
          for(int j=0;j<50;j++) {
                    while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                    base->TXDAT = LED_data ;
          }
}

inline uint32_t setRGB(uint8_t r, uint8_t g, uint8_t b) {

          return ((g<<16) | (r<<8) | b);
}

void Animate(uint32_t color)
{
          for(int j=0;j<LEDS;j++)
                    colors[j]=0x000000;

          colors[k]=color;

          k++;
          if(k>=LEDS)
                    k=0;
}

void SysTick_Handler(void) {

          rgbColor = setRGB(0, 0, 15); // MAX RGB: (15, 15, 15)
          Animate(rgbColor);

          Neopixels_Send(SPI0_PERIPHERAL, LEDS, colors);
}

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif

          SPI_WriteConfigFlags(SPI0_PERIPHERAL,  kSPI_ReceiveIgnore);

          SysTick_Config(SystemCoreClock / 10); // 10 Hz

          while(1) {

          }
          return 0 ;
}
```

2. Build a project, program the microcontroller and test the program.

IV. Animation LUT (*lookup table*)

1. Write an animation using the LUT table. They do not exceed the value of 15 for each RGB component!

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"

#define LEDS 37
#define GET_BIT(k, n)     (k &   (1 << (n)))
#define SET_BIT(k, n)     (k |=  (1 << (n)))
#define CLR_BIT(k, n)     (k &= ~(1 << (n)))

#define CODE_0                                  0b10000
#define CODE_1                                  0b11100

uint32_t colors[LEDS]={0};
uint8_t k=0;
uint32_t rgbColor=0;


const bool pic1[4][LEDS] = {{

                        0,0,0,0,
                      0,0,0,0,0,
                     0,0,0,0,0,0,0,
                    0,0,0,1,0,0,0,
                     0,0,0,0,0,0,0,
                       0,0,0,0,0,
                        0,0,0,0,
},{
                        0,0,0,0,
                      0,0,0,0,0,
                     0,0,1,1,0,0,
                    0,0,1,0,1,0,0,
                     0,0,1,1,0,0,
                       0,0,0,0,0,
                        0,0,0,0,
},{
                        0,0,0,0,
                      0,1,1,1,0,
                     0,1,0,0,1,0,
                    0,1,0,0,0,1,0,
                     0,1,0,0,1,0,
                       0,1,1,1,0,
                        0,0,0,0,
},{
                        1,1,1,1,
                      1,0,0,0,1,
                     1,0,0,0,0,1,
                    1,0,0,0,0,0,1,
                     1,0,0,0,0,1,
                       1,0,0,0,1,
                        1,1,1,1,
}};

void Neopixels_Send(SPI_Type *base, uint32_t n, uint32_t *value) {
          uint16_t LED_data=0;


          for(int j=0;j<n;j++) {
                    for(int i=23;i>=0;i--) {
                              LED_data = GET_BIT(value[j], i) ? CODE_1 : CODE_0;

                              while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                              base->TXDAT = LED_data ;
                    }
          }
          // Reset >= 50 us
          LED_data=0;
          for(int j=0;j<50;j++) {
                    while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                    base->TXDAT = LED_data ;
          }
}

inline uint32_t setRGB(uint8_t r, uint8_t g, uint8_t b) {

          return ((g<<16) | (r<<8) | b);
}

void setImage(const bool *image,  uint32_t color) {

          for(int i=0; i<LEDS; i++) {

                    colors[i] = image[i] * color;
          }
}

void SysTick_Handler(void) {
```

```
          rgbColor = setRGB(0, 0, 15); // MAX RGB: (15, 15, 15)
          setImage(pic1[k++ % 4], rgbColor);
          Neopixels_Send(SPI0_PERIPHERAL, LEDS, colors);
}

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif

          SPI_WriteConfigFlags(SPI0_PERIPHERAL,  kSPI_ReceiveIgnore);

          SysTick_Config(SystemCoreClock / 5); // 5 Hz

          while(1) {

          }
          return 0 ;
}
```

2. Build a project, program the microcontroller and test the program.
3. Add another board with animation:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"

#define LEDS 37
#define GET_BIT(k, n)     (k &   (1 << (n)))
#define SET_BIT(k, n)     (k |=  (1 << (n)))
#define CLR_BIT(k, n)     (k &= ~(1 << (n)))

