# F28P55x编程实例Labs-AD采样

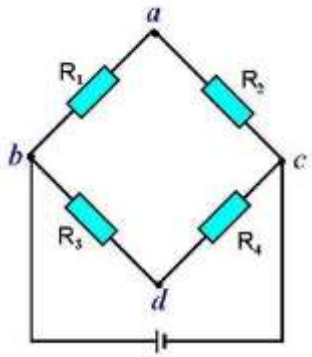
- Code Composer Studio
- C2000Ware
- LaunchXL-F28P55x

# ADC模数转换器

温度信号

压力信号


重量信号




电压信号


C2000


温度传感器


压力传感器

电子秤

# ADC模数转换器

## 参考电压分配

**ADC：**
VREFHIx/VSSA
内部: (0-3.3)V/(0-2.5)V

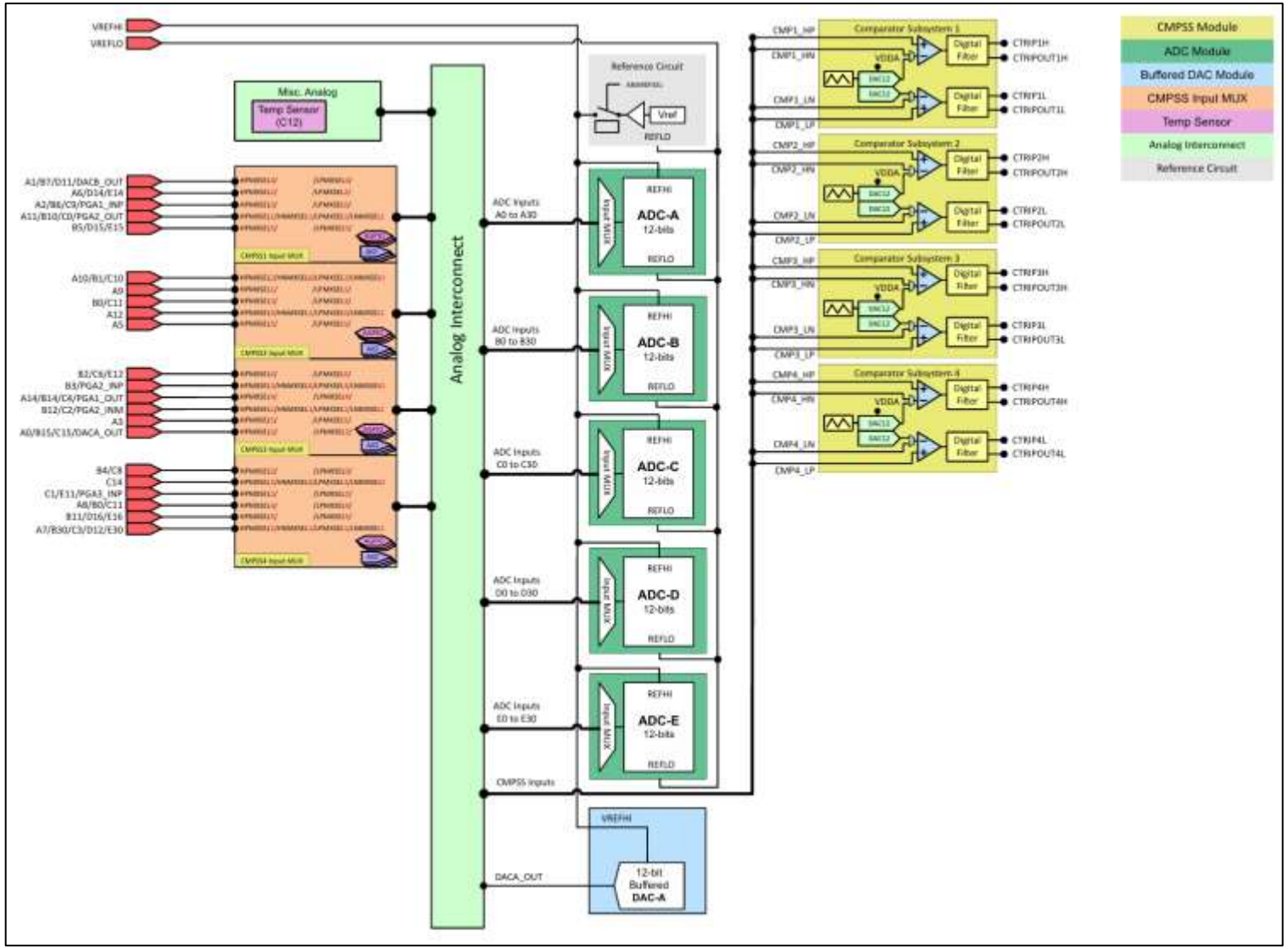
