

# Key control

---

## Key control

1. Learning objectives
2. Hardware construction
  - Independent key principle
  - Key debounce
3. Experimental steps
  1. Open the SYSCONFIG configuration tool
  2. Configure pin mode
  3. Write the program
  4. Compile
4. Program Analysis
5. Experimental phenomenon

## 1. Learning objectives

---

1. Learn the basic working principle of the keys of the MSPM0G3507 motherboard.
2. Use the key module to control the on-board LED light switch.

## 2. Hardware construction

---

### Independent key principle

Independent keys are usually implemented through **mechanical switches**, and their hardware structure consists of keys, connecting lines, and pull-up/pull-down resistors. When the key is closed, the line is connected and the signal level changes; when the key is released, the line is disconnected and the signal returns to its initial state.

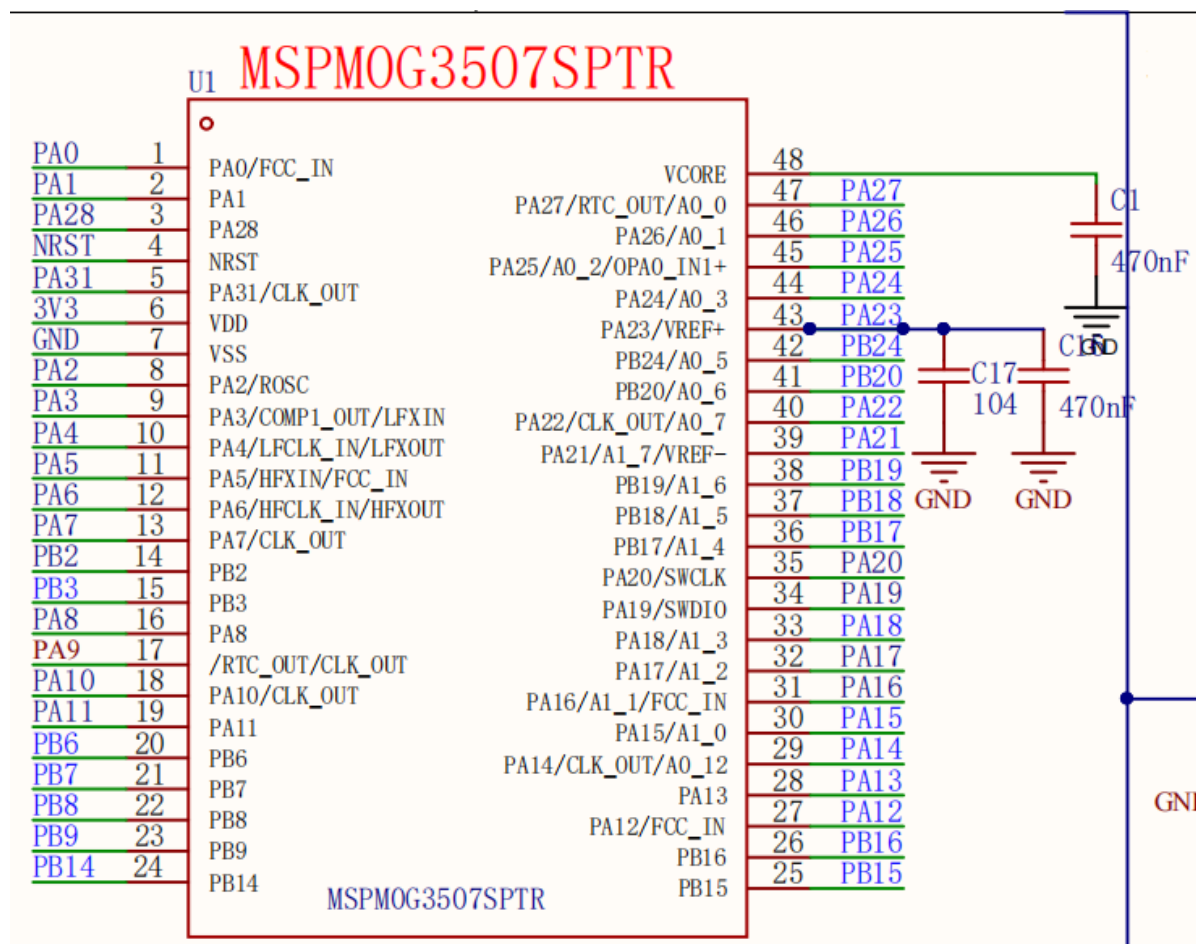
