

Buzzer Control

Buzzer Control

- I. Learning Objectives
- II. Hardware Setup
- III. Experiment Steps
 1. Open SYS CONFIG Configuration Tool
 2. Parameter Settings
 - 2.1 Set Port Parameters
 - 2.3 Save and Update Configuration File
 3. Write Program
 4. Compile
- IV. Program Analysis
- V. Experiment Phenomenon

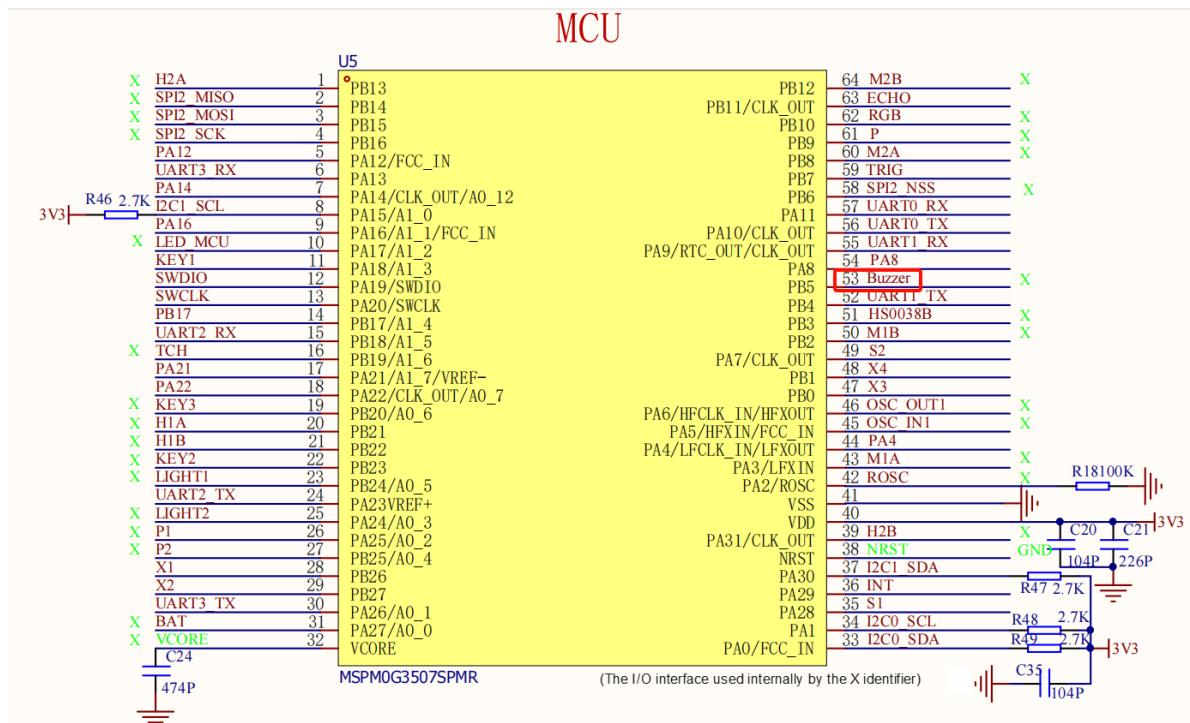
I. Learning Objectives

1. Learn the basic usage of MSPM0G3507 development board pins.
2. Understand how to control the onboard active buzzer.

II. Hardware Setup

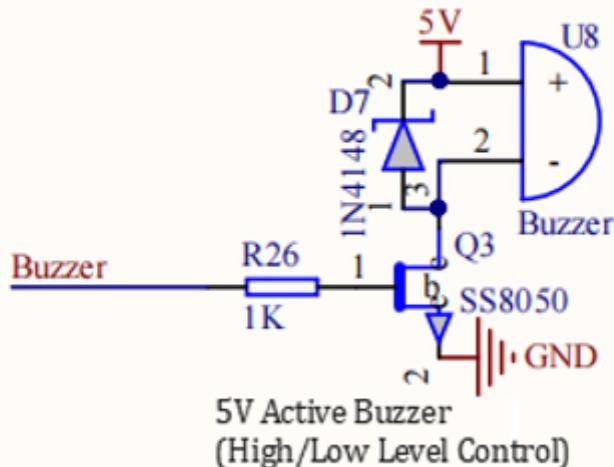
This course requires no additional hardware devices, directly using the onboard buzzer on the MSPM0G3507 development board.

