Light up the LED

1. Learning objectives

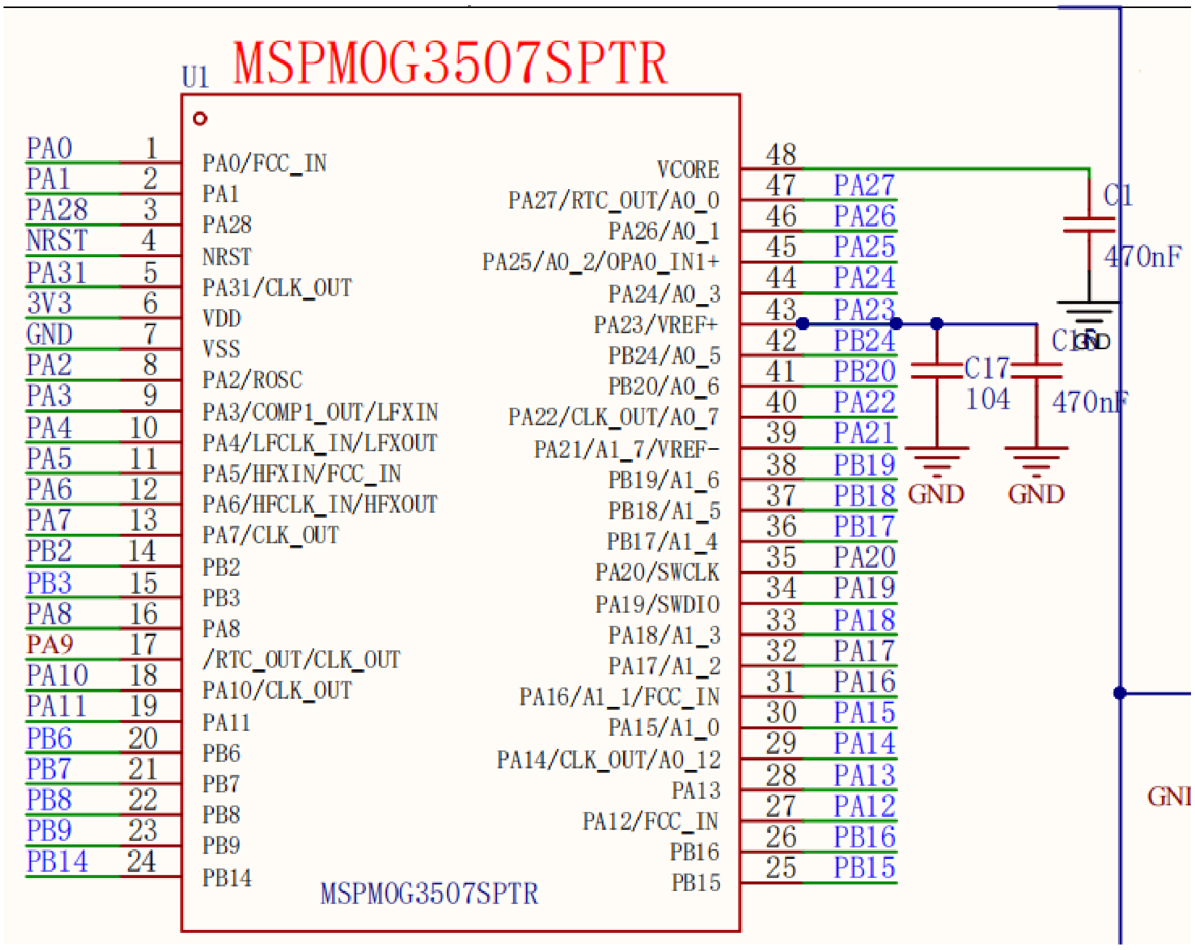
1. Learn the basic use of the pins of the MSPM0G3507 motherboard.
2. Understand how to control the onboard LED lights.

2. Hardware construction

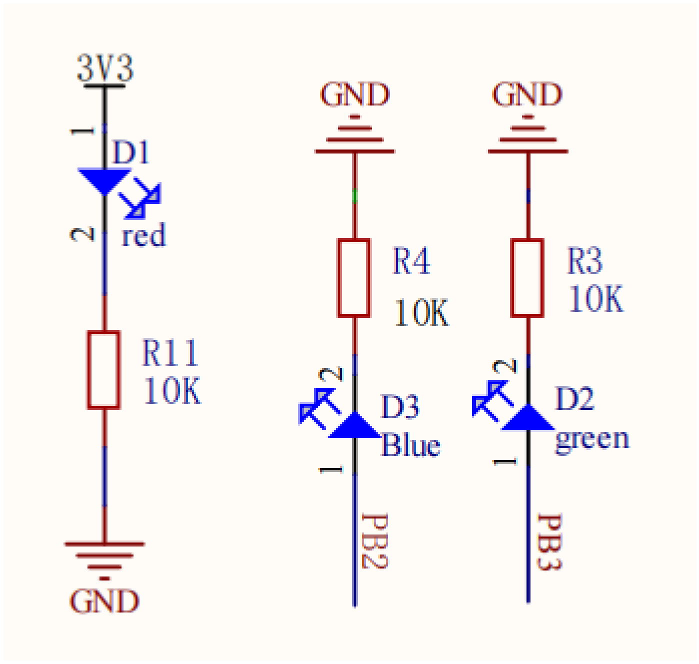
This course does not require additional hardware equipment, and can directly use the onboard LED lights on the MSPM0G3507 motherboard.

We set up two LED user lights on the motherboard, and users can DIY their functions. Take **LED1** as an example below.