#define CODE_0                                      0b10000
#define CODE_1                                      0b11100

uint32_t colors[LEDS]={0};
uint8_t k=0;
uint32_t rgbColor=0;


const bool pic1[4][LEDS] = {{

                        0,0,0,0,
                      0,0,0,0,0,
                    0,0,0,0,0,0,0,
                  0,0,0,1,0,0,0,
                    0,0,0,0,0,0,0,
                      0,0,0,0,0,
                        0,0,0,0,
},{
                        0,0,0,0,
                      0,0,0,0,0,
                    0,0,1,1,0,0,
                  0,0,1,0,1,0,0,
                    0,0,1,1,0,0,
                      0,0,0,0,0,
                        0,0,0,0,
},{
                        0,0,0,0,
                      0,1,1,1,0,
                    0,1,0,0,1,0,
                  0,1,0,0,0,1,0,
                    0,1,0,0,1,0,
                      0,1,1,1,0,
                        0,0,0,0,
},{
                        1,1,1,1,
                      1,0,0,0,1,
                    1,0,0,0,0,0,1,
                  1,0,0,0,0,0,1,
                    1,0,0,0,0,1,
                      1,0,0,0,1,
                        1,1,1,1,
}};

const bool pic2[6][LEDS] = {{

                        1,1,1,1,
                      0,1,0,1,0,
                    0,0,1,1,0,0,
                  0,0,0,1,0,0,0,
                    0,0,0,0,0,0,0,
                      0,0,0,0,0,
                        0,0,0,0,
```

```
},{
                        0,0,0,1,
                       0,0,0,1,1,
                      0,0,0,1,0,1,
                     0,0,0,1,1,1,1,
                      0,0,0,0,0,0,
                       0,0,0,0,0,
                        0,0,0,0,
},{
                        0,0,0,0,
                       0,0,0,0,0,
                      0,0,0,0,0,0,
                     0,0,0,1,1,1,1,
                      0,0,0,1,0,1,
                       0,0,0,1,1,
                        0,0,0,1,
},{
                        0,0,0,0,
                       0,0,0,0,0,
                      0,0,0,0,0,0,
                     0,0,0,1,0,0,0,
                      0,0,1,1,0,0,
                       0,1,0,1,0,
                        1,1,1,1,
},{
                        0,0,0,0,
                       0,0,0,0,0,
                      0,0,0,0,0,0,
                     1,1,1,1,0,0,0,
                      1,0,1,0,0,0,
                       1,1,0,0,0,
                        1,0,0,0,
},{
                        1,0,0,0,
                       1,1,0,0,0,
                      1,0,1,0,0,0,
                     1,1,1,1,0,0,0,
                      0,0,0,0,0,0,
                       0,0,0,0,0,
                        0,0,0,0,
}};

void Neopixels_Send(SPI_Type *base, uint32_t n, uint32_t *value) {
          uint16_t LED_data=0;


          for(int j=0;j<n;j++) {
                    for(int i=23;i>=0;i--) {
                              LED_data = GET_BIT(value[j], i) ? CODE_1 : CODE_0;

                              while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                              base->TXDAT = LED_data ;
                    }
          }
          // Reset >= 50 us
          LED_data=0;
          for(int j=0;j<50;j++) {
                    while(!(base->STAT & SPI_STAT_TXRDY_MASK));
                    base->TXDAT = LED_data ;
          }
}

inline uint32_t setRGB(uint8_t r, uint8_t g, uint8_t b) {

          return ((g<<16) | (r<<8) | b);
}

void setImage(const bool *image,  uint32_t color) {

          for(int i=0; i<LEDS; i++) {

                    colors[i] = image[i] * color;
          }
}

void SysTick_Handler(void) {

          rgbColor = setRGB(0, 0, 15); // MAX RGB: (15, 15, 15)
          setImage(pic2[k++ % 6], rgbColor);
          Neopixels_Send(SPI0_PERIPHERAL, LEDS, colors);
}

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif

          SPI_WriteConfigFlags(SPI0_PERIPHERAL,  kSPI_ReceiveIgnore);
```

```
        SysTick_Config(SystemCoreClock / 5); // 5 Hz

        while(1) {

        }
        return 0 ;
}
```

4. Build a project, program the microcontroller and test the program.

V. Exercises

1. Create your own LUT animations
2. Create a color animation (remember not to exceed RGB values: 15, 15, 15 on many LEDs)