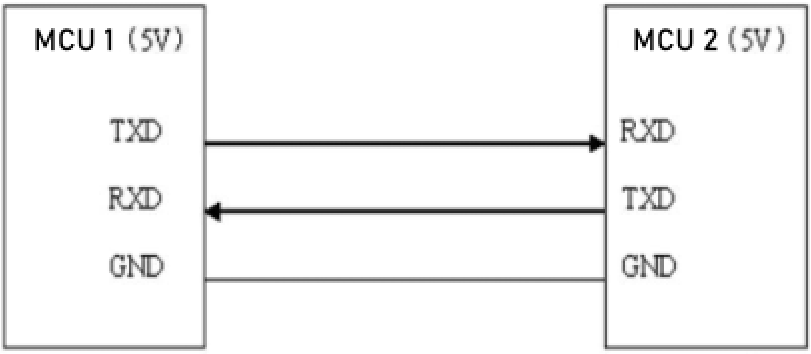
K210 Vision Module

2.1 Serial Communication

USART: **Universal Synchronous/Asynchronous Receiver/Transmitter**. It can automatically generate data frame timing based on one byte of data from the data register and send it out through the TX pin, or automatically receive data frame timing from the RX pin and concatenate it into one byte of data, stored in the data register. This experiment uses UART2.

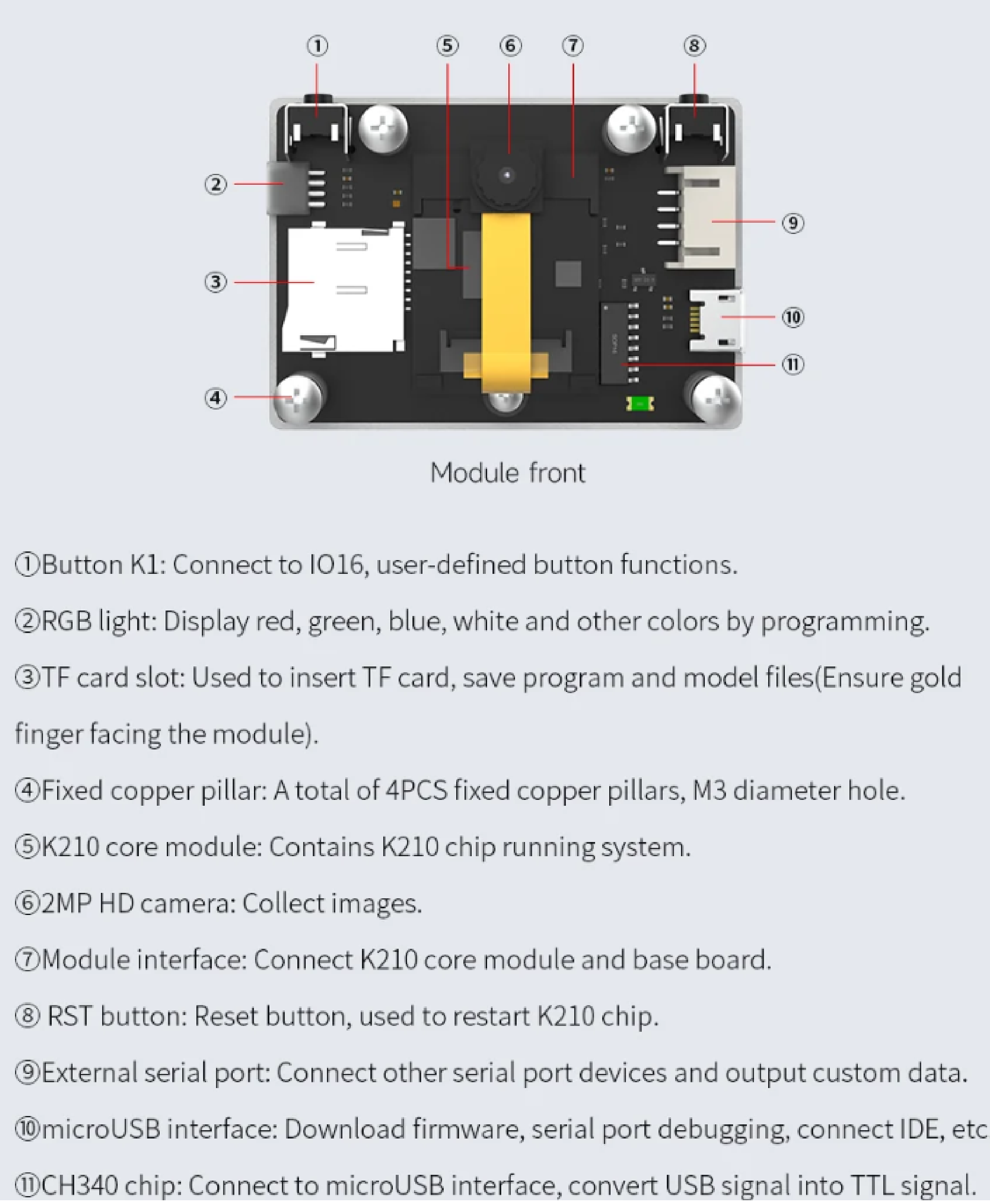


When using serial port for communication, both communication parties must be on the "same channel". "Same channel" means the same communication protocol. Serial port (USART) stipulates: data must be transmitted in the form of "frames" during communication. One frame of serial port data includes: start bit + data bit + parity bit + stop bit. Where: 1) Start bit: fixed is 1 cycle low level signal 2) Data bit: can be agreed by both parties as 5 ~ 9 bits 3) Parity bit: serial port uses parity check, can be agreed by both parties 4) Stop bit: optional 0.5 ~ 2 cycles high level. At the same time, to synchronize the transmission speed of both parties, it is also necessary to agree on the number of data frames transmitted per second, called **baud rate**. Typical baud rates include 9600, 115200, 57600...

2.2 k210 Vision Module

Kanzhi K210 is a system-on-chip that integrates machine vision and machine hearing capabilities. It uses TSMC's ultra-low power 28nm advanced process, has a dual-core 64-bit processor with total computing power up to 1TOPS, built-in various hardware acceleration units (KPU, FPU, FFT, etc.), and has good power performance, stability and reliability.

Function distribution is as follows:



Module front

①Button K1: Connect to IO16, user-defined button functions.

②RGB light: Display red, green, blue, white and other colors by programming.

③TF card slot: Used to insert TF card, save program and model files(Ensure gold finger facing the module).

④Fixed copper pillar: A total of 4PCS fixed copper pillars, M3 diameter hole.

⑤K210 core module: Contains K210 chip running system.

⑥2MP HD camera: Collect images.

⑦Module interface: Connect K210 core module and base board.

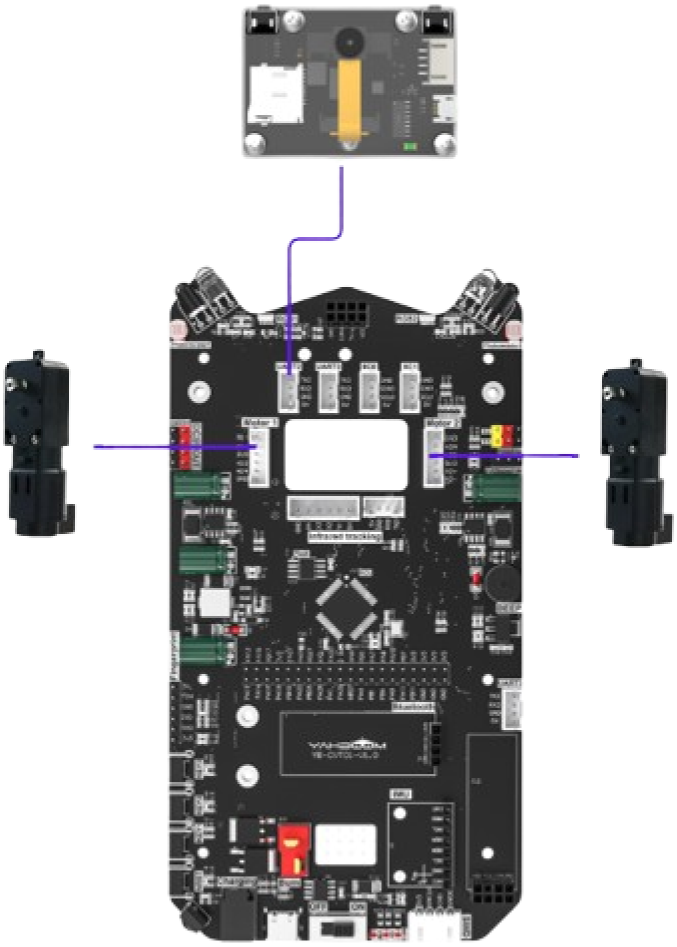
⑧ RST button: Reset button, used to restart K210 chip.

⑨External serial port: Connect other serial port devices and output custom data.

⑩microUSB interface: Download firmware, serial port debugging, connect IDE, etc

⑪CH340 chip: Connect to microUSB interface, convert USB signal into TTL signal.