MSPM0G3507 main control diagram



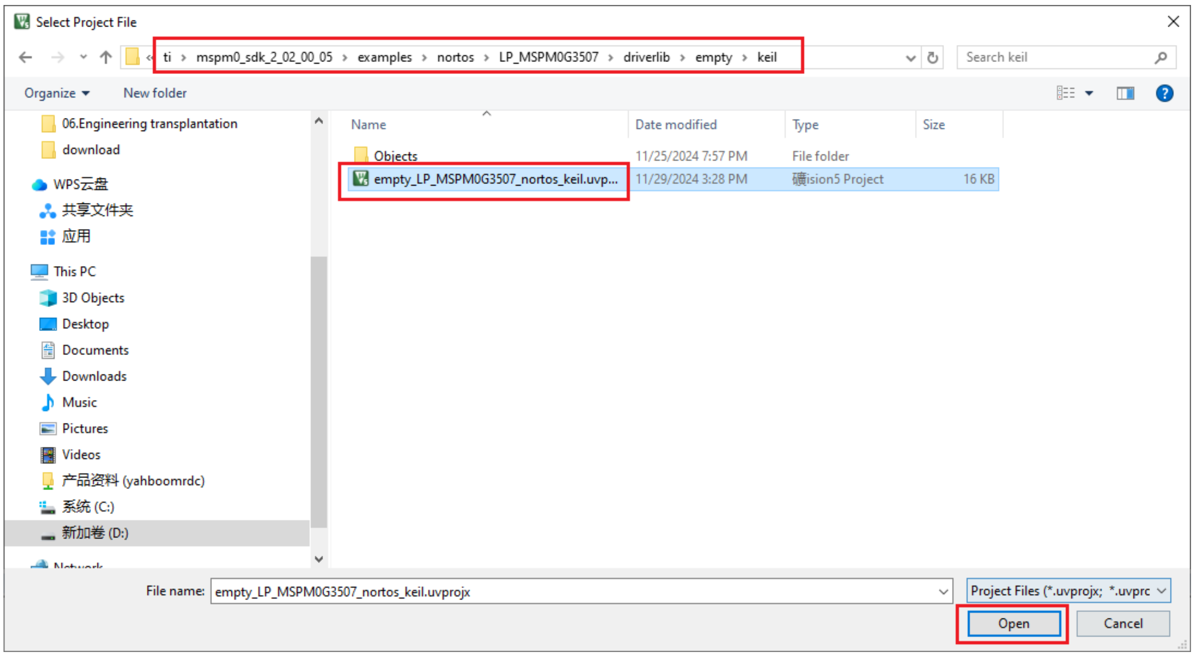
LED part schematic diagram



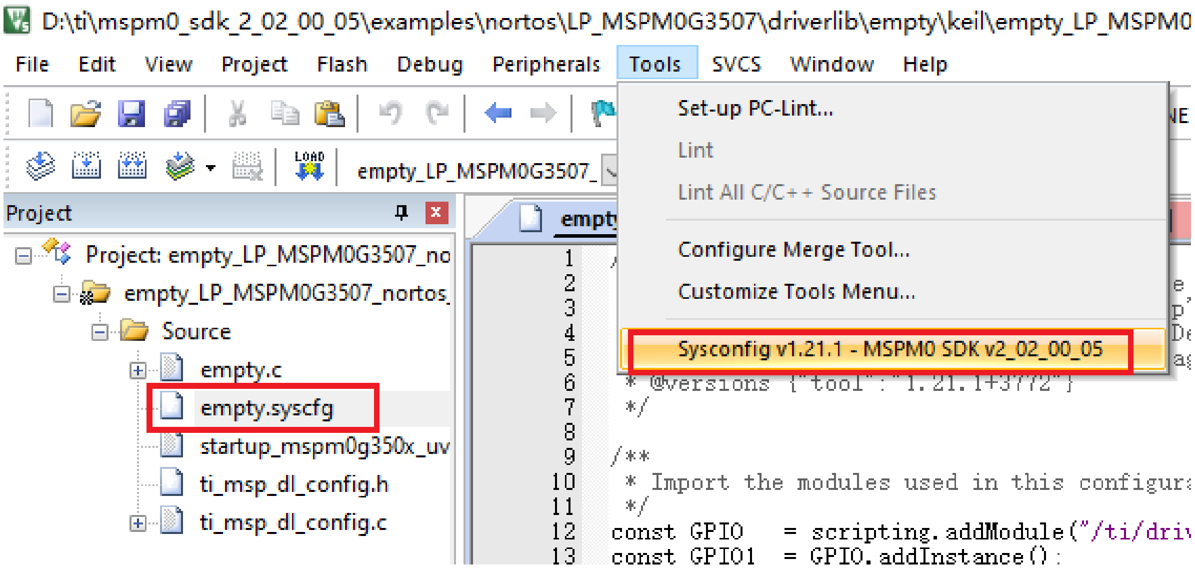
3. Experimental steps

1. Open the SYSCONFIG configuration tool

Open the blank project empty in the SDK in KEIL.

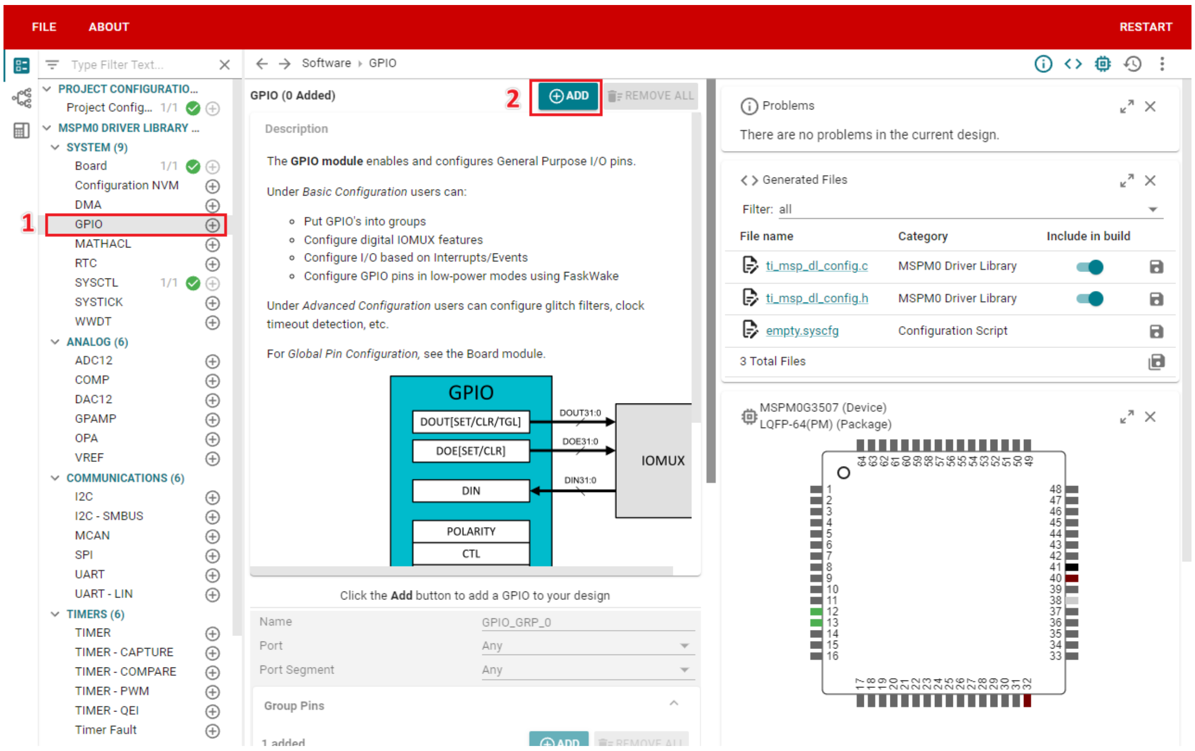


Open after selecting, open the empty.syscfg file in the keil interface, **when the empty.syscfg file is open**, then select to open the SYSCONFIG GUI interface in the Tools bar.



```
 * @versions {"tool":"1.21.1+3772"}
 */

/**
 * Import the modules used in this configur
 */
const GPIO    = scripting.addModule("/ti/driv
const GPIO1   = GPIO.addInstance();
```

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.