MSPM0G3507 Main Control Diagram



Buzzer Part Circuit Diagram

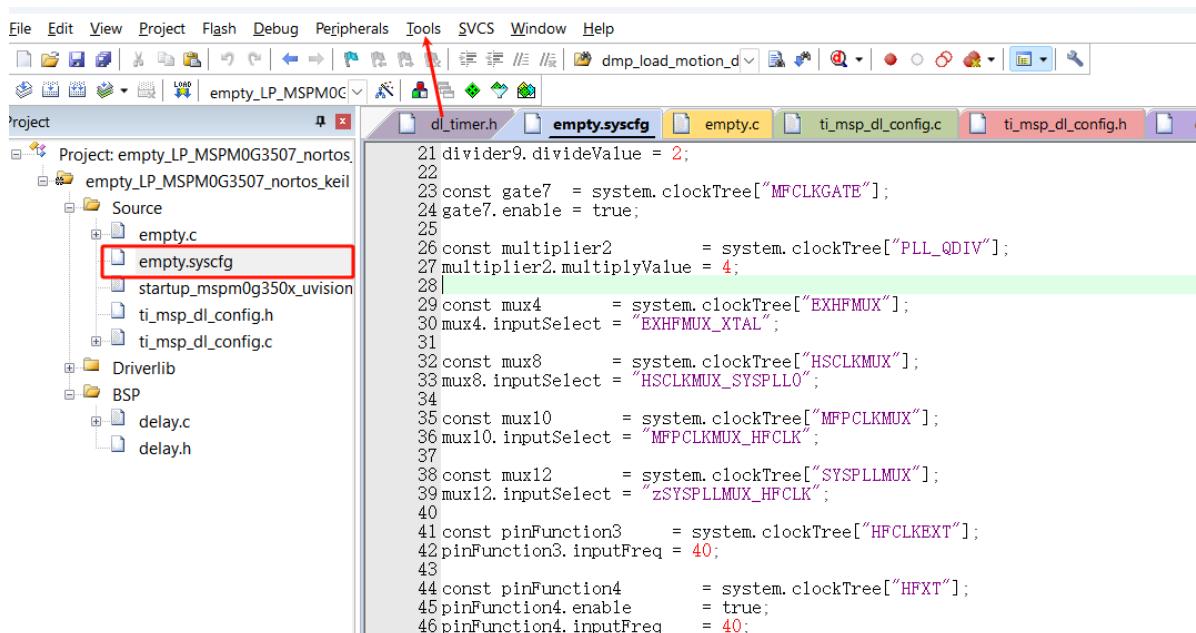
Buzzer



III. Experiment Steps

1. Open SYSCONFIG Configuration Tool

In KEIL, open the empty project from the SDK. After selecting and opening, in the KEIL interface, open the empty.syscfg file. **With the empty.syscfg file open**, select Open SYSCONFIG GUI interface from the Tools menu.