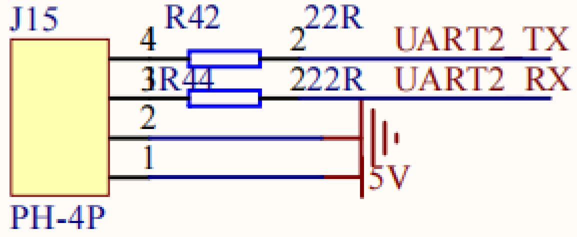
2.3 Hardware Connection



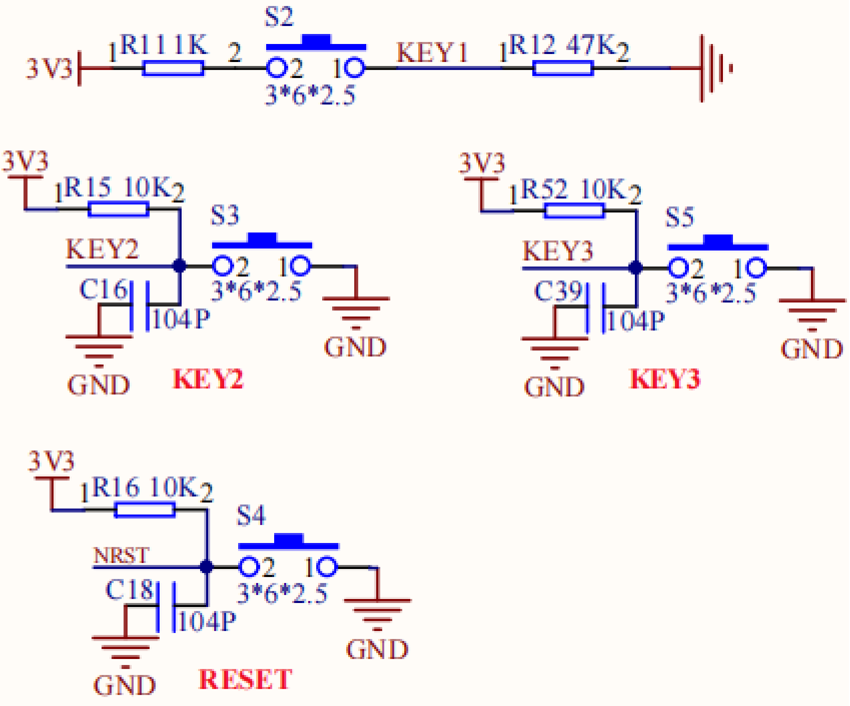
K210 Vision Module	MSPM0G3507
5V	5V
GND	GND
TX	RX2
RX	TX2

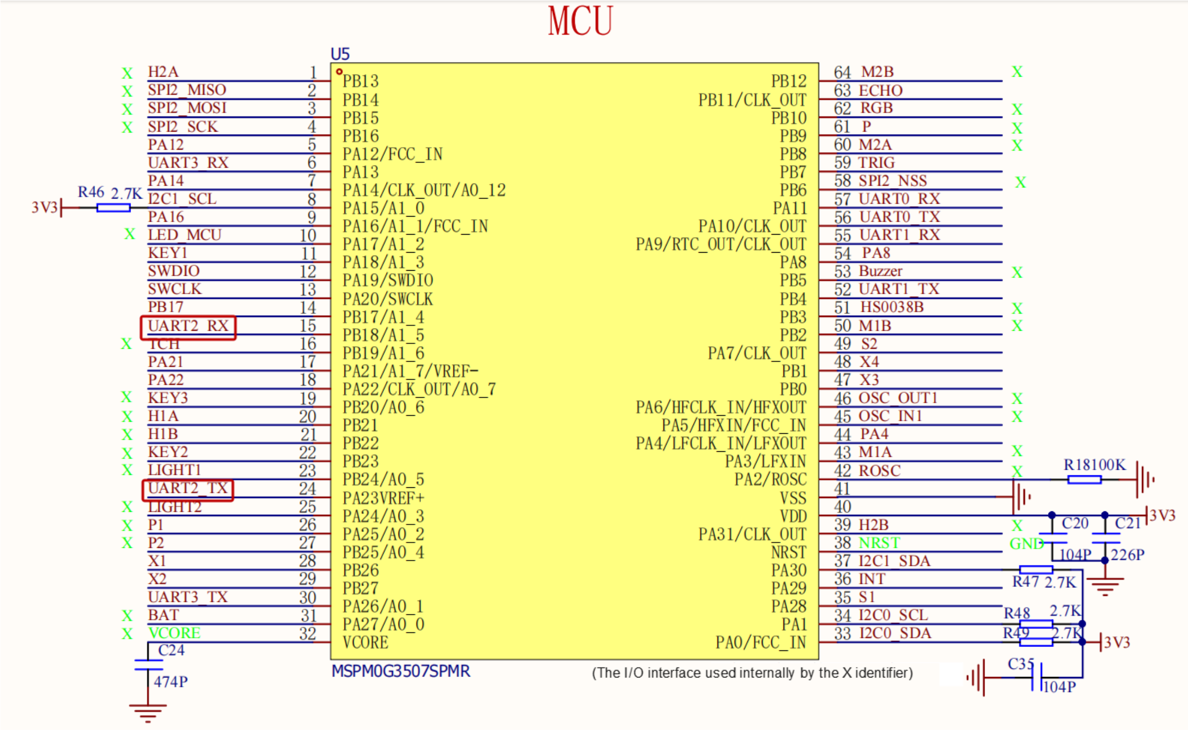
UART 2

Vision module interface
K210/K230 module interface



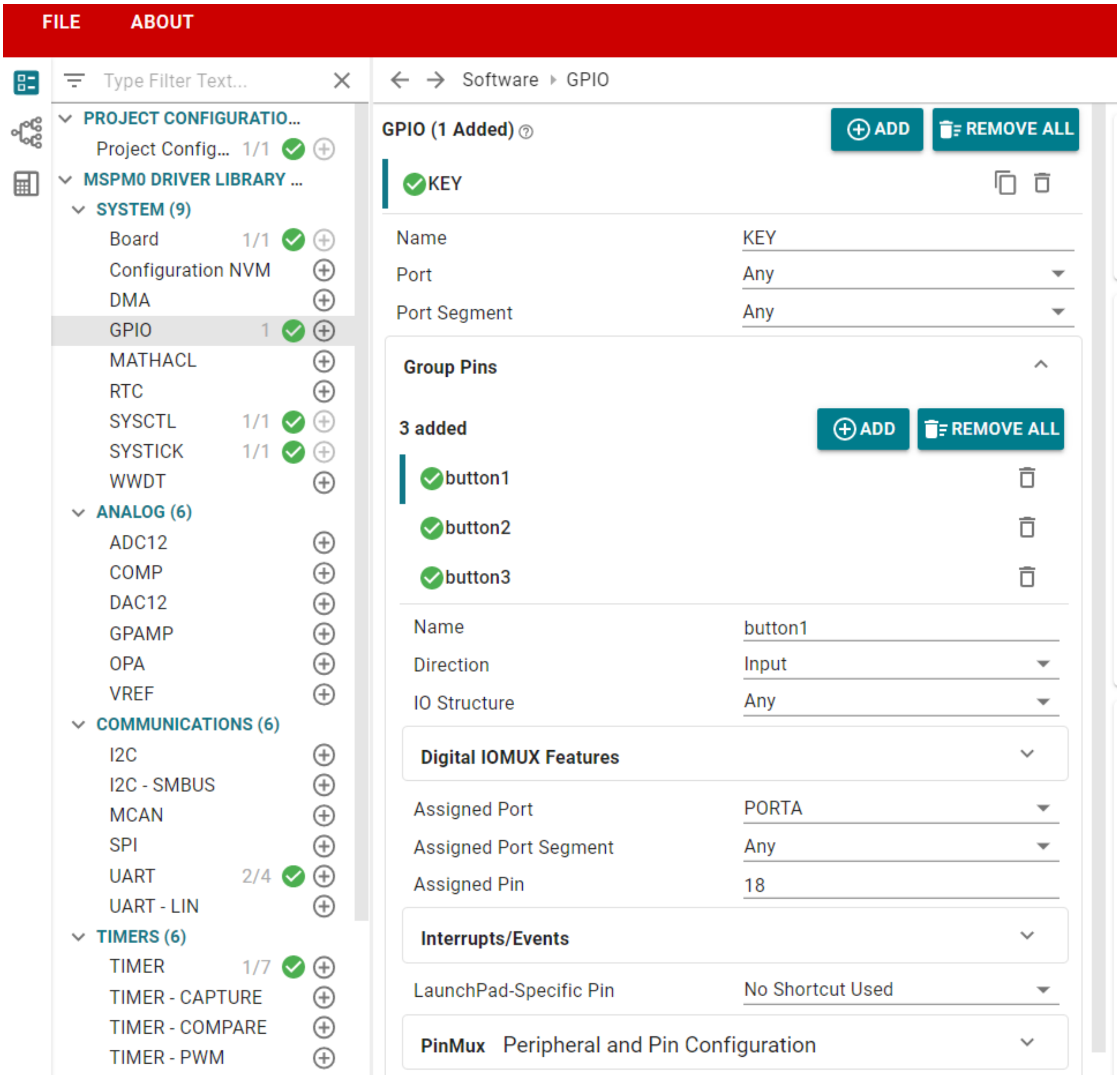
Function Key





2.4 Environment Setup

We open the sysconfig tool and add configuration as follows



GPIO (1 Added)