PROJECT CONFIGURATIO...
Project Config... 1/1
MSPM0 DRIVER LIBRARY ...
SYSTEM (9)
Board 1/1
Configuration NVM
DMA
GPIO
MATHACL
RTC
SYSCTL 1/1
SYSTICK
WWDT
ANALOG (6)
ADC12
COMP
DAC12
GPAMP
OPA
VREF
COMMUNICATIONS (6)
I2C
I2C - SMBUS
MCAN
SPI
UART
UART - LIN
TIMERS (6)
TIMER
TIMER - CAPTURE
TIMER - COMPARE
TIMER - PWM
TIMER - QEI
Timer Fault

GPIO (0 Added)

Description

The **GPIO module** enables and configures General Purpose I/O pins.

Under *Basic Configuration* users can:

- Put GPIO's into groups
- Configure digital IOMUX features
- Configure I/O based on Interrupts/Events
- Configure GPIO pins in low-power modes using FaskWake

Under *Advanced Configuration* users can configure glitch filters, clock timeout detection, etc.

For *Global Pin Configuration*, see the Board module.

Click the **Add** button to add a GPIO to your design

Name	GPIO_GRP_0
Port	Any
Port Segment	Any

Group Pins

1 added

Problems

There are no problems in the current design.

Generated Files

Filter: all

File name	Category	Include in build
ti_msp_dl_config.c	MSPM0 Driver Library	●
ti_msp_dl_config.h	MSPM0 Driver Library	●
empty.syscfg	Configuration Script	

3 Total Files

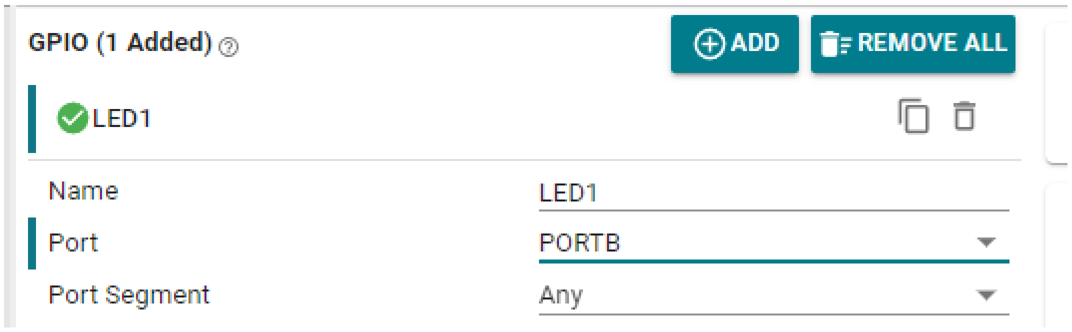
MSPM0G3507 (Device)
LQFP-64(PM) (Package)

2. Parameter settings

2.1 Set port parameters

Set GPIO parameters and name the pin LED1

From the LED schematic, we can see that the port where LED1 is located is PB2, so set the Port to PORTB.



GPIO (1 Added)

LED1

Name	LED1
Port	PORTB
Port Segment	Any

Parameter description:

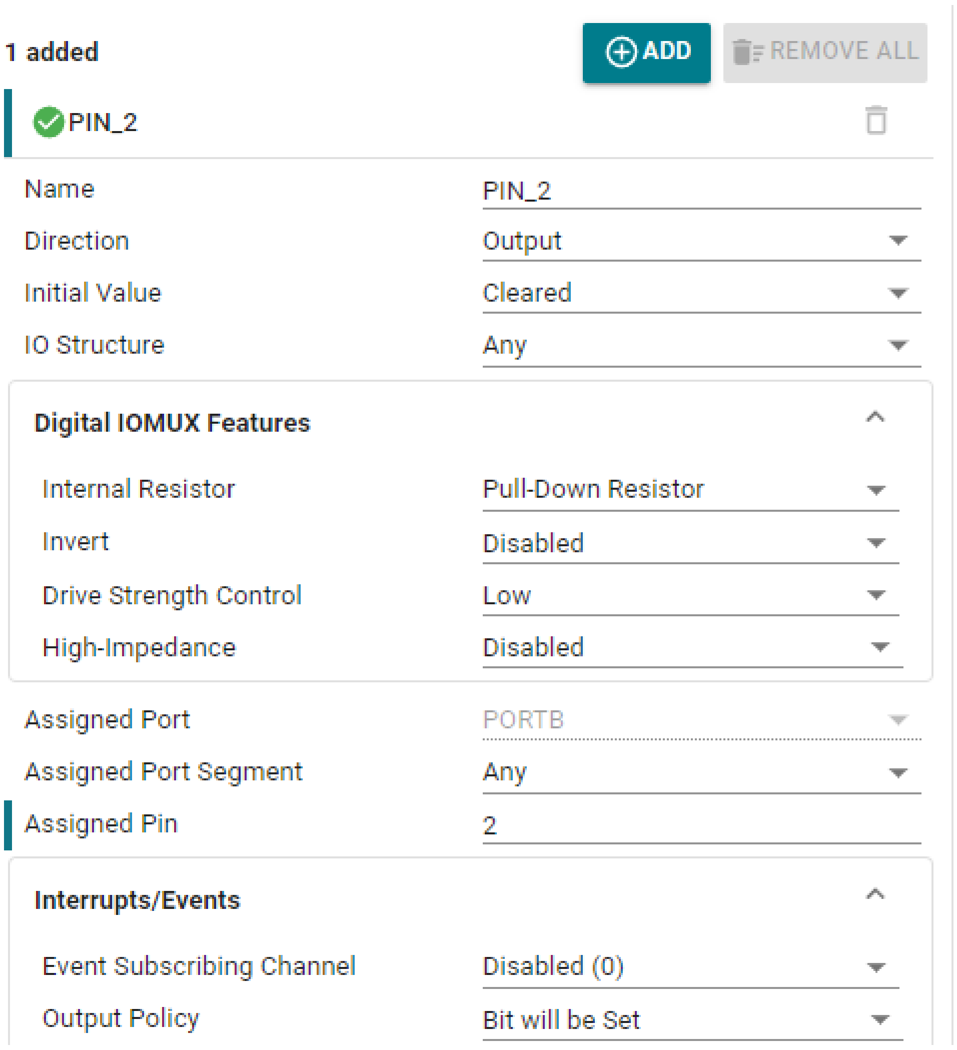
Name: Custom name of the GPIO instance. By default, the name starts with a numeric suffix of "0"; we can customize the name to reflect the purpose of the module (for example, name the GPIO "LED1" so that we know that this pin is specifically used to control LED1).

Port: The port where the GPIO instance is located. The LED is connected to the GPIOB2 pin, so only PORTB can be selected.

Port Segment: Set the pull-up and pull-down resistors on the port. Note that it is the pull-up and pull-down of the port, which sets the entire GPIOB port.

2.2 Set pin parameters

Set the pin to output mode, set the pin to output low level by default, set the pin to pull-down mode, set the pin to GPIOB_2 pin, and do not enable interrupt events.