**DAC:**
VREFHIx/VSSA：

**CMPSS:**
VDDA/VSSA

**VREFLO:**
用于ADC的失调电压自我校准

**单端采样模式：**ADCINx、GND

**差分采样模式：**ADCINxP、ADCINxN



# ADC模数转换器

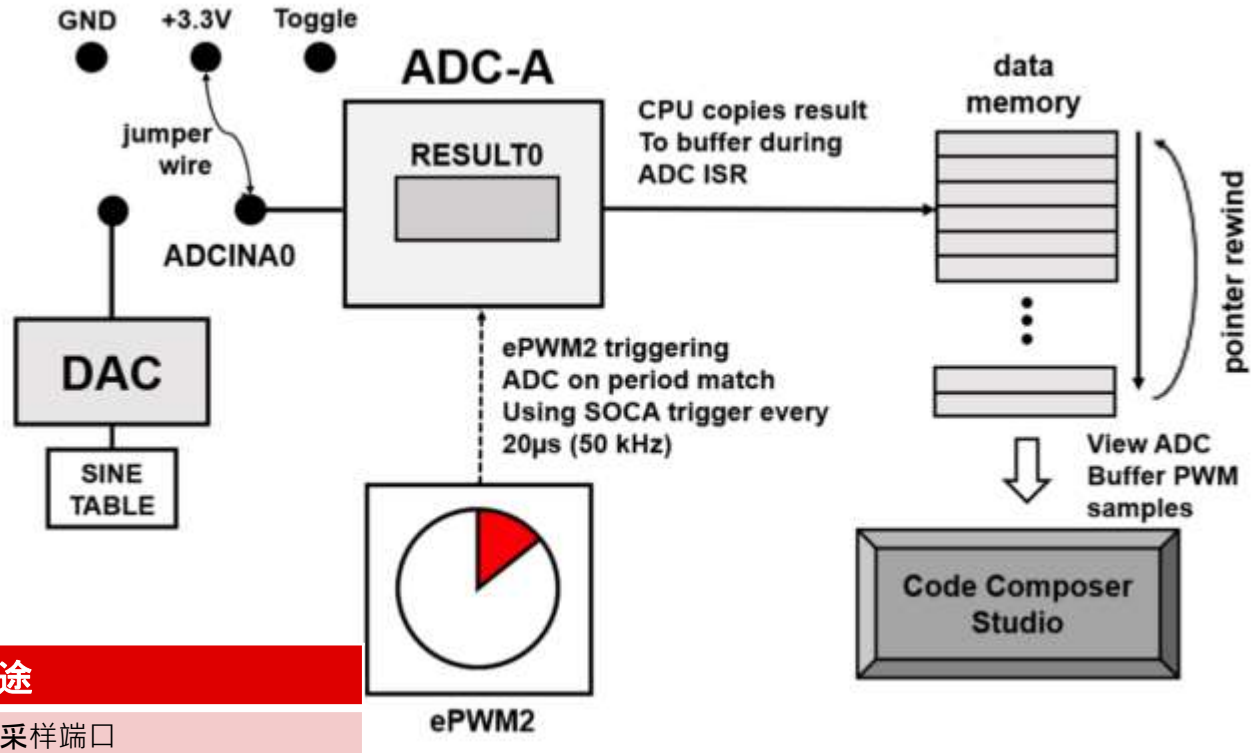
**功能实现**

单片机的EPWM1产生方波，用ADCA0通道采集3.3V和EWPM1，以Debug的方式查看数据采集的变化情况；

GPIO16—高电平
GPIO3 —电平翻转