KEY

Name: KEY
Port: Any
Port Segment: Any

Group Pins

3 added

button1
button2
button3

Name: button1
Direction: Input
IO Structure: Any

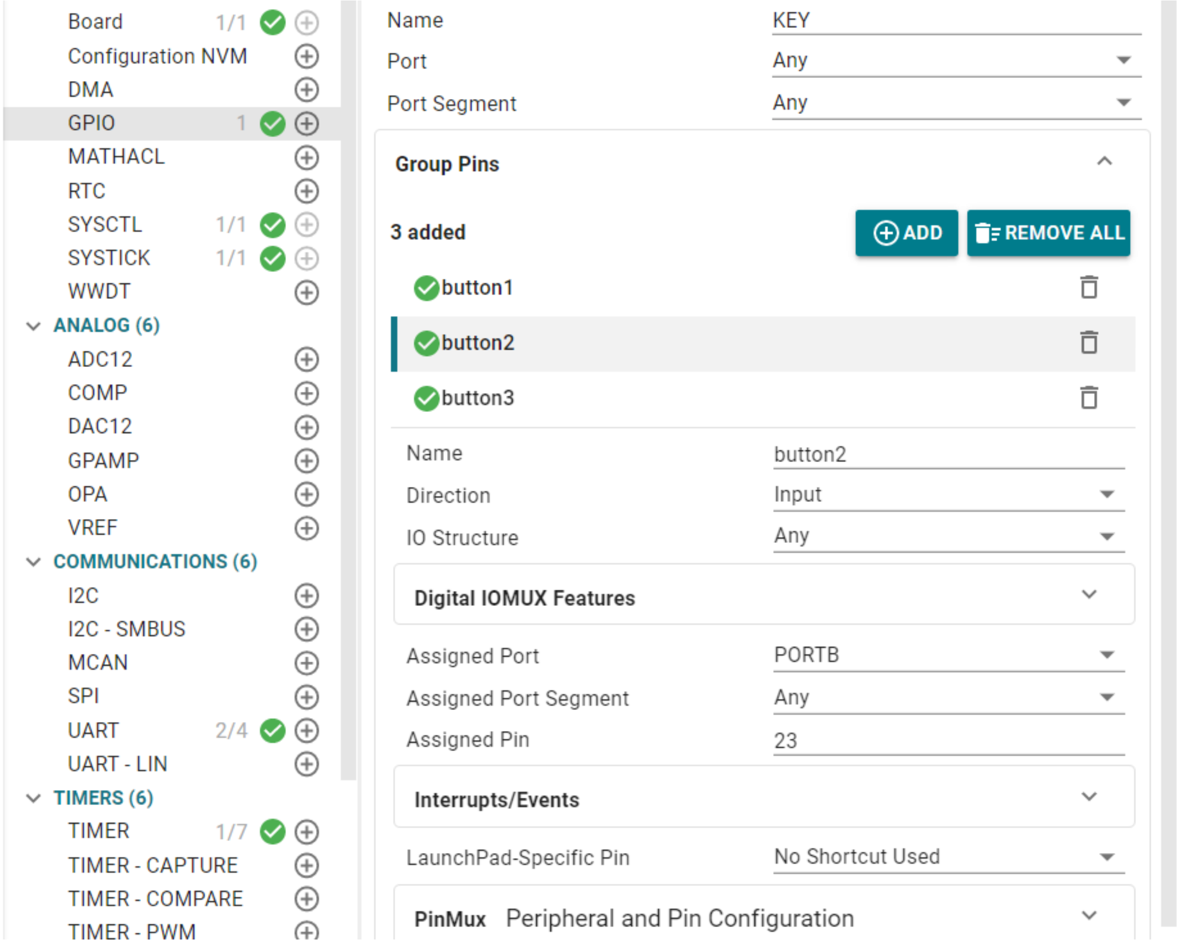
Digital IOMUX Features

Assigned Port: PORTA
Assigned Port Segment: Any
Assigned Pin: 18

Interrupts/Events

LaunchPad-Specific Pin: No Shortcut Used

PinMux Peripheral and Pin Configuration



Board 1/1 ✓ ⊕
Configuration NVM ⊕
DMA ⊕
GPIO 1 ✓ ⊕
MATHACL ⊕
RTC ⊕
SYSCTL 1/1 ✓ ⊕
SYSTICK 1/1 ✓ ⊕
WWDT ⊕
ANALOG (6)
ADC12 ⊕
COMP ⊕
DAC12 ⊕
GPAMP ⊕
OPA ⊕
VREF ⊕
COMMUNICATIONS (6)
I2C ⊕
I2C - SMBUS ⊕
MCAN ⊕
SPI ⊕
UART 2/4 ✓ ⊕
UART - LIN ⊕
TIMERS (6)
TIMER 1/7 ✓ ⊕
TIMER - CAPTURE ⊕
TIMER - COMPARE ⊕
TIMER - PWM ⊕

Name KEY
Port Any
Port Segment Any

Group Pins

3 added ⊕ ADD REMOVE ALL

✓ button1
✓ button2
✓ button3

Name button2
Direction Input
IO Structure Any

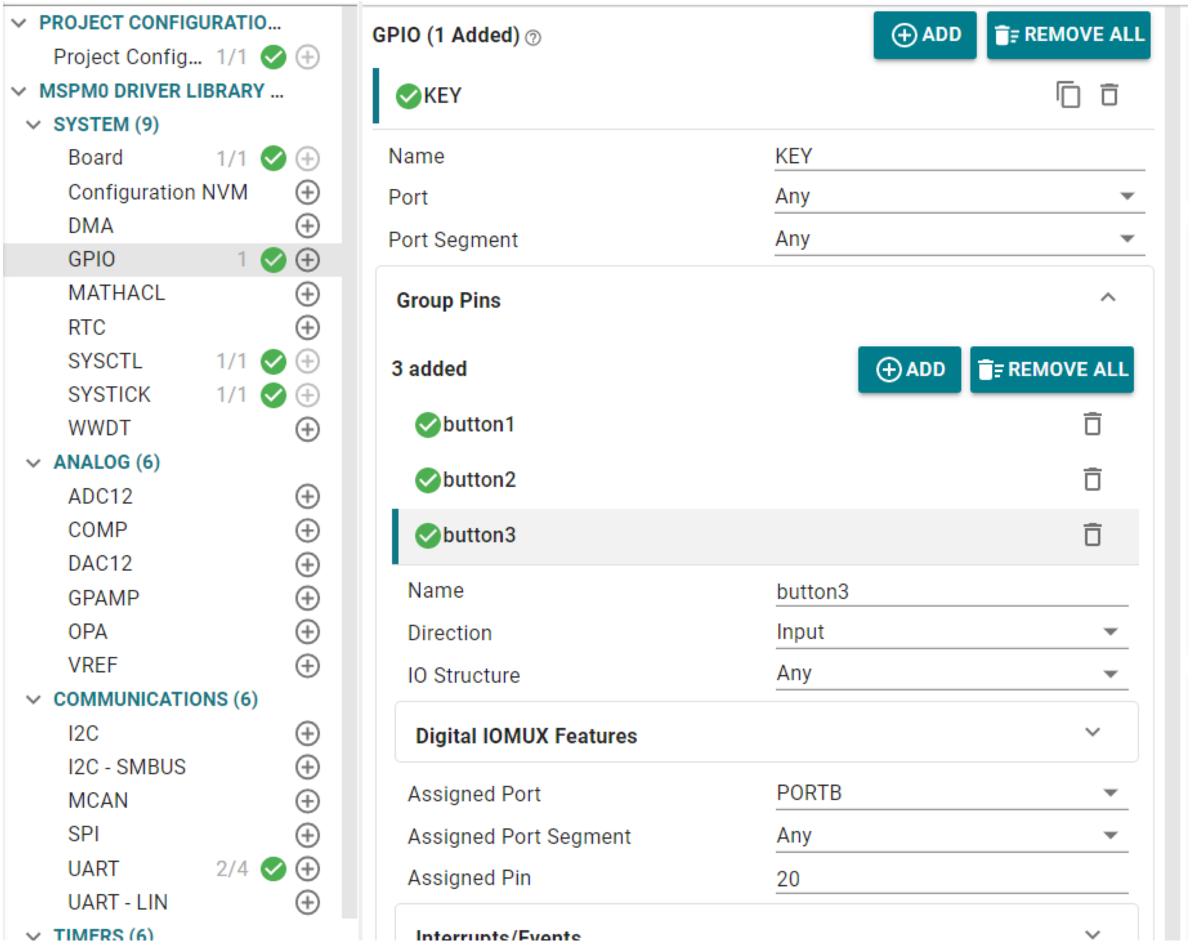
Digital IOMUX Features

Assigned Port PORTB
Assigned Port Segment Any
Assigned Pin 23

Interrupts/Events

LaunchPad-Specific Pin No Shortcut Used

PinMux Peripheral and Pin Configuration



PROJECT CONFIGURATIO...
Project Config... 1/1 ✓ ⊕
MSPM0 DRIVER LIBRARY ...
SYSTEM (9)
Board 1/1 ✓ ⊕
Configuration NVM ⊕
DMA ⊕
GPIO 1 ✓ ⊕
MATHACL ⊕
RTC ⊕
SYSCTL 1/1 ✓ ⊕
SYSTICK 1/1 ✓ ⊕
WWDT ⊕
ANALOG (6)
ADC12 ⊕
COMP ⊕
DAC12 ⊕
GPAMP ⊕
OPA ⊕
VREF ⊕
COMMUNICATIONS (6)
I2C ⊕
I2C - SMBUS ⊕
MCAN ⊕
SPI ⊕
UART 2/4 ✓ ⊕
UART - LIN ⊕
TIMERS (6)