Name	PIN_2
Direction	Output
Initial Value	Cleared
IO Structure	Any

Digital IOMUX Features

Internal Resistor	Pull-Down Resistor
Invert	Disabled
Drive Strength Control	Low
High-Impedance	Disabled

Assigned Port	PORTB
Assigned Port Segment	Any
Assigned Pin	2

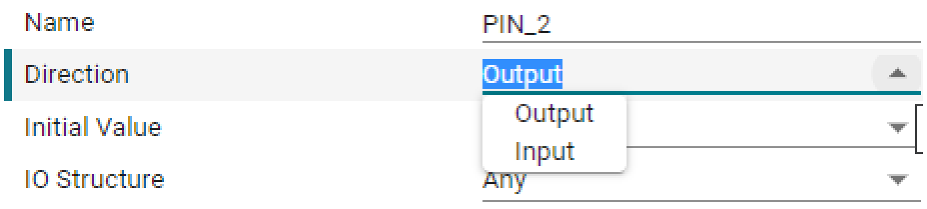
Interrupts/Events

Event Subscribing Channel	Disabled (0)
Output Policy	Bit will be Set

Parameter description:

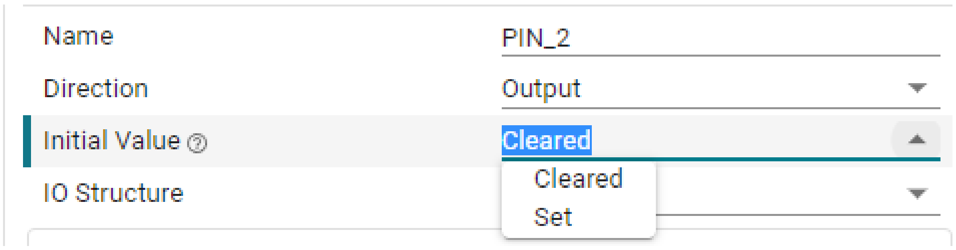
Name: User-defined pin name, set to PIN_2.

Direction: Set the pin mode, input and output. Here we control the LED light and select the output mode.



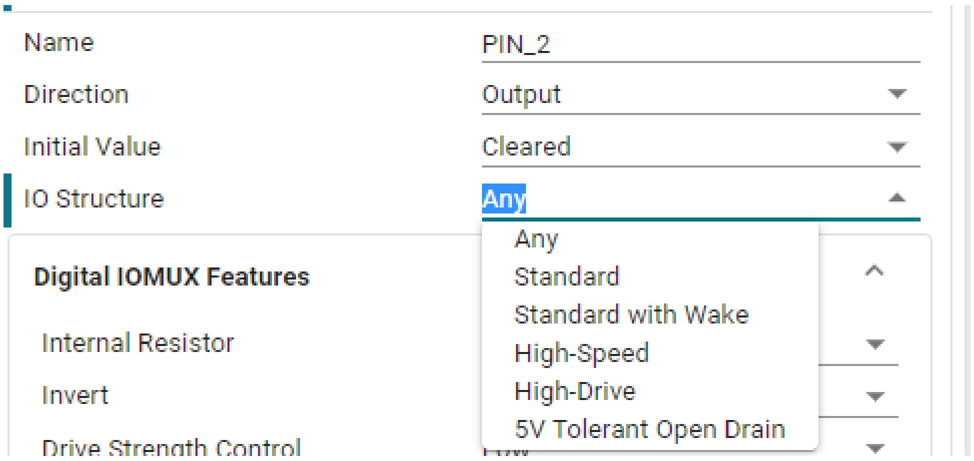
Name	PIN_2
Direction	Output
Initial Value	
IO Structure	Any

Initial Value: Set the initial state of the pin. It can only be set when configured as output mode. There are two options, clear and set. Clear means output low level, set means output high level.



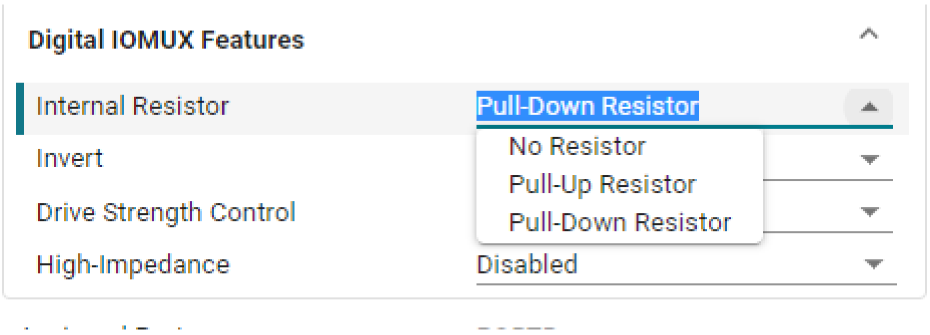
Name	PIN_2
Direction	Output
Initial Value	Cleared
IO Structure	

IO Structure: Set the IO structure. There are multiple options, default (Any), standard (Standard), wake up (Standard with Wake), high speed (High-Speed) and can tolerate 5V (5V Tolerant Open Drain) structure.



Name PIN_2
Direction Output
Initial Value Cleared
IO Structure Any

Internal Resistor: Set the pull-up and pull-down resistors of the pin. There are three options: no setting, pull-up setting, and pull-down setting. Here, the pull-down resistor is selected according to the connection method of the LED light.

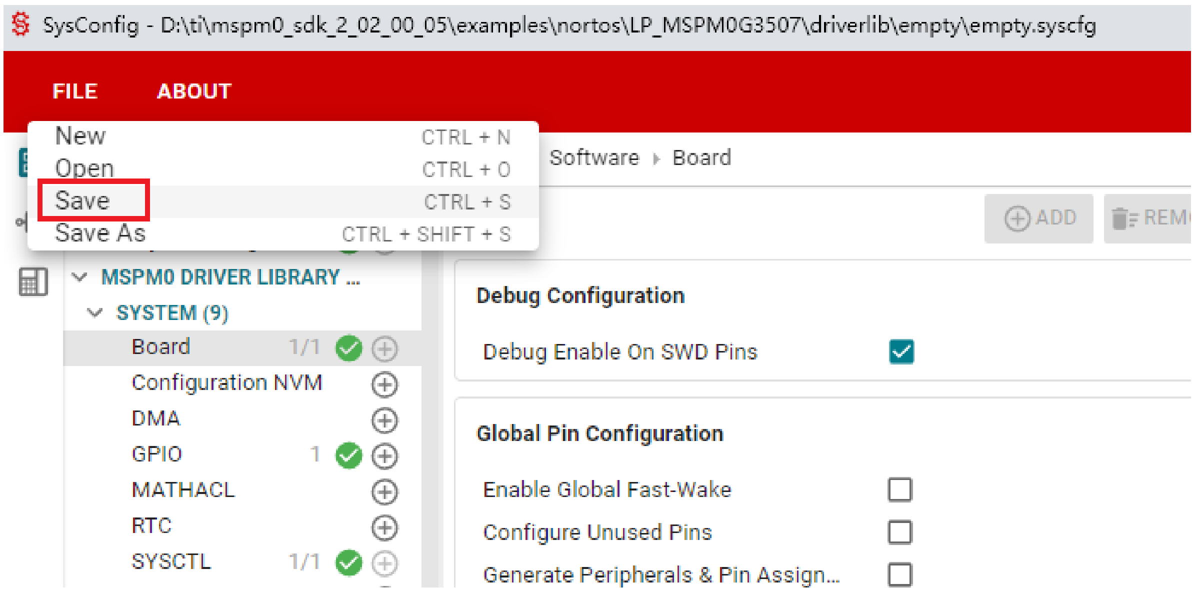


Digital IOMUX Features
Internal Resistor Pull-Down Resistor
High-Impedance Disabled

Assigned Pin: Set the pin number. Fill in the corresponding pin number for the pin to be controlled. For example, if the LED in the development board is connected to GPIOB2, fill in 2.