### Key debounce

The mechanical elasticity of the key will cause the signal to jump multiple times in a short period of time when the key is closed or released, which is called **key jitter**. Usually, jitter is eliminated by **software delay debounce** or hardware plus **RC filter circuit**.

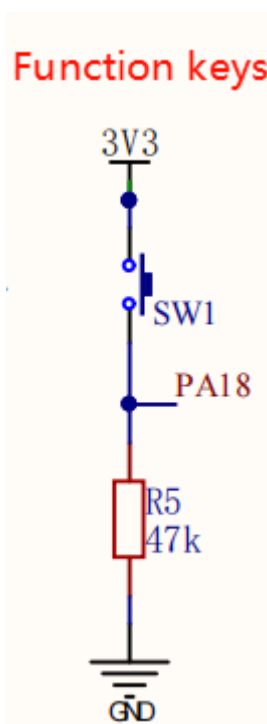
This course does not require additional hardware equipment, and directly uses the onboard function keys on the MSPM0G3507 motherboard.

In the schematic diagram of the development board, a button is connected to the PA18 pin. One end of the button is connected to a high level (3.3V), and the other end is connected to a GPIO pin. At the same time, the PA18 pin is connected to the ground through a 47kΩ pull-down resistor R5. When the button is not pressed, the PA18 pin level is pulled down to 0V due to the pull-down resistor; when the button is pressed, the PA18 pin is directly connected to a high level of 3.3V, thereby achieving a change in high and low levels. The development board can determine whether the button is pressed by detecting the level state of the PA18 pin. This design is simple and stable, and can effectively avoid signal interference caused by pin suspension.