**实现步骤**

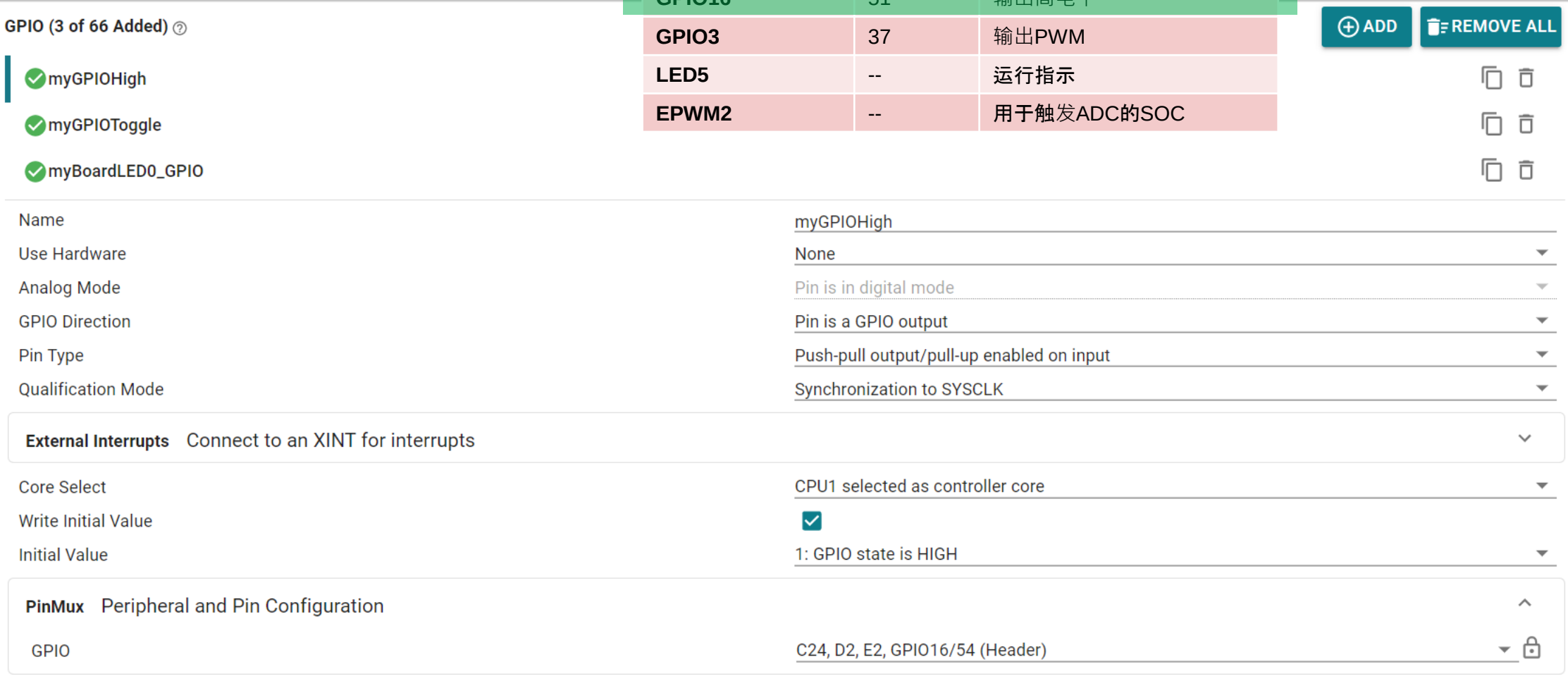
- 复制空白工程
- Sysconfig配置GPIO
- Sysconfig配置EPWM
- Sysconfig配置ADCA0




| GPIO | PIN脚 | 用途 |
|---|---|---|
| ADCINA0 | 70 | AD采样端口 |
| GPIO16 | 51 | 输出高电平 |
| GPIO3 | 37 | 输出PWM |
| LED5 | -- | 运行指示 |
| EPWM2 | -- | 用于触发ADC的SOC |