2.3 Save and update the configuration file

Click SAVE to save the configuration in SYSCONFIG, then turn off SYSCONFIG and return to keil.

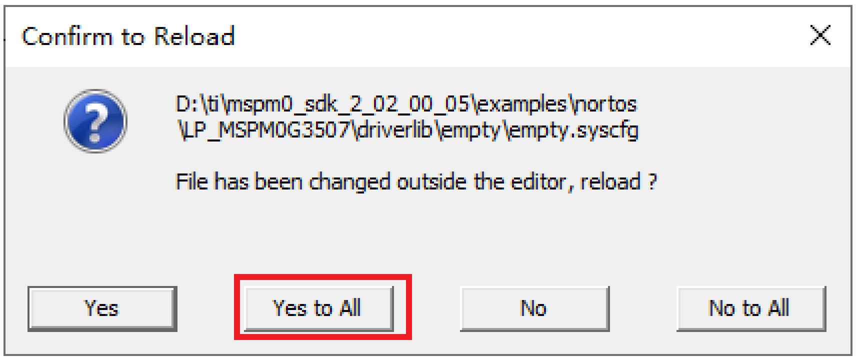


SysConfig - D:\ti\mspm0_sdk_2_02_00_05\examples\nortos\LP_MSPM0G3507\driverlib\empty\empty.syscfg

Debug Configuration
Debug Enable On SWD Pins ☑

Global Pin Configuration
Enable Global Fast-Wake ☐
Configure Unused Pins ☐
Generate Peripherals & Pin Assign... ☐

Select `Yes to All` in the pop-up window

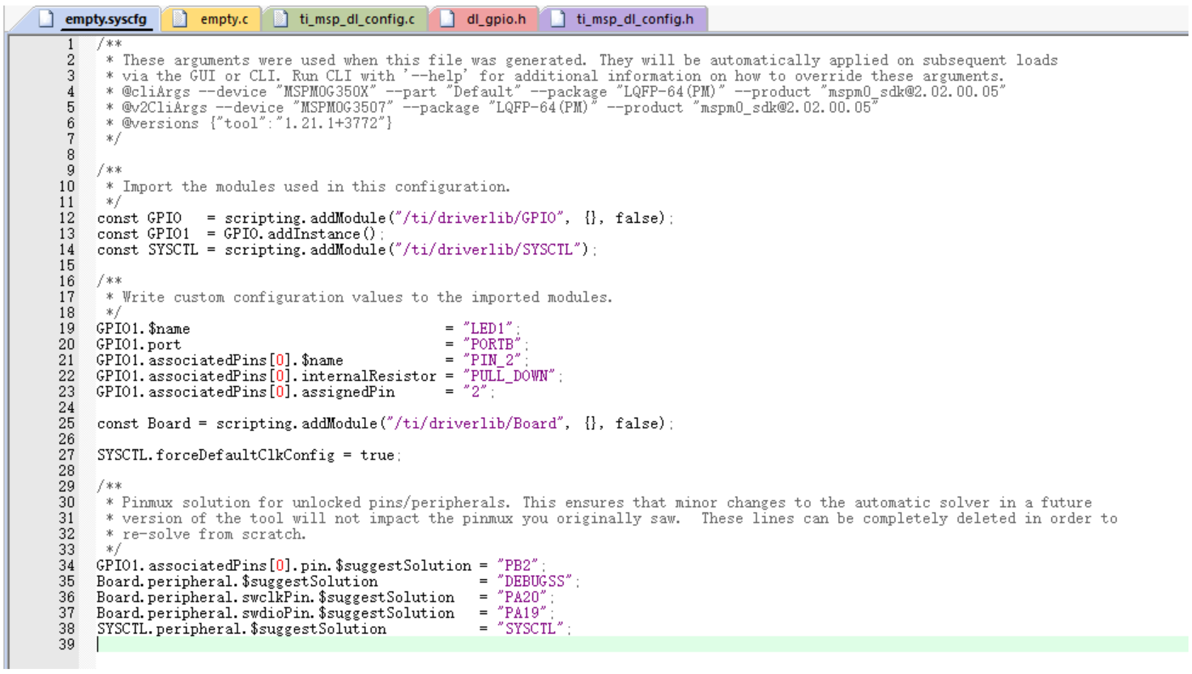


Confirm to Reload

D:\ti\mspm0_sdk_2_02_00_05\examples\nortos
\LP_MSPM0G3507\driverlib\empty\empty.syscfg

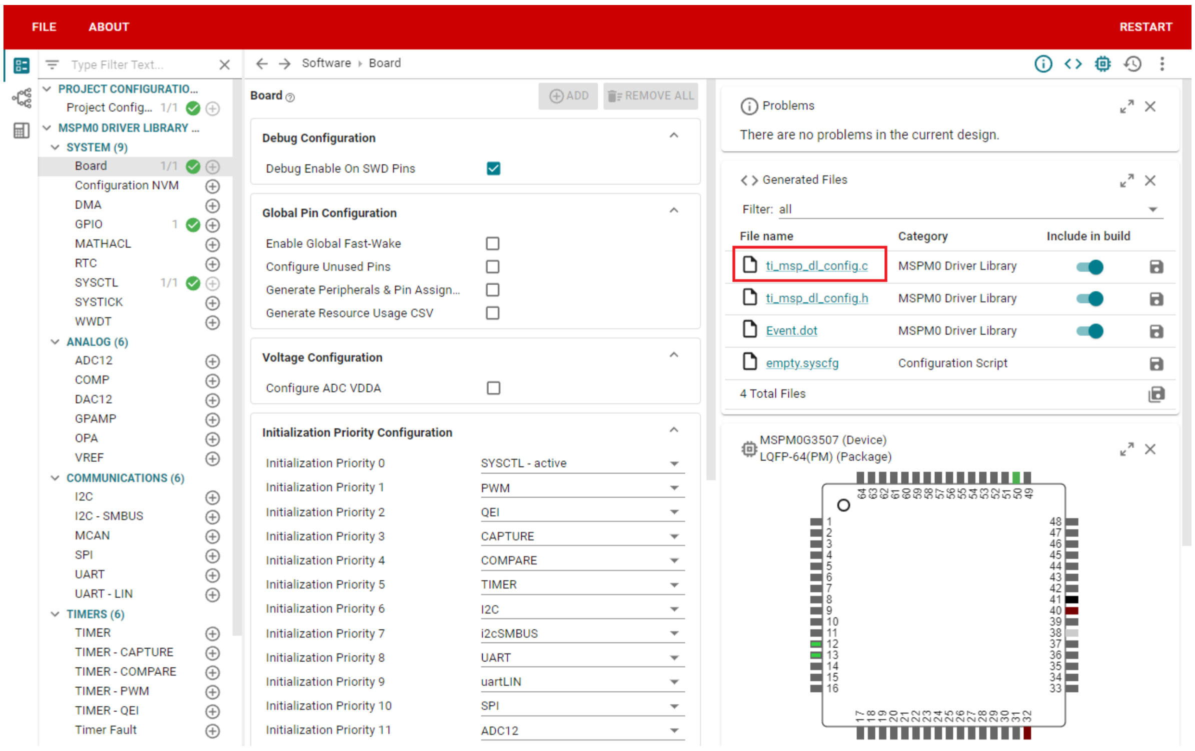
File has been changed outside the editor, reload ?