GPIO (1 Added) ? ⊕ ADD REMOVE ALL

✓ KEY

Name KEY
Port Any
Port Segment Any

Group Pins

3 added ⊕ ADD REMOVE ALL

✓ button1
✓ button2
✓ button3

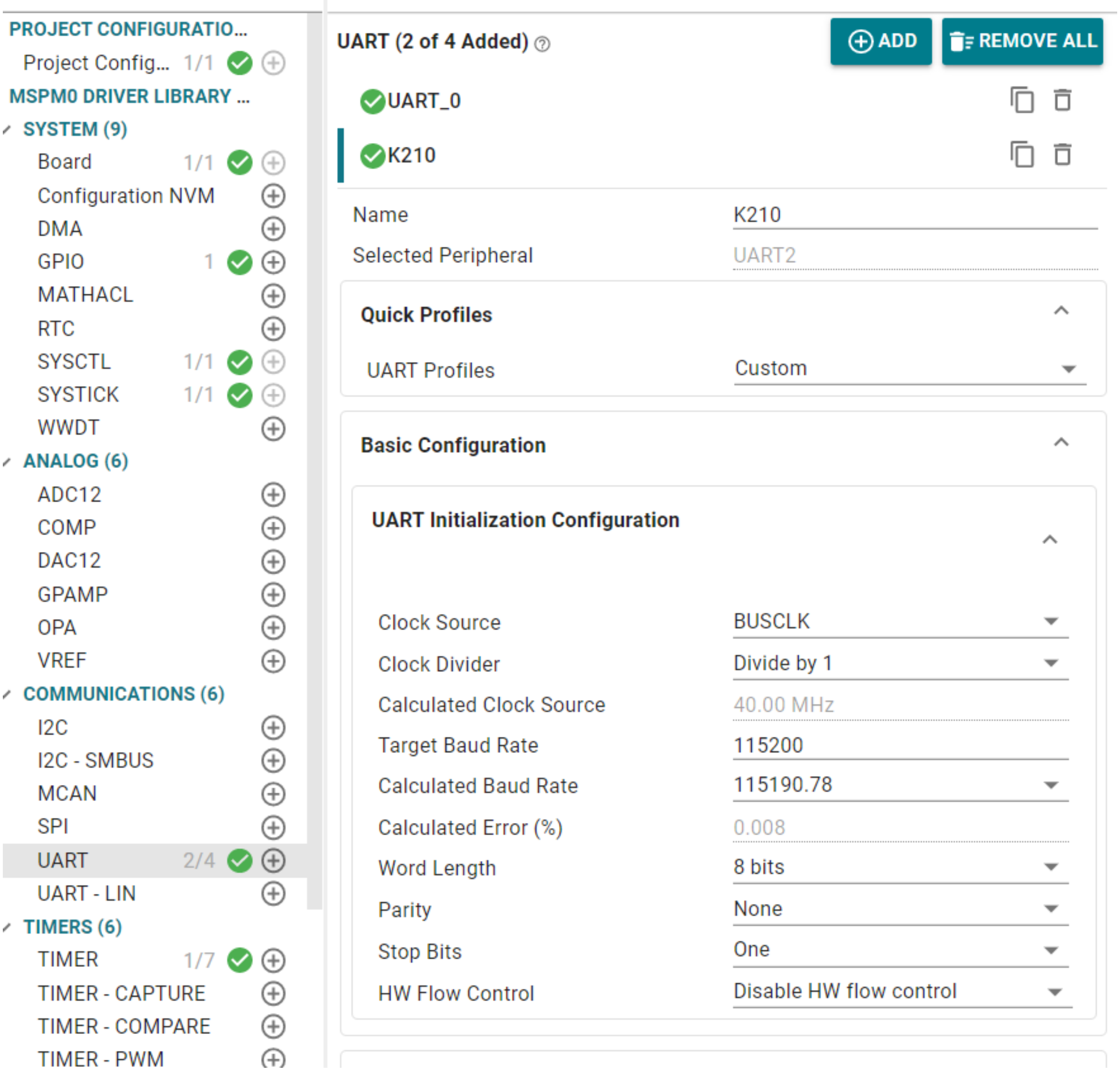
Name button3
Direction Input
IO Structure Any

Digital IOMUX Features

Assigned Port PORTB
Assigned Port Segment Any
Assigned Pin 20

Interrupts/Events

K210 serial port configuration is as follows



UART (2 of 4 Added)

UART_0

K210

Name	K210
Selected Peripheral	UART2

Quick Profiles

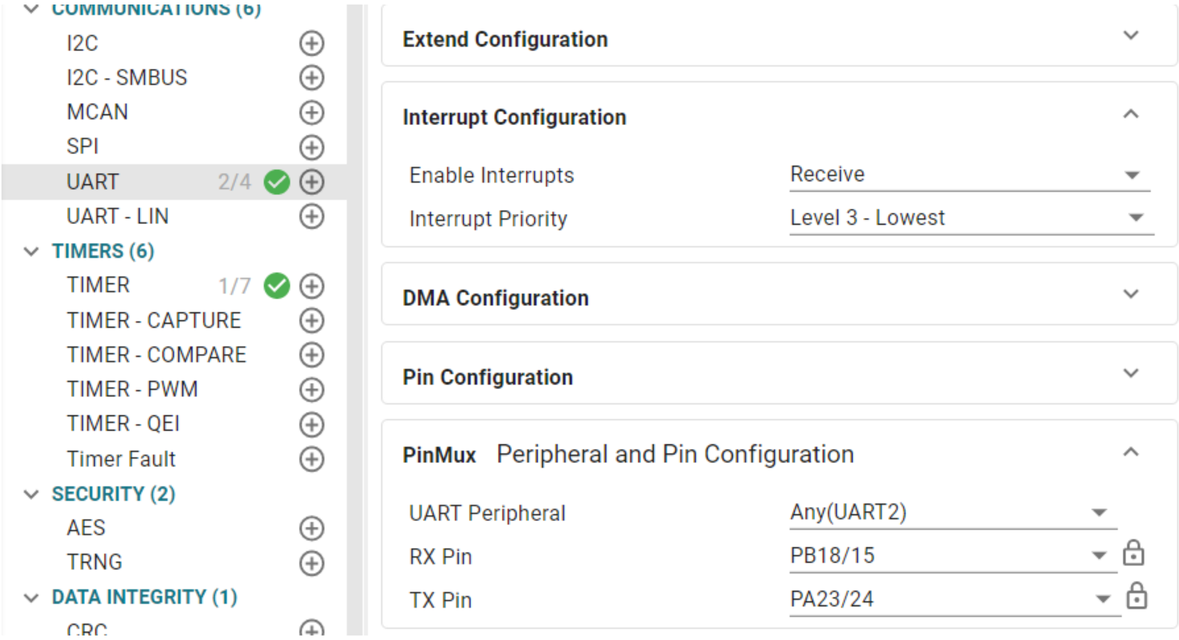
UART Profiles	Custom

Basic Configuration

UART Initialization Configuration

Clock Source	BUSCLK
Clock Divider	Divide by 1
Calculated Clock Source	40.00 MHz
Target Baud Rate	115200
Calculated Baud Rate	115190.78
Calculated Error (%)	0.008
Word Length	8 bits
Parity	None
Stop Bits	One
HW Flow Control	Disable HW flow control