### MSPM0G3507 main control diagram



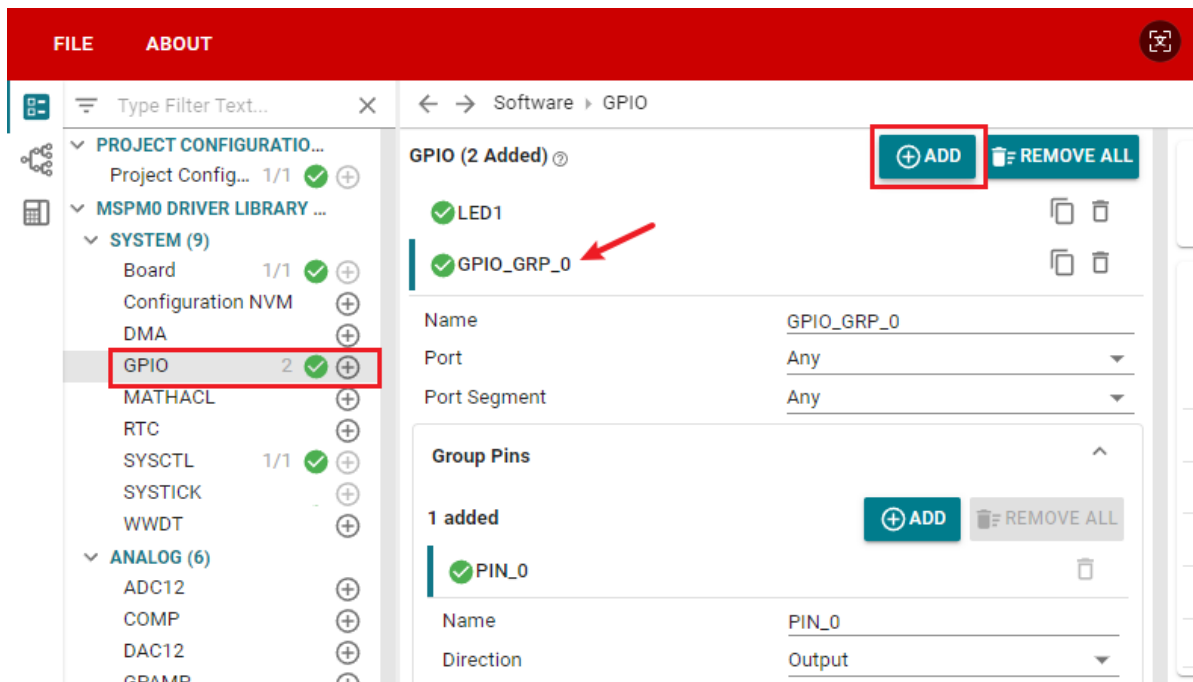
Key schematic diagram



### 3. Experimental steps

In the microcontroller, the input mode of the pin refers to configuring a pin as an input function to receive changes in external signals. The working principle of the key is based on this point: by setting the pin connected to the key to input mode, the microcontroller can detect the level change on the pin. When the key is not pressed, the pin level remains at a low level; when the key is pressed, the pin level becomes a high level. This change in high and low levels is the embodiment of the key input signal, and it is also the basis for the microcontroller to judge the

In SYSCONFIG, select MCU peripherals on the left, find the GPIO option and click to enter. Click ADD in GPIO to add a group of GPIO. This GPIO\_GRP\_0 is the newly added one.

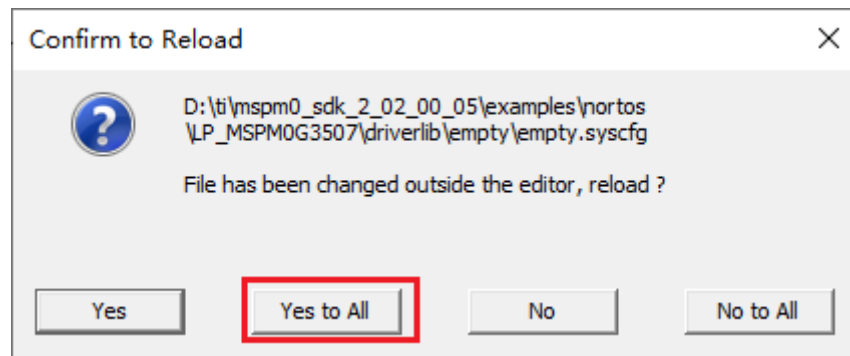


## 2. Configure pin mode

Configure pin parameters, name the GPIO instance as `KEY`, and select `PORTA` for Port because it is connected to PA18.

Set the pin name to `PIN_18` and the input mode to `Input`. Set Internal Resistor to pull-down resistor `Pull-Down Resistor`, and set the pin to be controlled in Assigned Pin to 18.





Similarly, we also need to confirm whether the `ti_msp_dl_config.c` and `ti_msp_dl_config.h` files are updated. Compile directly, and the compilation will automatically update to keil. If there is no update, we can also copy the file content in `SYSCONFIG`.