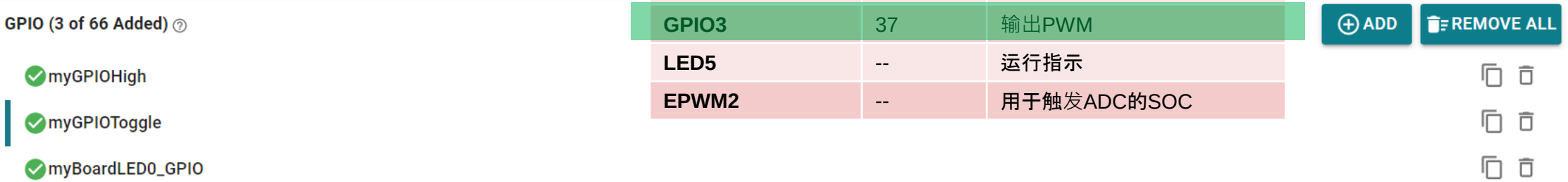
# Sysconfig配置GPIO

| GPIO | PIN脚 | 用途 |
|---|---|---|
| ADCINA0 | 70 | AD采样端口 |
| GPIO16 | 51 | 输出高电平 |
| GPIO3 | 37 | 输出PWM |
| LED5 | -- | 运行指示 |
| EPWM2 | -- | 用于触发ADC的SOC |