Enable receive interrupt



Extend Configuration

Interrupt Configuration

Enable Interrupts	Receive
Interrupt Priority	Level 3 - Lowest

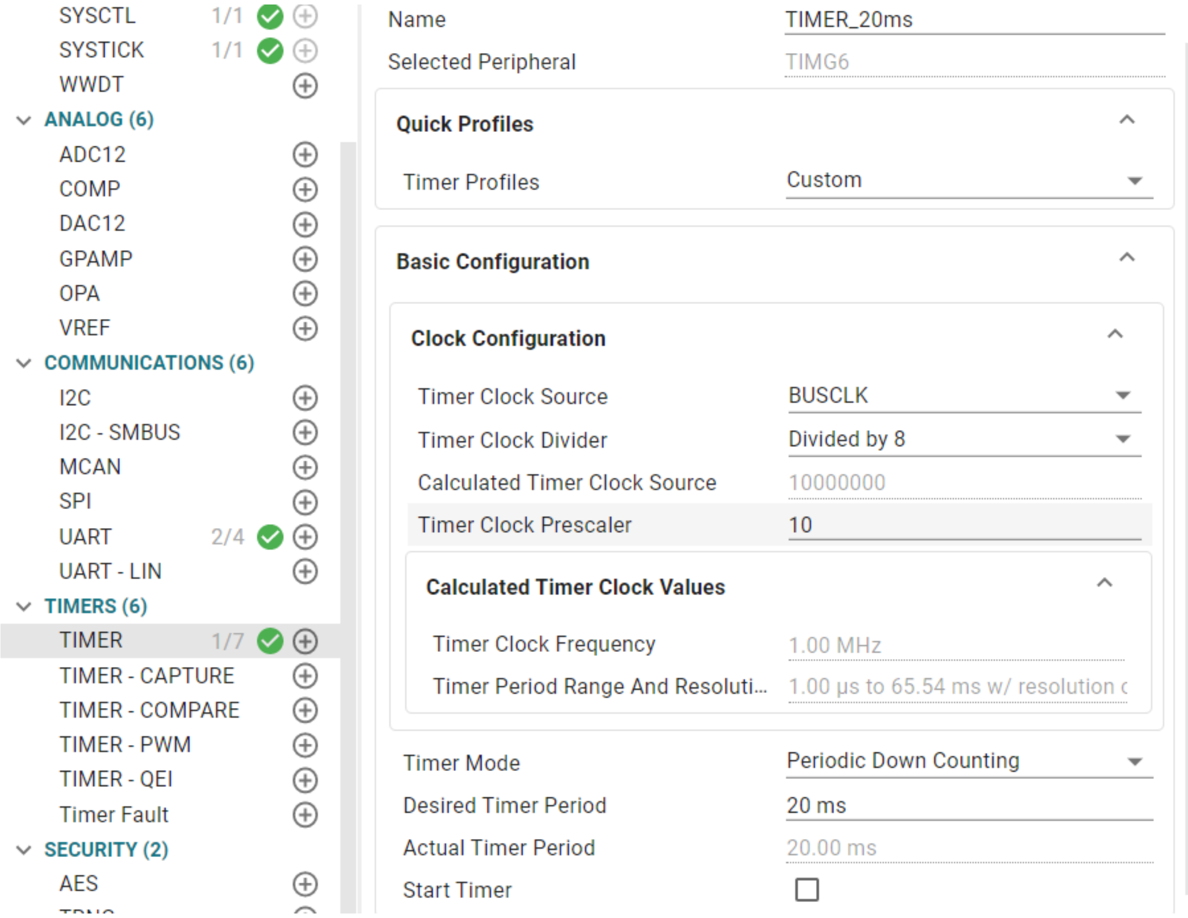
DMA Configuration

Pin Configuration

PinMux Peripheral and Pin Configuration

UART Peripheral	Any(UART2)
RX Pin	PB18/15
TX Pin	PA23/24

Add timer configuration for button detection



Name: TIMER_20ms

Selected Peripheral: TIMG6

Quick Profiles

Timer Profiles: Custom

Basic Configuration

Clock Configuration

Timer Clock Source: BUSCLK

Timer Clock Divider: Divided by 8

Calculated Timer Clock Source: 10000000

Timer Clock Prescaler: 10

Calculated Timer Clock Values

Timer Clock Frequency: 1.00 MHz

Timer Period Range And Resoluti...: 1.00 µs to 65.54 ms w/ resolution c

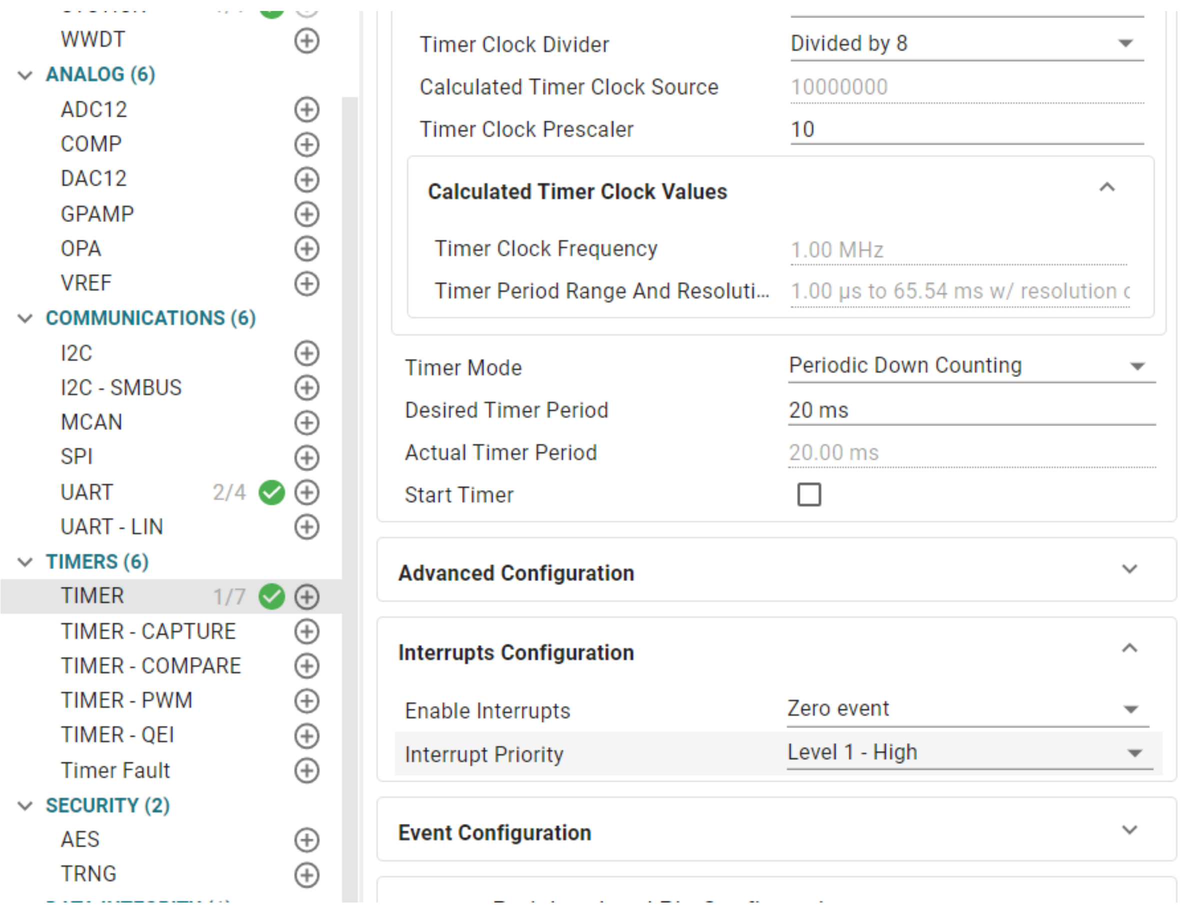
Timer Mode: Periodic Down Counting

Desired Timer Period: 20 ms

Actual Timer Period: 20.00 ms

Start Timer: □

Enable zero event interrupt



Timer Clock Divider: Divided by 8

Calculated Timer Clock Source: 10000000

Timer Clock Prescaler: 10

Calculated Timer Clock Values

Timer Clock Frequency: 1.00 MHz

Timer Period Range And Resoluti...: 1.00 µs to 65.54 ms w/ resolution c

Timer Mode: Periodic Down Counting

Desired Timer Period: 20 ms

Actual Timer Period: 20.00 ms

Start Timer: □

Advanced Configuration

Interrupts Configuration

Enable Interrupts: Zero event

Interrupt Priority: Level 1 - High

Event Configuration

2.5 Partial Code Explanation

bsp_key.c

```
#include "bsp_key.h"

// Define global button handle array, storing state information for each button
KeyHandle_T keyHandle[KEY_NUM] = {0};
uint8_t g_key_flag = 0;

// Button initialization function
void Key_Init(void) {
    for (int i = 0; i < KEY_NUM; i++) {
        keyHandle[i].state = KEY_STATE_IDLE;

        keyHandle[i].press_time = 0;

        keyHandle[i].key_event_short = 0;

        keyHandle[i].key_event_long = 0;

    }
}

// Read physical state of a single button
uint8_t Key_Read_Pin(uint8_t key_index) {
    // Read different pins based on button index
    switch (key_index) {
        case 0:
            // Read button 1 pin state, return pressed/released state based on
hardware connection
            return (DL_GPIO_readPins(KEY_button1_PORT, KEY_button1_PIN) &
KEY_button1_PIN) ?  KEY_PRESSED : KEY_RELEASED ;
        case 1:
            // Read button 2 pin state
            return (DL_GPIO_readPins(KEY_button2_PORT, KEY_button2_PIN) &
KEY_button2_PIN) ? KEY_RELEASED : KEY_PRESSED;
        case 2:
            // Read button 3 pin state
            return (DL_GPIO_readPins(KEY_button3_PORT, KEY_button3_PIN) &
KEY_button3_PIN) ? KEY_RELEASED : KEY_PRESSED;
        default:
            return KEY_RELEASED;                      // Return released state
by default
                                                     // Return released state by
default
    }
}

// Button scan handler function (needs to be called in 20ms timer interrupt)
void Key_Scan_Handler(void) {
    for (int i = 0; i < KEY_NUM; i++) {
        uint8_t current_key_state = Key_Read_Pin(i);


        switch (keyHandle[i].state) {
            case KEY_STATE_IDLE:
                if (current_key_state == KEY_PRESSED) {
                    // Enter debounce state
                    keyHandle[i].state = KEY_STATE_DEBOUNCE;

                    keyHandle[i].press_time = 0;

                }
                break;

            case KEY_STATE_DEBOUNCE:

                if (current_key_state == KEY_PRESSED) {
                    keyHandle[i].press_time++;

                    // Debounce time: 20ms (1 cycle), continue pressing to confirm

                    if (keyHandle[i].press_time >= 1) {
                        keyHandle[i].state = KEY_STATE_PRESSED;

                        keyHandle[i].key_event_short = 1;

                        keyHandle[i].press_time = 0;

                    }
                } else {
                    // Released during debounce, considered as jitter, return to idle state
                    keyHandle[i].state = KEY_STATE_IDLE;
                }
                break;

            case KEY_STATE_PRESSED:
                if (current_key_state == KEY_PRESSED) {
                    keyHandle[i].press_time++;

                    // Long press judgment, for example 50 * 20ms = 1000ms (1 second)

                    if (keyHandle[i].press_time >= 50) {
                        keyHandle[i].key_event_long = 1;  // Trigger long press flag

                    }
                } else {
                    // Release detected, enter release debounce state
                    keyHandle[i].state = KEY_STATE_RELEASE;
                    keyHandle[i].press_time = 0;
                }
                break;

            case KEY_STATE_RELEASE:
                 // Release debounce state
                keyHandle[i].press_time++;

                // Release debounce, continue for 20ms (1 cycle) to consider stable release

                if (keyHandle[i].press_time >= 1) {
                    keyHandle[i].state = KEY_STATE_IDLE;

                    // Note: Short press and long press flags are cleared after main loop query, not cleared here

                }
                break;

            default:
                // Abnormal state reset to idle
                keyHandle[i].state = KEY_STATE_IDLE;

                break;
        }
    }
}
```