### 3. Write the program

In the `empty.c` file, write the following code

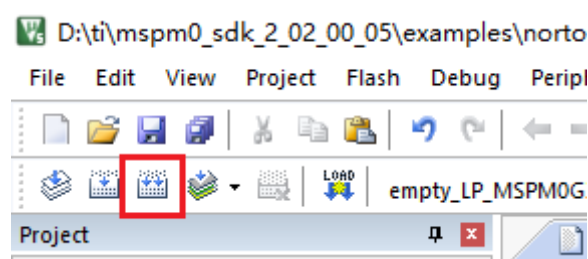
```
#include "ti_msp_dl_config.h"

int main(void)
{
    SYSCFG_DL_init();

    while (1)
    {
        //如果读取到的引脚值大于0，说明PA18引脚为高电平
        // If the pin value read is greater than 0, it means that the
        PA18 pin is high level.
        if( DL_GPIO_readPins(KEY_PORT, KEY_PIN_18_PIN) > 0 )
        {
            DL_GPIO_setPins(LED1_PORT, LED1_PIN_2_PIN); //LED控制输出
            高电平 LED control output high level
        }
        else//如果PA18引脚为低电平 If the PA18 pin is low
        {
            DL_GPIO_clearPins(LED1_PORT, LED1_PIN_2_PIN); //LED控制输出
            低电平 LED control output low level
        }
    }
}
```

### 4. Compile

Click the Rebuild icon. The following prompt appears, indicating that the compilation is complete and there are no errors.



```

Generating Code (empty.syscfg)...
Unchanged D:\ti\mspm0_sdk_2_02_00_05\examples\nortos\LP_MSPM0G3507\driverlib\empty\ti_msp_dl_config.c...
Unchanged D:\ti\mspm0_sdk_2_02_00_05\examples\nortos\LP_MSPM0G3507\driverlib\empty\ti_msp_dl_config.h...
Unchanged D:\ti\mspm0_sdk_2_02_00_05\examples\nortos\LP_MSPM0G3507\driverlib\empty\Event.dot...
assembling startup_mspm0g350x_uvision.s...
compiling empty.c...
compiling ti_msp_dl_config.c...
linking...
Program Size: Code=544 RO-data=208 RW-data=0 ZI-data=352
FromELF: creating hex file...
".\Objects\empty_LP_MSPM0G3507_nortos_keil.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:06

```

## 4. Program Analysis

- dl\_gpio.h

```

/**
 * @brief      Read a group of GPIO pins
 *
 * @param[in]  gpio  Pointer to the register overlay for the peripheral
 * @param[in]  pins  Pins to read. Bitwise OR of @ref DL_GPIO_PIN.
 *
 * @return     The pins (from the selection) that are currently high
 *
 * @retval     Bitwise OR of @ref DL_GPIO_PIN of pins that are currently high
 *             from the input selection.
 */
__STATIC_INLINE uint32_t DL_GPIO_readPins(GPIO_Regs* gpio, uint32_t pins)
{
    return (gpio->DIN31_0 & pins);
}

```

**uint32\_t DL\_GPIO\_readPins(GPIO\_Regs\* gpio, uint32\_t pins)** is used to read the level status of the specified GPIO pin. Get the status of the specified pin by reading the value of the GPIO register. Enter the GPIO port and pin number to be read, and return the level status of the pin.

**gpio:** The GPIO port to be read.

**pins:** The pin number to be read.

- empty.c

```

int main(void)
{
    SYSCFG_DL_init();

    while (1)
    {
        //如果读取到的引脚值大于0，说明PA18引脚为高电平
        // If the pin value read is greater than 0, it means that the
        PA18 pin is high level.
        if( DL_GPIO_readPins(KEY_PORT, KEY_PIN_18_PIN) > 0 )
        {
            DL_GPIO_setPins(LED1_PORT, LED1_PIN_2_PIN); //LED控制输出
            高电平 LED control output high level
        }
        else//如果PA18引脚为低电平 If the PA18 pin is low
        {

```



```
DL_GPIO_clearPins(LED1_PORT,LED1_PIN_2_PIN);  
//LED控制输出  
低电平 LED control output low level  
    }  
}  
}
```

This code continuously detects the level state of pin `PA18` (button access pin) and controls the output level of `LED1_PIN_2` (LED control pin) to realize the function of key control LED. When `PA18` is high level, the LED is turned on (LED pin outputs high level); when `PA18` is low level, the LED is turned off (LED pin outputs low level).

## 5. Experimental phenomenon

After the program is downloaded, the LED light can be controlled by the button. Press the button to turn on the LED light, and release the button to turn off the LED light.