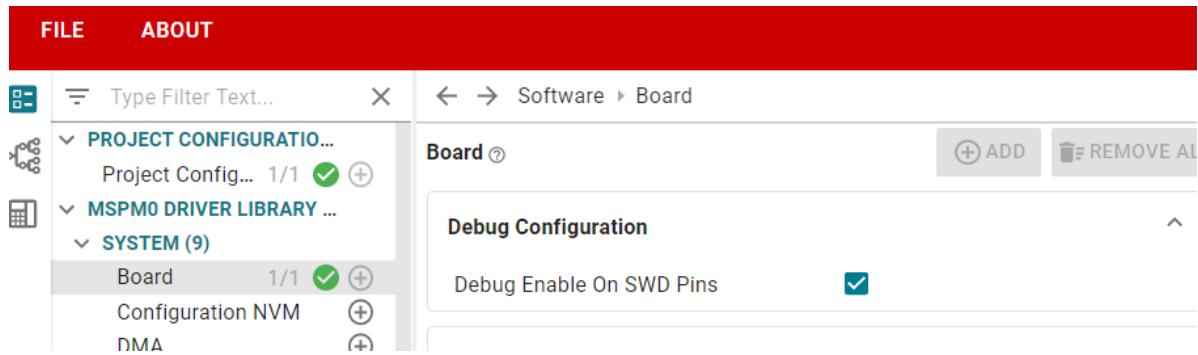
The screenshot shows the TI Code Composer Studio interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The Tools menu is open, displaying options like Set-up PC-Lint..., Lint, Lint All C/C++ Source Files, Configure Merge Tool..., Customize Tools Menu..., and Sysconfig v1.21.1 - MSPM0 SDK v2_02_00_05. The main workspace shows a project named 'empty_LP_MSPM0G3507_nortos' with a file tree containing Source, Driverlib, and BSP folders. A code editor window on the right displays a portion of the 'ti_msp_dl_config.c' file.

```

25
26 const multiplier2      = system.clockTree["PLL_QDIV"];
27 multiplier2.multiplyValue = 4;
28
29 const mux4      = system.clockTree["EXHFMUX"];
30 mux4.inputSelect = "EXHFMUX_XTAL";
31
32 const mux8      = system.clockTree["HSCLKMUX"];
33 mux8.inputSelect = "HSCLKMUX_SYSPLL0";
34
35 const mux10     = system.clockTree["MFPCCLKMUX"];
36 mux10.inputSelect = "MFPCCLKMUX_HFCLK";
37
38 const mux12     = system.clockTree["SYSPLLMUX"];
39 mux12.inputSelect = "zSYSPLLMUX_HFCLK";
40
41 const pinFunction3 = system.clockTree["HFCLKEXT"];
42 pinFunction3.inputFreq = 40;
43
44 const pinFunction4 = system.clockTree["HFXT"];

```

We first enable SWD debugging so that we can use debugging tools (DAP-Link, Jlink) to burn and debug the project.



We need to enable the SYSTICK clock for delay, set it to 1us here. Whether to enable SYSTICK interrupt depends on needs, it is disabled by default here (the period can be adjusted according to the clock tree configuration).

The screenshot shows the detailed configuration for the SYSTICK component. The left sidebar shows the SYSTICK component selected under SYSTEM. The right panel displays the 'Configuration' and 'Other Dependencies' sections for SYSTICK. In the Configuration section, 'Initialize Period' is checked, 'Period (In MCLK Cycles)' is set to 80, and 'Calculated Period' is 1.00 μs. The 'Enable SysTick Interrupt' checkbox is unchecked. The 'Enable SysTick And Start Counting...' checkbox is checked. The 'Other Dependencies' section shows 'Board Generic Board Configuration' and 'SYSCTL System Control Module. Controls Power and Clocking'. The 'Graphical Clock Configuration' section has 'Use Clock Tree' checked.

2. Parameter Settings

2.1 Set Port Parameters

In SYSCONFIG, select MCU peripherals on the left, find GPIO option and click to enter. In GPIO, click ADD to add a GPIO group, the name can be customized, here it is BEEP.

Set GPIO parameters, name this pin as OUT

From the buzzer part circuit diagram, we can see that the buzzer is located on port PB5, so set Port to PORTB.

The screenshot shows the GPIO configuration interface with the following details:

- GPIO (2 Added) :** Shows two groups: LED and BEEP.
- BEEP Group Details:**
 - Name: BEEP
 - Port: PORTB
 - Port Segment: Lower
- Group Pins:**
 - 1 added:** Shows one pin: OUT.
 - OUT Pin Details:**
 - Name: OUT
 - Direction: Output
 - Initial Value: Cleared
 - IO Structure: Any
- Digital IOMUX Features:**
 - Assigned Port: PORTB
 - Assigned Port Segment: Lower
 - Assigned Pin: 5
- Interrupts/Events:** (This section is currently collapsed).