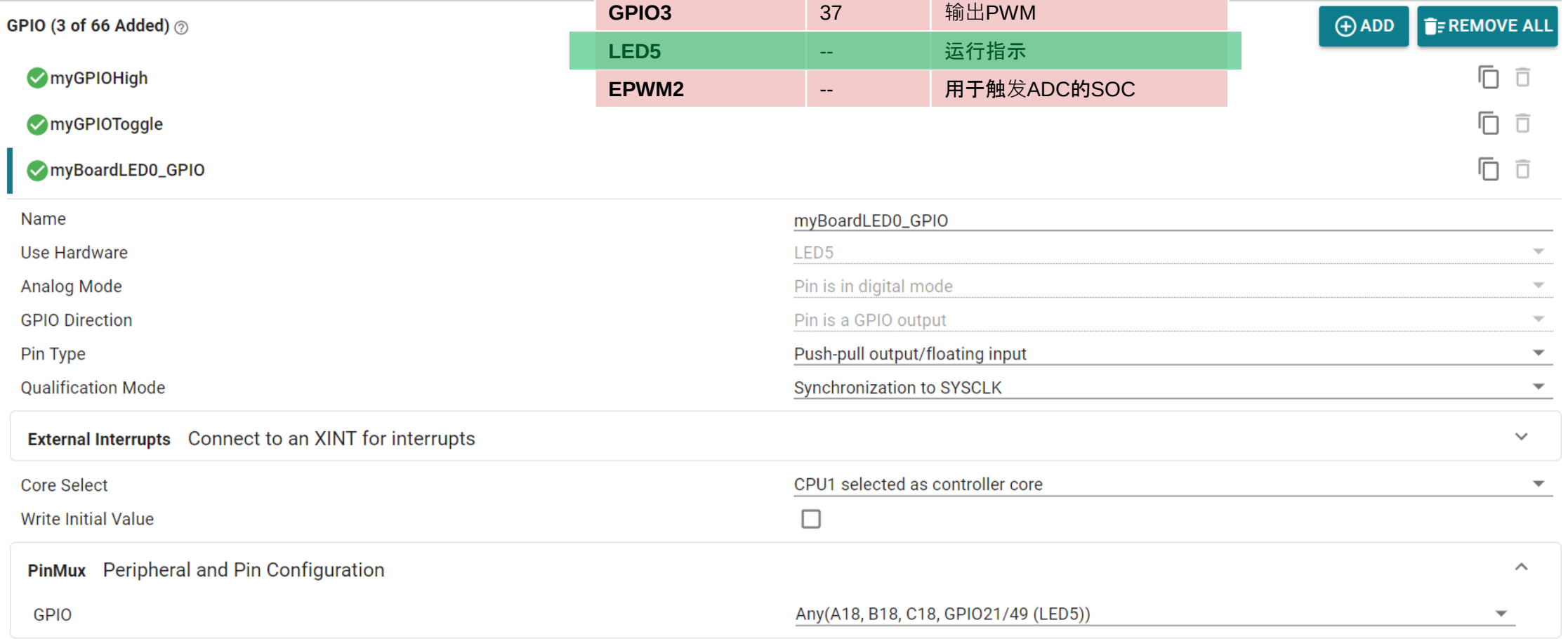
# Sysconfig配置GPIO

| GPIO | PIN脚 | 用途 |
|---|---|---|
| ADCINA0 | 70 | AD采样端口 |
| GPIO16 | 51 | 输出高电平 |
| GPIO3 | 37 | 输出PWM |
| LED5 | -- | 运行指示 |
| EPWM2 | -- | 用于触发ADC的SOC |



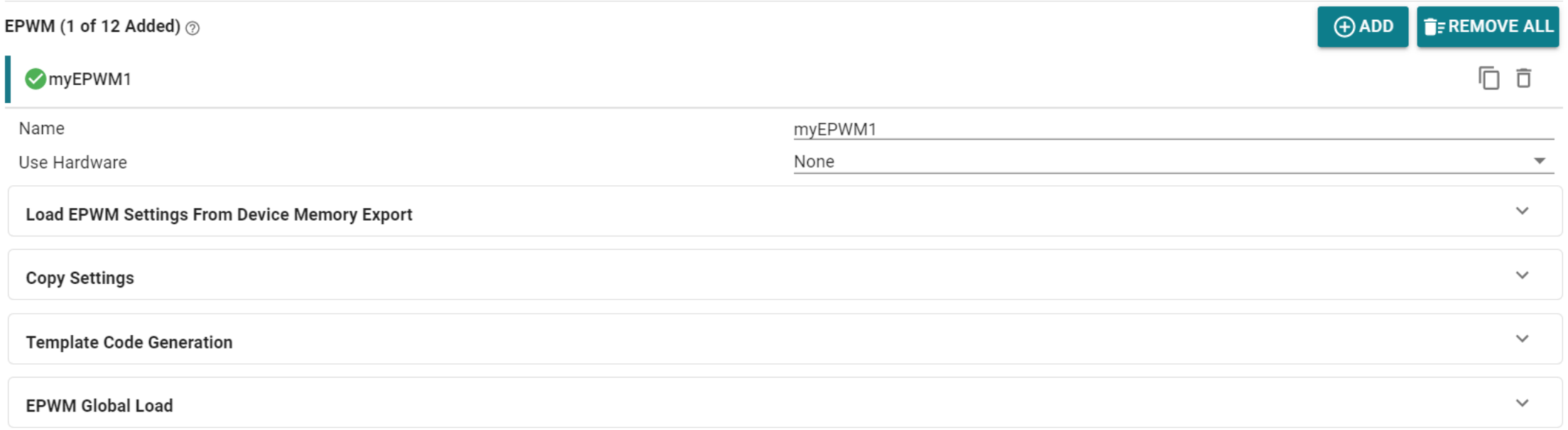
# Sysconfig配置GPIO

| GPIO | PIN脚 | 用途 |
|---|---|---|
| ADCINA0 | 70 | AD采样端口 |
| GPIO16 | 51 | 输出高电平 |
| GPIO3 | 37 | 输出PWM |
| LED5 | -- | 运行指示 |
| EPWM2 | -- | 用于触发ADC的SOC |



# Sysconfig配置EPWM

| GPIO | PIN脚 | 用途 |
|---|---|---|
| ADCINA0 | 70 | AD采样端口 |
| GPIO16 | 51 | 输出高电平 |
| GPIO3 | 37 | 输出PWM |
| LED5 | -- | 运行指示 |
| EPWM2 | -- | 用于触发ADC的SOC |