After that, we can see that the empty.syscfg file is updated with the parameters we set.



```
/**
 * These arguments were used when this file was generated. They will be automatically applied on subsequent loads
 * via the GUI or CLI. Run CLI with '--help' for additional information on how to override these arguments.
 * @cliArgs --device "MSPM0G350X" --part "Default" --package "LQFP-64(PM)" --product "mspm0_sdk@2.02.00.05"
 * @v2CliArgs --device "MSPM0G3507" --package "LQFP-64(PM)" --product "mspm0_sdk@2.02.00.05"
 * @versions {"tool":"1.21.1+3772"}
 */

/**
 * Import the modules used in this configuration.
 */
const GPIO   = scripting.addModule("/ti/driverlib/GPIO", {}, false);
const GPIO1  = GPIO.addInstance();
const SYSCTL = scripting.addModule("/ti/driverlib/SYSCTL");

/**
 * Write custom configuration values to the imported modules.
 */
GPIO1.$name                              = "LED1";
GPIO1.port                               = "PORTB";
GPIO1.associatedPins[0].$name            = "PIN_2";
GPIO1.associatedPins[0].internalResistor = "PULL_DOWN";
GPIO1.associatedPins[0].assignedPin      = "2";

const Board = scripting.addModule("/ti/driverlib/Board", {}, false);

SYSCTL.forceDefaultClkConfig = true;

/**
 * Pinmux solution for unlocked pins/peripherals. This ensures that minor changes to the automatic solver in a future
 * version of the tool will not impact the pinmux you originally saw.  These lines can be completely deleted in order to
 * re-solve from scratch.
 */
GPIO1.associatedPins[0].pin.$suggestSolution = "PB2";
Board.peripheral.$suggestSolution            = "DEBUGSS";
Board.peripheral.swclkPin.$suggestSolution   = "PA20";
Board.peripheral.swdioPin.$suggestSolution   = "PA19";
SYSCTL.peripheral.$suggestSolution           = "SYSCTL";
```

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



Debug Configuration

Debug Enable On SWD Pins ☑

Global Pin Configuration

Enable Global Fast-Wake ☐
Configure Unused Pins ☐
Generate Peripherals & Pin Assign... ☐
Generate Resource Usage CSV ☐

Voltage Configuration

Configure ADC VDDA ☐

Initialization Priority Configuration

Initialization Priority 0	SYSCTL - active
Initialization Priority 1	PWM
Initialization Priority 2	QEI
Initialization Priority 3	CAPTURE
Initialization Priority 4	COMPARE
Initialization Priority 5	TIMER
Initialization Priority 6	I2C
Initialization Priority 7	i2cSMBUS
Initialization Priority 8	UART
Initialization Priority 9	uartLIN
Initialization Priority 10	SPI
Initialization Priority 11	ADC12

Problems

There are no problems in the current design.

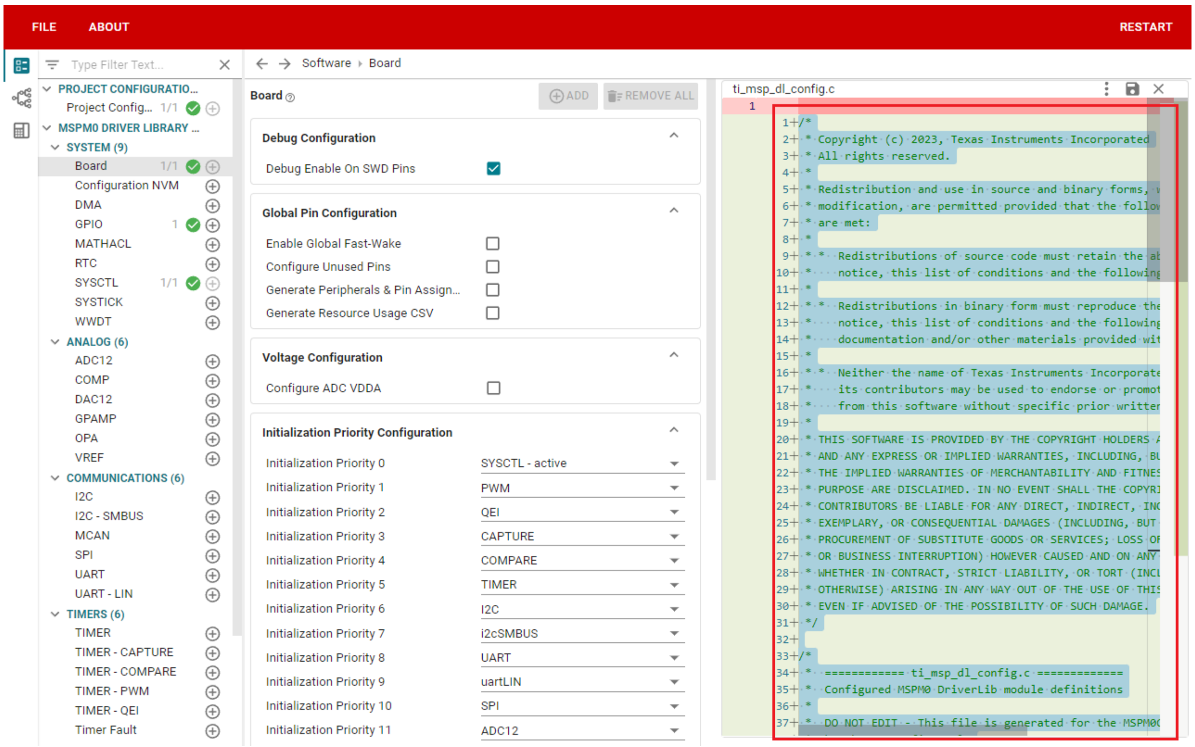
Generated Files

Filter: all

File name	Category	Include in build
ti_msp_dl_config.c	MSPM0 Driver Library	●
ti_msp_dl_config.h	MSPM0 Driver Library	●
Event.dot	MSPM0 Driver Library	●
empty.syscfg	Configuration Script	