Parameter Description:

Name: Custom name for the GPIO instance. By default, the name starts with numeric suffix "0"; we can customize the name to reflect the module purpose (for example, naming GPIO as "MCU", so we know this pin is specifically for controlling LED_MCU).

Initial Value: Configure the initial level of GPIO, can only be set when configured as output mode. Here it is set to Cleared, default low level. High level is (Set)

Direction: Configure GPIO input/output, here it is output Output, Input is optional

Port: The port where the GPIO instance is located. Buzzer is connected to GPIOB pin, can only select PORTB.

Port Segment: Set port pull-up/pull-down resistor. Note that this is port pull-up/pull-down, setting the entire GPIOB port.

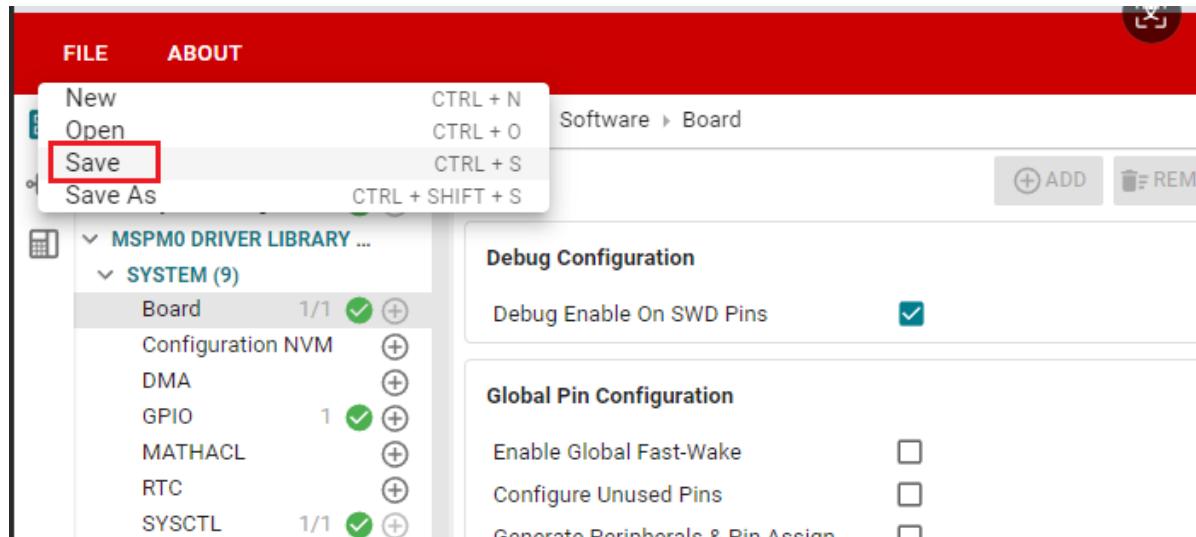
IO Structure: Set IO structure. There are multiple options, default (Any), standard (Standard), wake-up (Standard with Wake), high-speed (High-Speed) and 5V tolerant (5V Tolerant Open Drain) structures.

Internal Resistor: Set pin pull-up/pull-down resistor. There are three options, no setting, set pull-up and set pull-down resistor. Here according to the buzzer connection, pull-down resistor is selected.