bsp_k210_usart.c

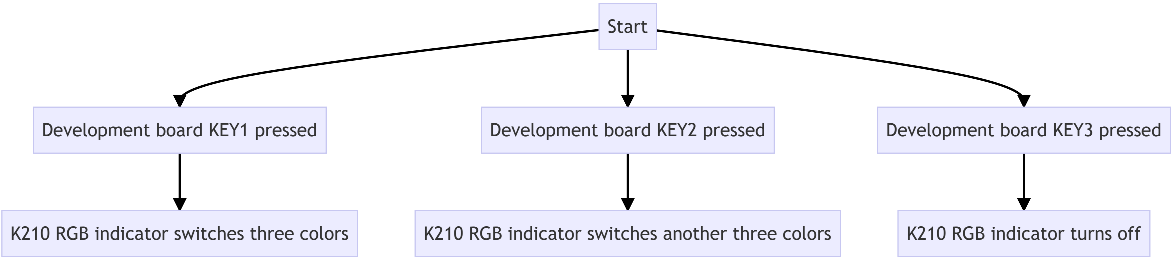
```
// Use button, press once to change k210's RGB color
void K210_Rgb_Key(void)
{
    // If button 0 short press event is triggered
    if(keyHandle[0].key_event_short == 1)
    {
        // Switch sending different color commands based on send_key1 value
        switch (send_key1)
        {
            case 0:
                 // Send blue command
                strcpy(send_buf, "$blue#");
                break;
            case 1:
                // Send red command
                strcpy(send_buf, "$red#");
                break;
            case 2:
                // Send green command
                strcpy(send_buf, "$green#");

                break;
            default:
                send_key1 = 0;               // Default value reset to 0

        }
        send_key1 = (send_key1 + 1) % 3;
        uart2_send_string(send_buf);
        keyHandle[0].key_event_short = 0;

    }

    // If button 1 short press event is triggered
    if(keyHandle[1].key_event_short == 1)
    {
        // Switch sending different color commands based on send_key2 value
        switch (send_key2)
        {
            case 0:
                // Send yellow command
                strcpy(send_buf, "$yellow#");

                break;
            case 1:
                // Send purple command (Note: correct spelling should be purple)
                strcpy(send_buf, "$purple#");
                break;
            case 2:
                strcpy(send_buf, "$lake#");

                break;
            default:
                send_key2 = 0;

        }
        send_key2 = (send_key2 + 1) % 3;

        uart2_send_string(send_buf);

        keyHandle[1].key_event_short = 0;

    }

    // If button 2 short press event is triggered
    if(keyHandle[2].key_event_short == 1)
    {
         // Send command to turn off RGB
        strcpy(send_buf, "$close#");
        // Send close command through UART2
        uart2_send_string(send_buf);
         // Clear button 2 short press event flag

        keyHandle[2].key_event_short = 0;

    }
}
```

K210 partial source code

```
while True:
    # Read data from serial port
    read_data = uart_A.read()
    # Judge whether valid data is read
    if read_data is not None:
     ...
        # Filter out invalid 0xFF bytes
        if read_data[0] != 255:
            # Decode byte data as UTF-8 string
            read_str = read_data.decode('utf-8')
            # Remove null characters from string (STM32 might send packets with
null characters)
            read_str = read_str.replace('\x00', '')

            # Check if string matches "$xxx#" format (starts with $, ends with
#)
```

```
            if read_str.startswith('$') and read_str.endswith('#'):
                # Extract valid data between $ and # (remove special symbols at start and end)
                recv_data = read_str.strip('$#')
                # Set RGB color based on received command
                set_RGB(recv_data)

# Release serial port resources when program exits (actual运行时while True会一直循环，此处为收尾操作)
uart_A.deinit()
# Delete serial port object
# Delete UART object
del uart_A
```

Program Flowchart


Start
Development board KEY1 pressed
Development board KEY2 pressed
Development board KEY3 pressed
K210 RGB indicator switches three colors
K210 RGB indicator switches another three colors
K210 RGB indicator turns off

2.6 Main Functions

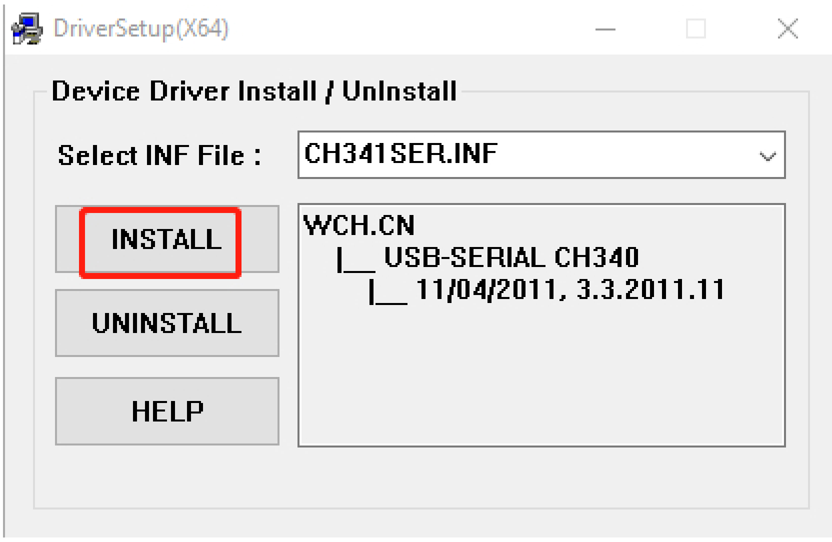
K210_Rgb_Key

Function Prototype	**void K210_Rgb_Key()**
Function Description	**Control the state of RGB light on k210**
Input Parameters	**None**
Output Parameters	**None**

2.7 K210 Program Burning

Download all files in the [Program Source Code Summary] -> [Vision Application Experiment] -> [K210_SD_File] directory to the root directory of the memory card through card reader.

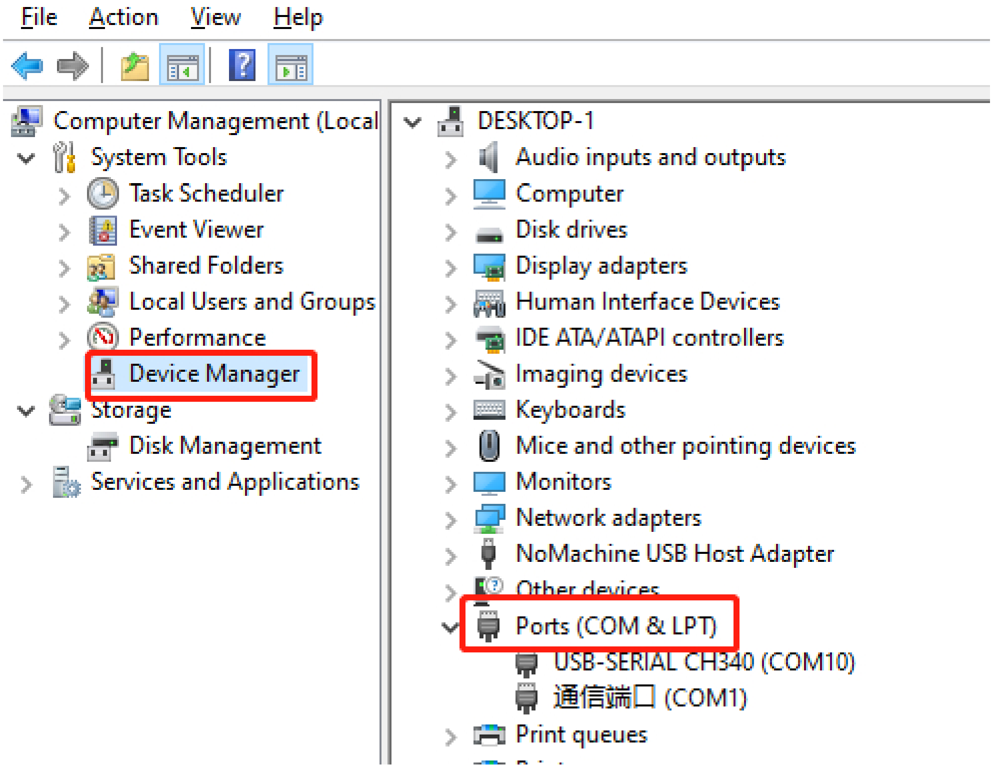
We can click on the tutorial link and select [Download] -> [Development Software] to access Google Drive and download the drivers and CanMV IDE .



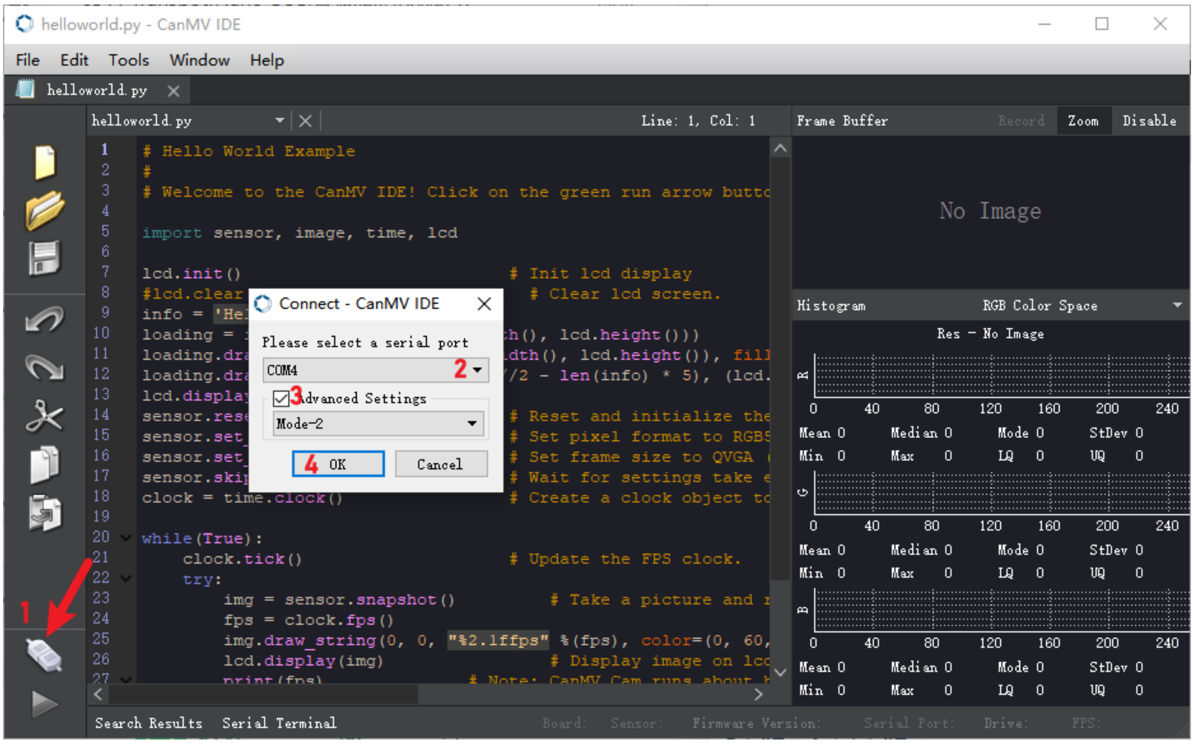
Select INF File : CH341SER.INF

```
WCH.CN
  |__ USB-SERIAL CH340
     |__ 11/04/2011, 3.3.2011.11
```