4 Total Files

MSPM0G3507 (Device)
LQFP-64(PM) (Package)



Debug Configuration

Debug Enable On SWD Pins	✓

Global Pin Configuration

Enable Global Fast-Wake	☐
Configure Unused Pins	☐
Generate Peripherals & Pin Assign...	☐
Generate Resource Usage CSV	☐

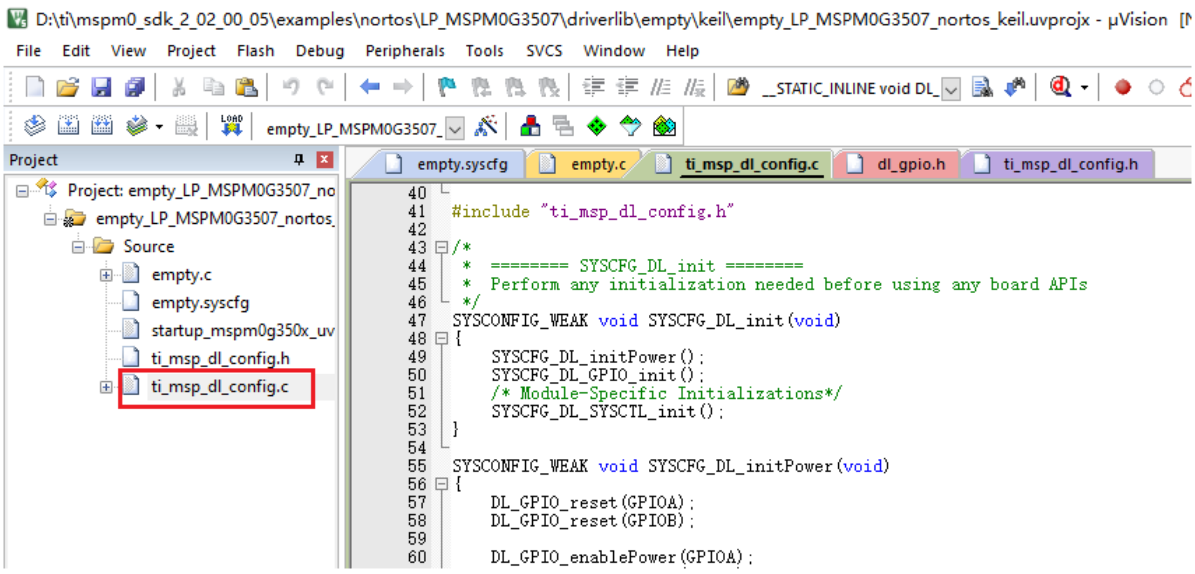
Voltage Configuration

Configure ADC VDDA	☐

Initialization Priority Configuration

Initialization Priority 0	SYSCTL - active
Initialization Priority 1	PWM
Initialization Priority 2	QEI
Initialization Priority 3	CAPTURE
Initialization Priority 4	COMPARE
Initialization Priority 5	TIMER
Initialization Priority 6	I2C
Initialization Priority 7	i2cSMBUS
Initialization Priority 8	UART
Initialization Priority 9	uartLIN
Initialization Priority 10	SPI
Initialization Priority 11	ADC12

```
/*
 * Copyright (c) 2023, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, w
 * modification, are permitted provided that the follow
 * are met:
 *
 * *  Redistributions of source code must retain the ab
 *    notice, this list of conditions and the following
 *
 * *  Redistributions in binary form must reproduce the
 *    notice, this list of conditions and the following
 *    documentation and/or other materials provided wit
 *
 * *  Neither the name of Texas Instruments Incorporate
 *    its contributors may be used to endorse or promot
 *    from this software without specific prior writter
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS A
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BU
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNES
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRI
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INC
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCL
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/*
 *  ============ ti_msp_dl_config.c =============
 *  Configured MSPM0 DriverLib module definitions
 *
 *  DO NOT EDIT - This file is generated for the MSPM0
```


D:\ti\mspm0_sdk_2_02_00_05\examples\nortos\LP_MSPM0G3507\driverlib\empty\keil\empty_LP_MSPM0G3507_nortos_keil.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
__STATIC_INLINE void DL_
empty_LP_MSPM0G3507_
Project
Project: empty_LP_MSPM0G3507_no
empty_LP_MSPM0G3507_nortos_
Source
empty.c
empty.syscfg
startup_mspm0g350x_uv
ti_msp_dl_config.h
ti_msp_dl_config.c
empty.syscfg
empty.c
ti_msp_dl_config.c
dl_gpio.h
ti_msp_dl_config.h

```
#include "ti_msp_dl_config.h"

/*
 *  ======== SYSCFG_DL_init ========
 *  Perform any initialization needed before using any board APIs
 */
SYSCONFIG_WEAK void SYSCFG_DL_init(void)
{
    SYSCFG_DL_initPower();
    SYSCFG_DL_GPIO_init();
    /* Module-Specific Initializations*/
    SYSCFG_DL_SYSCTL_init();
}

SYSCONFIG_WEAK void SYSCFG_DL_initPower(void)
{
    DL_GPIO_reset(GPIOA);
    DL_GPIO_reset(GPIOB);

    DL_GPIO_enablePower(GPIOA);
```

.h file is the same as above, you can copy the file content in SYSCONFIG.