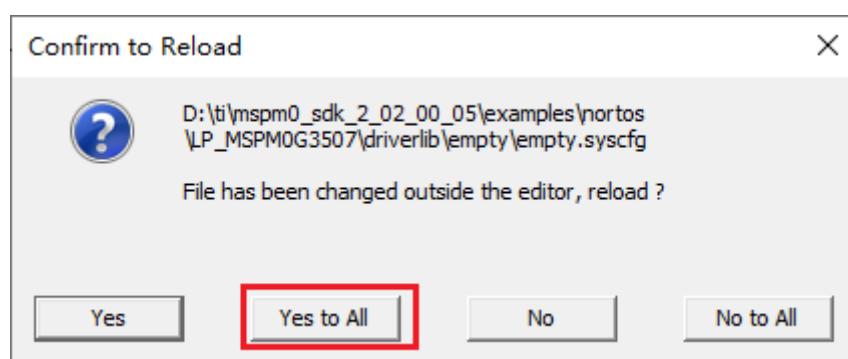
Assigned Pin: Set pin number. To control which pin, fill in the corresponding pin number, for example, the buzzer on the development board is connected to GPIOB5, then fill in 5.

2.3 Save and Update Configuration File

Click SAVE to save the configuration in SYS CONFIG, then close SYS CONFIG and return to keil.



Select Yes to All in the pop-up window



After that, we can see that the empty.syscfg file has been updated with our set parameters.

We also need to confirm whether ti_msp_dl_config.c and ti_msp_dl_config.h files are updated. Directly compile, compilation will automatically update to keil. If not updated, we manually select the project path to save

The screenshot shows the 'Generated Files' panel in the DesignIDE interface. The panel title is 'Generated Files' and it displays a list of files with their categories and build inclusion status. A red arrow points to the 'empty.syscfg' file in the list.

File name	Category	Include in build
ti_msp_dl_config.c	MSPM0 Driver Library	<input checked="" type="checkbox"/>
ti_msp_dl_config.h	MSPM0 Driver Library	<input checked="" type="checkbox"/>
Event.dot	MSPM0 Driver Library	<input checked="" type="checkbox"/>
empty.syscfg	Configuration Script	<input type="checkbox"/>

4 Total Files

Relative path as follows

The screenshot shows a file explorer window displaying the relative file structure. The 'empty.syscfg' file is highlighted with a red box, and the 'ti_msp_dl_config.c' and 'ti_msp_dl_config.h' files are also highlighted with a red box at the bottom of the list.

名称	修改日期	类型	大小
BSP	2025/9/27 10:47	文件夹	
keil	2025/10/10 12:11	文件夹	
ti	2025/9/27 10:47	文件夹	
C empty.c	2025/8/29 16:41	C 源文件	1 KB
empty.syscfg	2025/8/29 16:46	SYSCFG 文件	3 KB
Event.dot	2025/8/20 18:46	Microsoft Word ...	2 KB
README.html	2024/8/30 11:04	Microsoft Edge ...	69 KB
README.md	2024/8/30 11:04	Markdown File	2 KB
C ti_msp_dl_config.c	2025/8/29 16:46	C 源文件	5 KB
C ti_msp_dl_config.h	2025/8/29 16:41	C Header 源文件	4 KB

3. Write Program

In the empty.c file, write the following code

```
#include "ti_msp_dl_config.h"
#include "beep.h"
#include "delay.h"

int main(void)
{
    SYSCFG_DL_init();

    while (1)
    {
```

```

    //Buzzer starts for 200ms then stops for 200ms
    Beep_ON();
    delay_ms(200);
    Beep_OFF();
    delay_ms(200);
}

}

```

beep.c

```

void Beep_ON(void)
{
    DL_GPIO_set Pins(BEEP_PORT, BEEP_OUT_PIN);

}

void Beep_OFF(void)
{
    DL_GPIO_clear Pins(BEEP_PORT, BEEP_OUT_PIN);

}

```

delay.c

```

#include "ti_msp_dl_config.h"
//Using method 0, 1 to implement delay requires disabling systick interrupt
//The following implements delay using three methods: 0: using a loop of no
operations to implement delay 1: using polling to read systick count to implement
delay 2: using systick interrupt to implement delay (default) If using the first
two methods, disable systick interrupt

#define DELAY_SELECT 1
volatile unsigned int delay_times = 0;

//Custom delay (not precise)
#if DELAY_SELECT==0
void delay_ms(unsigned int ms)
{
    unsigned int i, j;
    // The number of nested loops below is roughly calculated based on the main
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++)
        {
            // Only execute a simple operation that can predict its execution
time
            // "nop" represents "no operation", which consumes one or several
clock cycles in most architectures, not fixed
            __asm__("nop");
        }
    }
}