# Sysconfig配置EPWM

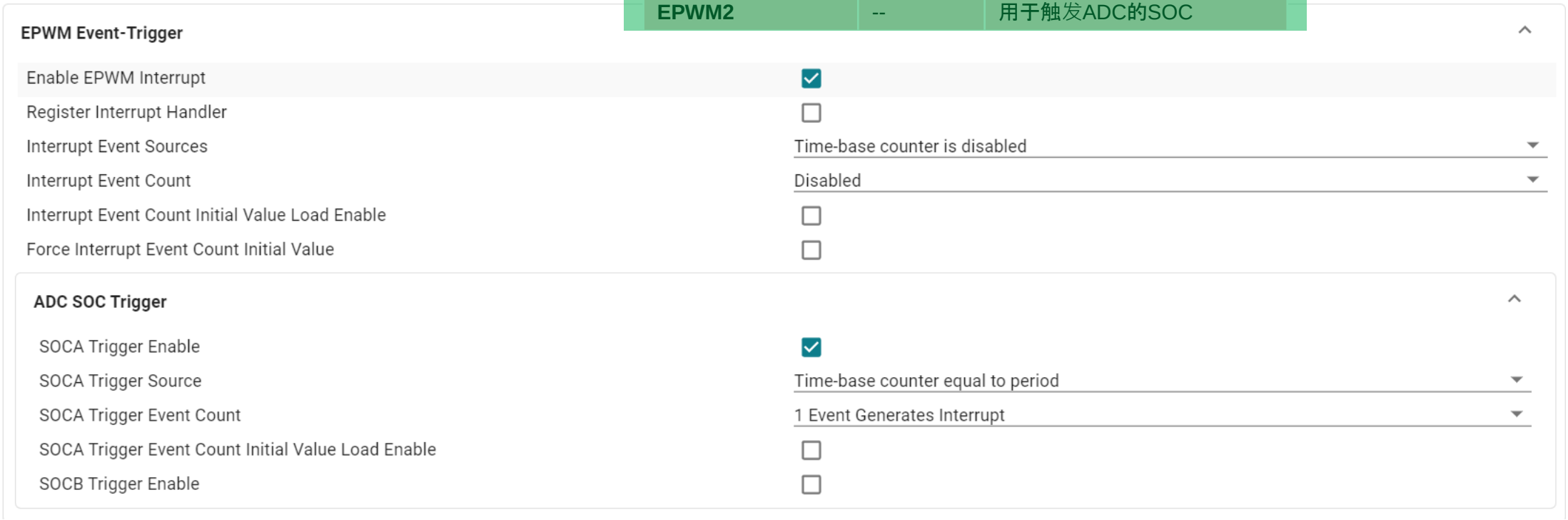
| GPIO | PIN脚 | 用途 |
|---|---|---|
| ADCINA0 | 70 | AD采样端口 |
| GPIO16 | 51 | 输出高电平 |
| GPIO3 | 37 | 输出PWM |
| LED5 | -- | 运行指示 |
| EPWM2 | -- | 用于触发ADC的SOC |

**EPWM Time Base**

| | |
|---|---|
| Emulation Mode | |
| | |
| Time Base Clock Divider | For perfectly synchronized TBCLKs across multiple EPWM modules, the prescaler bits in the TBCTL register of each EPWM module must be set identically |
| High Speed Clock Divider | Divide clock by 1 |
| Time Base Period Load Mode | PWM Period register access is through shadow register |
| Time Base Period Load Event | Shadow to active load occurs when time base counter reaches 0 |
| Time Base Period | 1999 |
| Time Base Period Link | Disable Linking |
| Enable Time Base Period Global Load | ☐ |
| Initial Counter Value | 0 |
| Counter Mode | Up - count mode |
| Enable Phase Shift Load | ☐ |
| Force a Sync Pulse | ☐ |
| Sync In Pulse Source | Sync-in source is EPWM1 sync-out signal |
| Sync Out Pulse | None |
| One-Shot Sync Out Trigger | Trigger is OSHT sync |
| EPWMxSYNCPER Source Select | Counter equals Period |