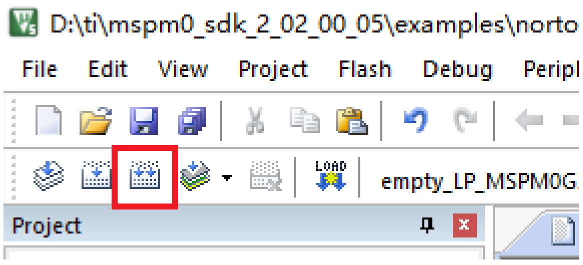
3. Write the program

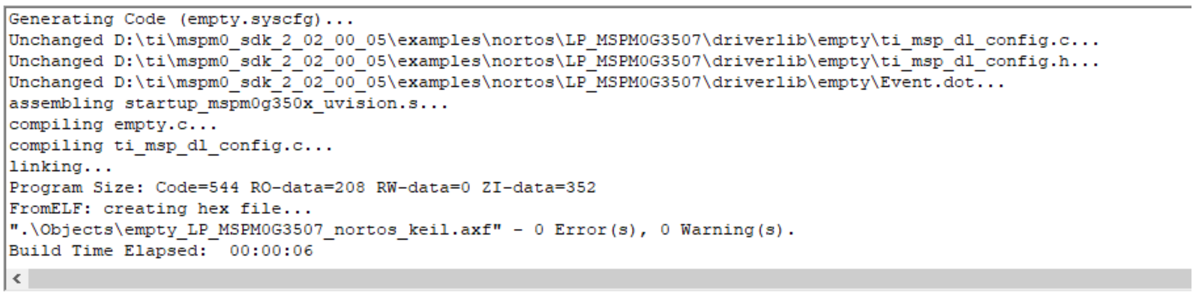
In the empty.c file, write the following code

```
#include "ti_msp_dl_config.h"

//自定义延时（不精确） Custom delay (not precise)
void delay_ms(unsigned int ms)
{
    unsigned int i, j;
    // 下面的嵌套循环的次数是根据主控频率和编译器生成的指令周期大致计算出来的，
    // 需要通过实际测试调整来达到所需的延时。
    // The number of nested loops below is roughly calculated based on the master
control frequency and the instruction cycle generated by the compiler,and needs
to be adjusted through actual testing to achieve the required delay.
    for (i = 0; i < ms; i++)
    {
        for (j = 0; j < 8000; j++)
        {
            // 仅执行一个足够简单以致于可以预测其执行时间的操作
            // Perform only one operation that is simple enough to predict its
execution time
            __asm__("nop"); // "nop" 代表“无操作”，在大多数架构中，这会消耗一个或几个时钟
周期 "nop" stands for "no operation", which on most architectures consumes one or
a few clock cycles
        }
    }
}

int main(void)
{
    SYSCFG_DL_init();
    while (1)
    {
        DL_GPIO_clearPins(LED1_PORT,LED1_PIN_2_PIN);//输出低电平 Output low level
        delay_ms(1000);//延时大概1S Delay about 1S
        DL_GPIO_setPins(LED1_PORT,LED1_PIN_2_PIN);  //输出高电平 Output high level
        delay_ms(1000);//延时大概1S Delay about 1S
    }
}
```

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      Set a group of GPIO pins
 *
 *  @param[in]  gpio  Pointer to the register overlay for the peripheral
 *  @param[in]  pins  Pins to set high. Bitwise OR of @ref DL_GPIO_PIN.
 */
__STATIC_INLINE void DL_GPIO_setPins(GPIO_Regs* gpio, uint32_t pins)
{
```

```c
    gpio->DOUTSET31_0 = pins;
}

/**
 *  @brief      Clear a group of GPIO pins
 *
 *  @param[in]  gpio  Pointer to the register overlay for the peripheral
 *  @param[in]  pins  Pins to clear. Bitwise OR of @ref DL_GPIO_PIN.
 */
__STATIC_INLINE void DL_GPIO_clearPins(GPIO_Regs* gpio, uint32_t pins)
{
    gpio->DOUTCLR31_0 = pins;
}

/**
 *  @brief      Toggle a group of GPIO pins
 *
 *  @param[in]  gpio  Pointer to the register overlay for the peripheral
 *  @param[in]  pins  Pins to toggle. Bitwise OR of @ref DL_GPIO_PIN.
 */
__STATIC_INLINE void DL_GPIO_togglePins(GPIO_Regs* gpio, uint32_t pins)
{
    gpio->DOUTTGL31_0 = pins;
}
```

__STATIC_INLINE void DL_GPIO_setPins(GPIO_Regs* gpio, uint32_t pins): This function outputs a high level for the control pin.

__STATIC_INLINE void DL_GPIO_clearPins(GPIO_Regs* gpio, uint32_t pins): This function outputs a low level for the control pin.

__STATIC_INLINE void DL_GPIO_togglePins(GPIO_Regs* gpio, uint32_t pins): This function flips the level of the control pin. If it was originally a high level, it will become a low level. If it was originally a low level, it will become a high level.

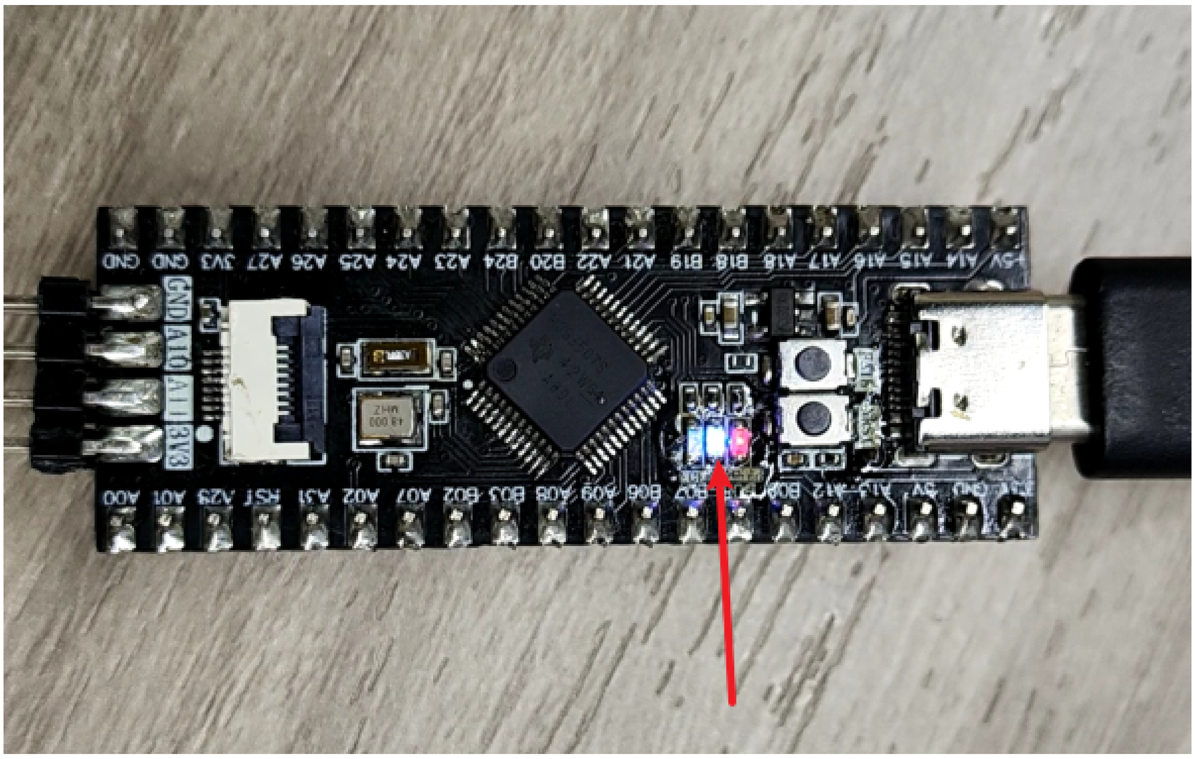
- empty.c

```c
int main(void)
{
    SYSCFG_DL_init();
    while (1)
    {
        DL_GPIO_clearPins(LED1_PORT,LED1_PIN_2_PIN);//输出低电平 Output low level
        delay_ms(1000);//延时大概1S Delay about 1S
        DL_GPIO_setPins(LED1_PORT,LED1_PIN_2_PIN);  //输出高电平 Output high level
        delay_ms(1000);//延时大概1S Delay about 1S
    }
}
```

Set PB2 to output low level, delay about 1s, then output high level, and delay about 1s again. Repeat this process.

5. Experimental phenomenon

After the program is downloaded, you can see that the LED1 light on the MSPM0 development board is lit and flashes every about 1s.


GND
A10
A11
3V3