After successful installation, we connect microusb to K210 and computer. Open Device Manager and see similar device as shown below, which means our driver installation is successful. The COMX here depends on your actual situation, then we open CanMV IDE



After downloading and opening CanMV IDE, we need to first drag the k210 source code provided in this course section k210_car.py to CanMV IDE to open, then connect IDE, **here take helloworld.py as example**



Connect - CanMV IDE

Please select a serial port

COM4 2

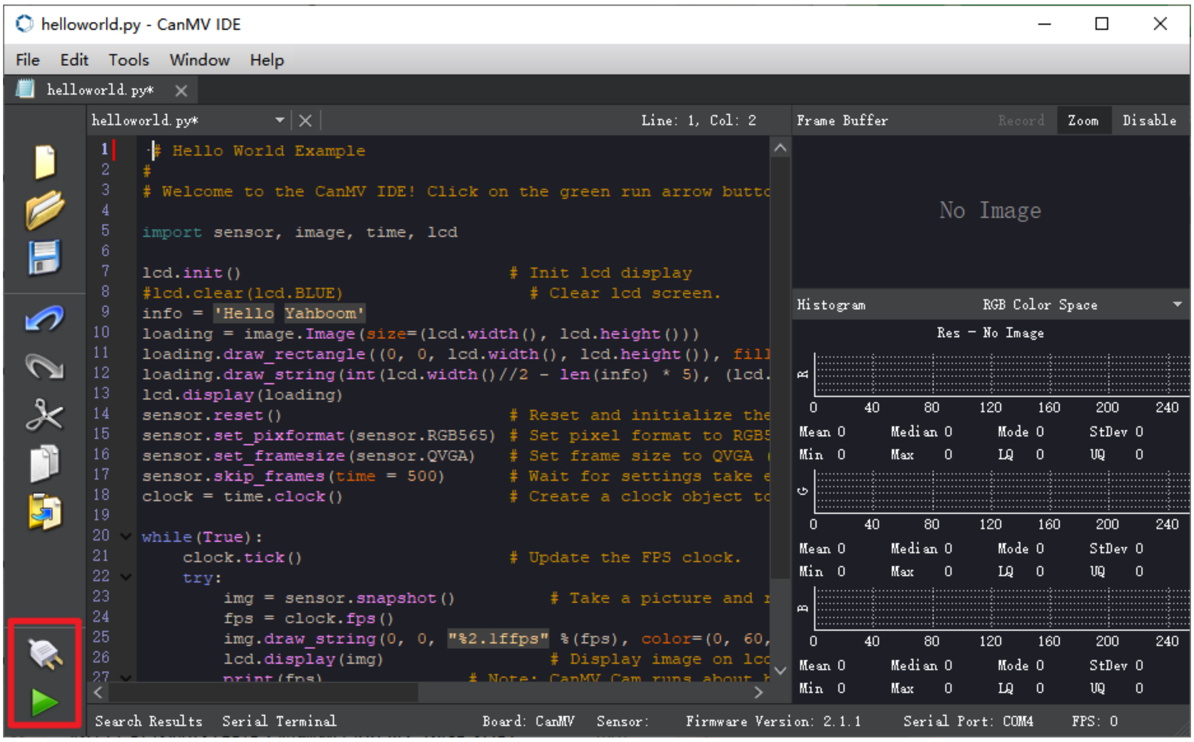
☑ 3 Advanced Settings

Mode-2

4 OK　Cancel

```
# Hello World Example
#
# Welcome to the CanMV IDE! Click on the green run arrow butto

import sensor, image, time, lcd

lcd.init()                              # Init lcd display
#lcd.clear                                # Clear lcd screen.
info = 'He
loading = i                             h(), lcd.height()))
loading.dra                             dth(), lcd.height()), fill
loading.dra                             /2 - len(info) * 5), (lcd.
lcd.display
sensor.rese                             # Reset and initialize the
sensor.set_                             # Set pixel format to RGB5
sensor.set_                             # Set frame size to QVGA (
sensor.skip                             # Wait for settings take e
clock = time.clock()                    # Create a clock object to

while(True):
    clock.tick()                        # Update the FPS clock.
    try:
        img = sensor.snapshot()             # Take a picture and r
        fps = clock.fps()
        img.draw_string(0, 0, "%2.1ffps" %(fps), color=(0, 60,
        lcd.display(img)                    # Display image on lcd
```

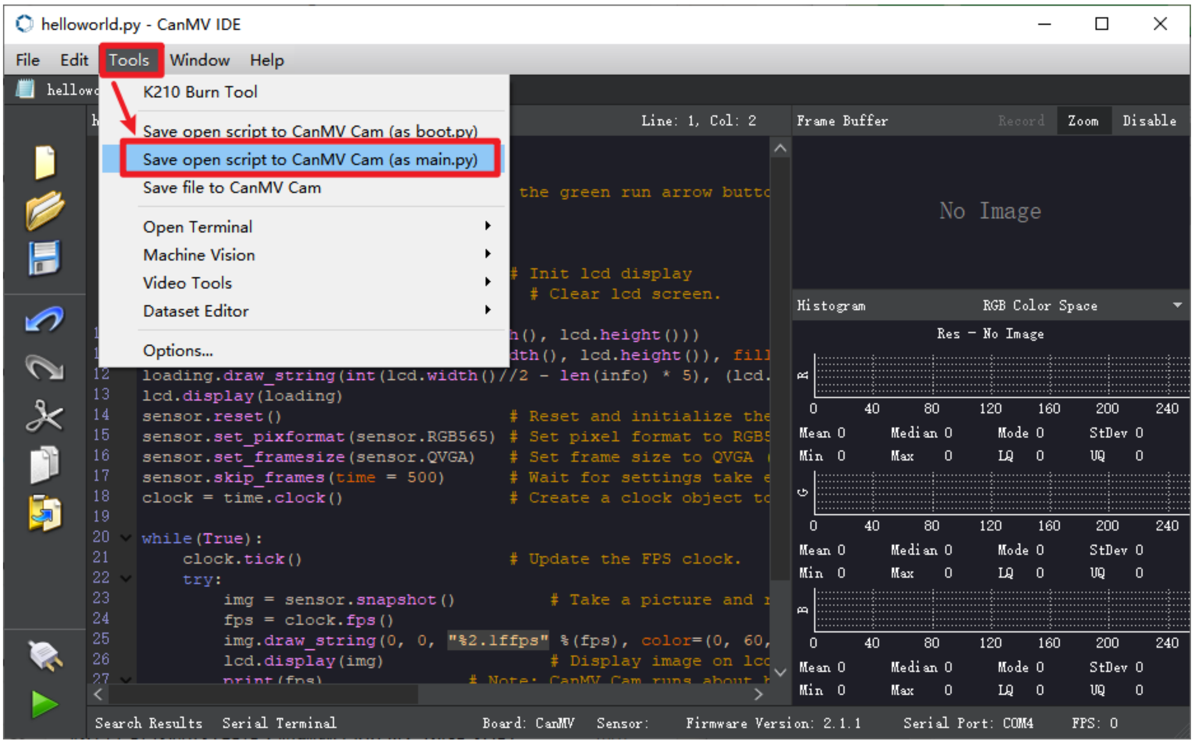
After IDE connection is successful, the phenomenon is as follows



```
# Hello World Example
#
# Welcome to the CanMV IDE! Click on the green run arrow butto

import sensor, image, time, lcd

lcd.init()                              # Init lcd display
#lcd.clear(lcd.BLUE)                      # Clear lcd screen.
info = 'Hello Yahboom'
loading = image.Image(size=(lcd.width(), lcd.height()))
loading.draw_rectangle((0, 0, lcd.width(), lcd.height()), fill
loading.draw_string(int(lcd.width()//2 - len(info) * 5), (lcd.
lcd.display(loading)
sensor.reset()                          # Reset and initialize the
sensor.set_pixformat(sensor.RGB565) # Set pixel format to RGB5
sensor.set_framesize(sensor.QVGA)   # Set frame size to QVGA (
sensor.skip_frames(time = 500)      # Wait for settings take e
clock = time.clock()                    # Create a clock object to

while(True):
    clock.tick()                        # Update the FPS clock.
    try:
        img = sensor.snapshot()             # Take a picture and r
        fps = clock.fps()
        img.draw_string(0, 0, "%2.1ffps" %(fps), color=(0, 60,
        lcd.display(img)                    # Display image on lcd
```

Board: CanMV　Sensor:　Firmware Version: 2.1.1　Serial Port: COM4　FPS: 0

Open the top menu bar Tools -> Save currently open script as (main.py) to CanMV Cam