```

```

        }
    }

#endif // __DELAY_H__

#endif // __HAL_LL_H__

```

```

#endif // __HAL_LL_H__


// Implement delay by polling systick count
void delay_us(unsigned long __us)
{
    uint32_t ticks;
    uint32_t told, tnow, tcnt = 0;

    ticks = __us * (80000000 / 1000000);
    told = SysTick->VAL;

    while (1)
    {
        tnow = SysTick->VAL;
        if (tnow != told)
        {
            if (tnow < told)
                tcnt += told - tnow;
            else
                tcnt += (SysTick->LOAD + 1) - tnow + told;

            told = tnow;
            if (tcnt >= ticks)
                break;
        }
    }
}

// Precise ms delay implemented with tick timer
void delay_ms(unsigned long ms)
{
    while (ms--)
    {
        delay_us(1000);
    }
}

#else
// Implement delay using systick interrupt, need to enable systick interrupt in
// sysconfig
void delay_ms(unsigned int ms)
{
    delay_times=ms;
    while(delay_times!=0);
}

```

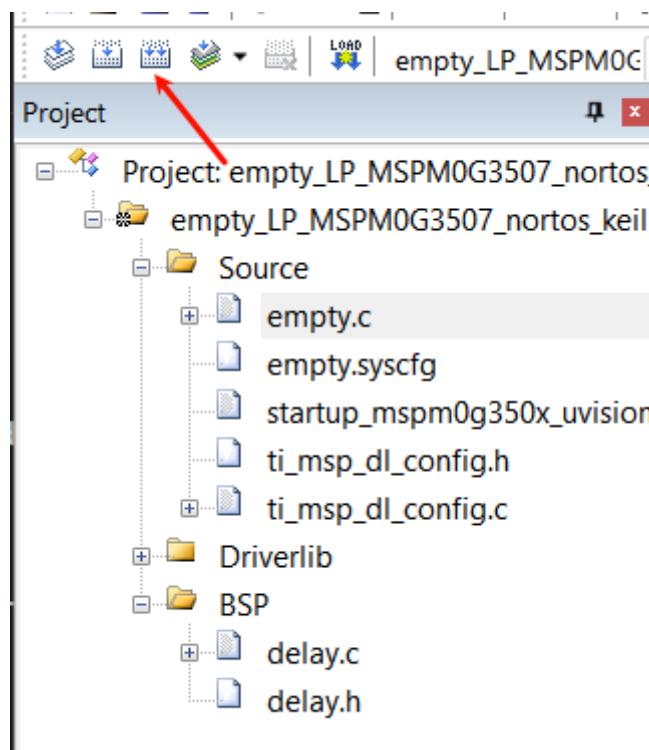
```

void SysTick_Handler(void)
{
    if( delay_times != 0 )
    {
        delay_times--;
    }
}
#endif

```

4. Compile

Click the Rebuild icon. The following prompt indicates that 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
```

IV. 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_set Pins(GPIO_Regs* gpio, uint32_t pins)
{
    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_clear Pins(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_toggle Pins(GPIO_Regs* gpio, uint32_t pins)
{
    gpio->DOUTTGL31_0 = pins;
}

```

__STATIC_INLINE void DL_GPIO_set Pins(GPIO_Regs* gpio, uint32_t pins): This function controls the pin to output high level.

__STATIC_INLINE void DL_GPIO_clear Pins(GPIO_Regs* gpio, uint32_t pins): This function controls the pin to output low level.

__STATIC_INLINE void DL_GPIO_toggle Pins(GPIO_Regs* gpio, uint32_t pins): This function controls the pin level toggle, if it was high level it becomes low level, if it was low level it becomes high level.

V. Experiment Phenomenon

After the program download is complete, you can see the buzzer on the MSPM0 development board starts for 200ms then stops for 200ms.