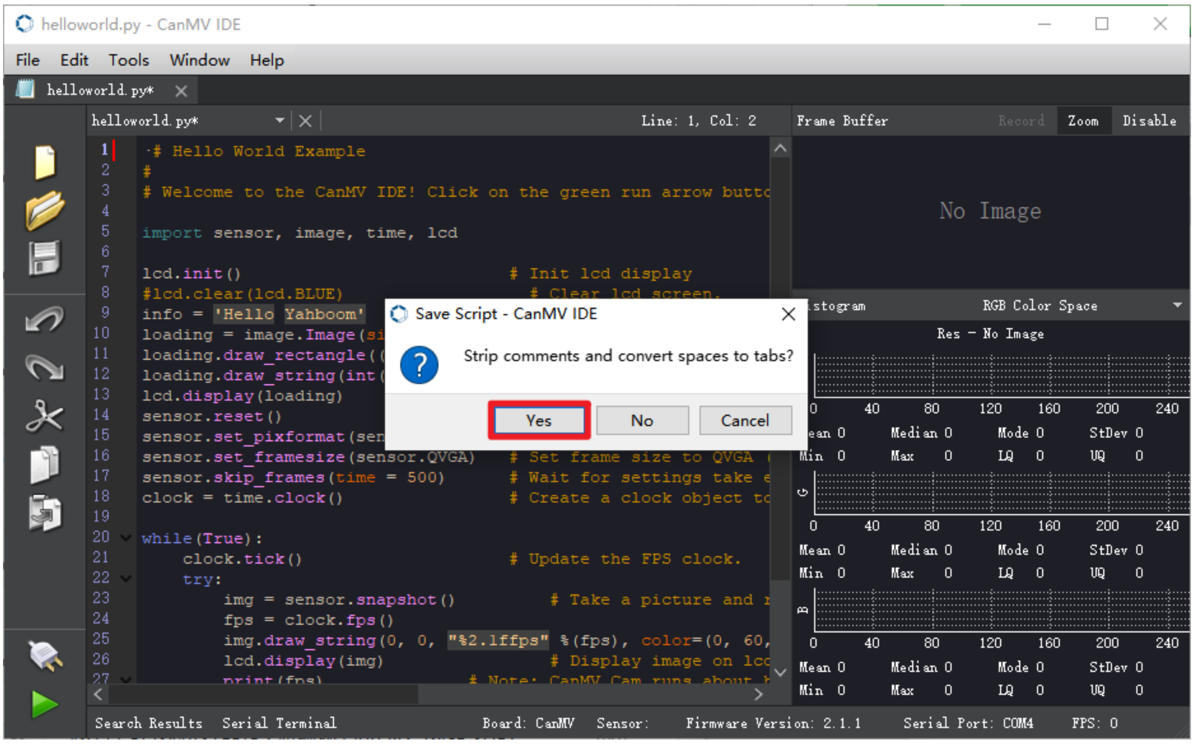


Line: 1, Col: 2

```
                                    the green run arrow butto

                                    # Init lcd display
                                      # Clear lcd screen.

                                    h(), lcd.height()))
                                    dth(), lcd.height()), fill
loading.draw_string(int(lcd.width()//2 - len(info) * 5), (lcd.
lcd.display(loading)
sensor.reset()                      # Reset and initialize the
sensor.set_pixformat(sensor.RGB565) # Set pixel format to RGB5
sensor.set_framesize(sensor.QVGA)   # Set frame size to QVGA (
sensor.skip_frames(time = 500)      # Wait for settings take e
clock = time.clock()                # Create a clock object to

while(True):
    clock.tick()                    # Update the FPS clock.
    try:
        img = sensor.snapshot()         # Take a picture and r
        fps = clock.fps()
        img.draw_string(0, 0, "%2.1ffps" %(fps), color=(0, 60,
        lcd.display(img)                # Display image on lcd
        print(fps)                  # Note: CanMV Cam runs about h
```

Mean 0 Median 0 Mode 0 StDev 0
Min 0 Max 0 LQ 0 UQ 0

Board: CanMV Sensor: Firmware Version: 2.1.1 Serial Port: COM4 FPS: 0



Line: 1, Col: 2

```
# Hello World Example
#
# Welcome to the CanMV IDE! Click on the green run arrow butto

import sensor, image, time, lcd

lcd.init()                          # Init lcd display
#lcd.clear(lcd.BLUE)                  # Clear lcd screen.
info = 'Hello Yahboom'
loading = image.Image(si
loading.draw_rectangle((
loading.draw_string(int(
lcd.display(loading)
sensor.reset()
sensor.set_pixformat(sen
sensor.set_framesize(sensor.QVGA)   # Set frame size to QVGA (
sensor.skip_frames(time = 500)      # Wait for settings take e
clock = time.clock()                # Create a clock object to

while(True):
    clock.tick()                    # Update the FPS clock.
    try:
        img = sensor.snapshot()         # Take a picture and r
        fps = clock.fps()
        img.draw_string(0, 0, "%2.1ffps" %(fps), color=(0, 60,
        lcd.display(img)                # Display image on lcd
        print(fps)                  # Note: CanMV Cam runs about h
```

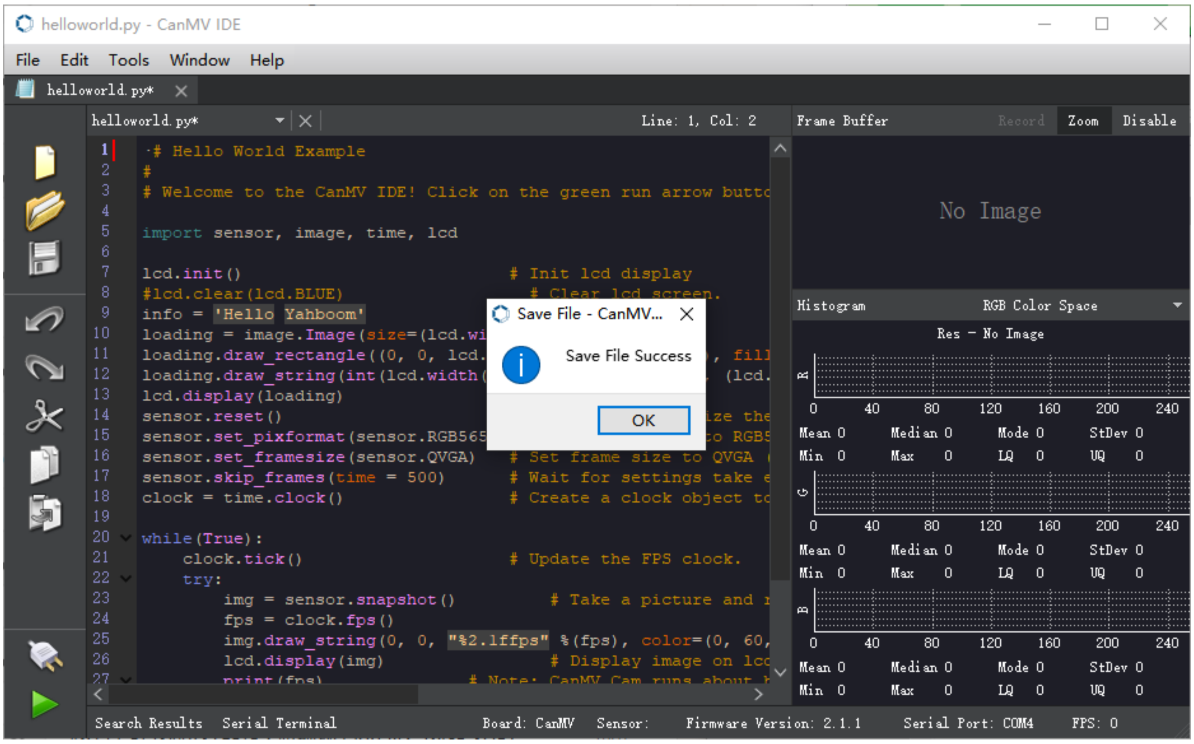
Save Script - CanMV IDE

Strip comments and convert spaces to tabs?

Mean 0 Median 0 Mode 0 StDev 0
Min 0 Max 0 LQ 0 UQ 0

Board: CanMV Sensor: Firmware Version: 2.1.1 Serial Port: COM4 FPS: 0

Here you can choose Yes/No. The following status indicates successful writing.



```
# Hello World Example
#
# Welcome to the CanMV IDE! Click on the green run arrow butto

import sensor, image, time, lcd

lcd.init()                           # Init lcd display
#lcd.clear(lcd.BLUE)                   # Clear lcd screen.
info = 'Hello Yahboom'
loading = image.Image(size=(lcd.wi
loading.draw_rectangle((0, 0, lcd.                    ), fill
loading.draw_string(int(lcd.width(                     (lcd.
lcd.display(loading)
sensor.reset()                                         ize the
sensor.set_pixformat(sensor.RGB565                     to RGB5
sensor.set_framesize(sensor.QVGA)    # Set frame size to QVGA (
sensor.skip_frames(time = 500)       # Wait for settings take e
clock = time.clock()                 # Create a clock object to

while(True):
    clock.tick()                     # Update the FPS clock.
    try:
        img = sensor.snapshot()          # Take a picture and r
        fps = clock.fps()
        img.draw_string(0, 0, "%2.1ffps" %(fps), color=(0, 60,
        lcd.display(img)                 # Display image on lcd
        print(fps)                   # Note: CanMV Cam runs about h
```

2.8 Experimental Phenomenon

After burning the program, press KEY1, KEY2, KEY3 buttons on the development board to switch the K210 indicator status once. KEY1 pressed makes K210 light up three colors, KEY2 pressed makes K210 light up another three colors, KEY3 pressed makes K210 indicator turn off.