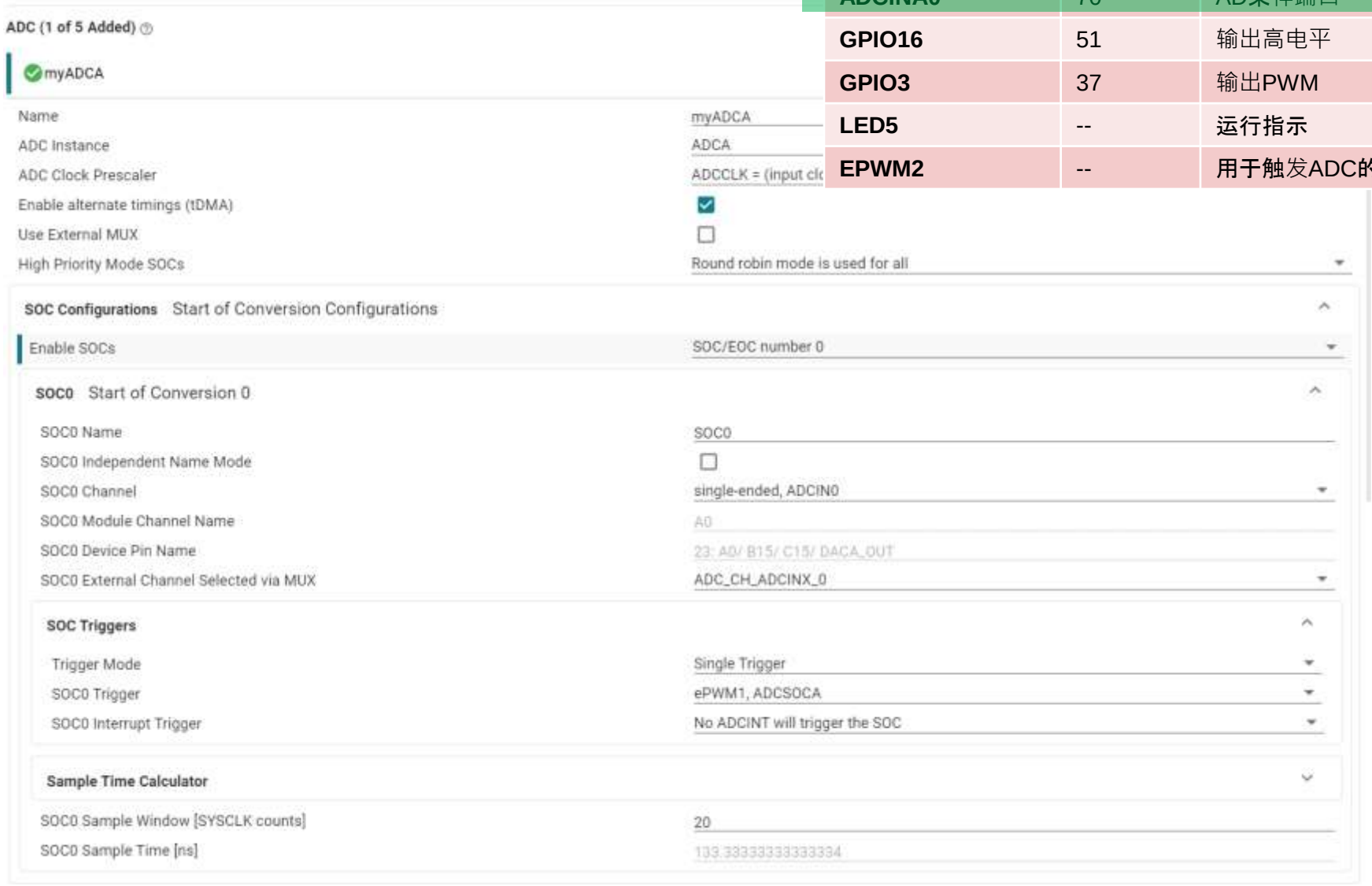
# Sysconfig配置EPWM

| GPIO | PIN脚 | 用途 |
|---|---|---|
| ADCINA0 | 70 | AD采样端口 |
| GPIO16 | 51 | 输出高电平 |
| GPIO3 | 37 | 输出PWM |
| LED5 | -- | 运行指示 |
| EPWM2 | -- | 用于触发ADC的SOC |



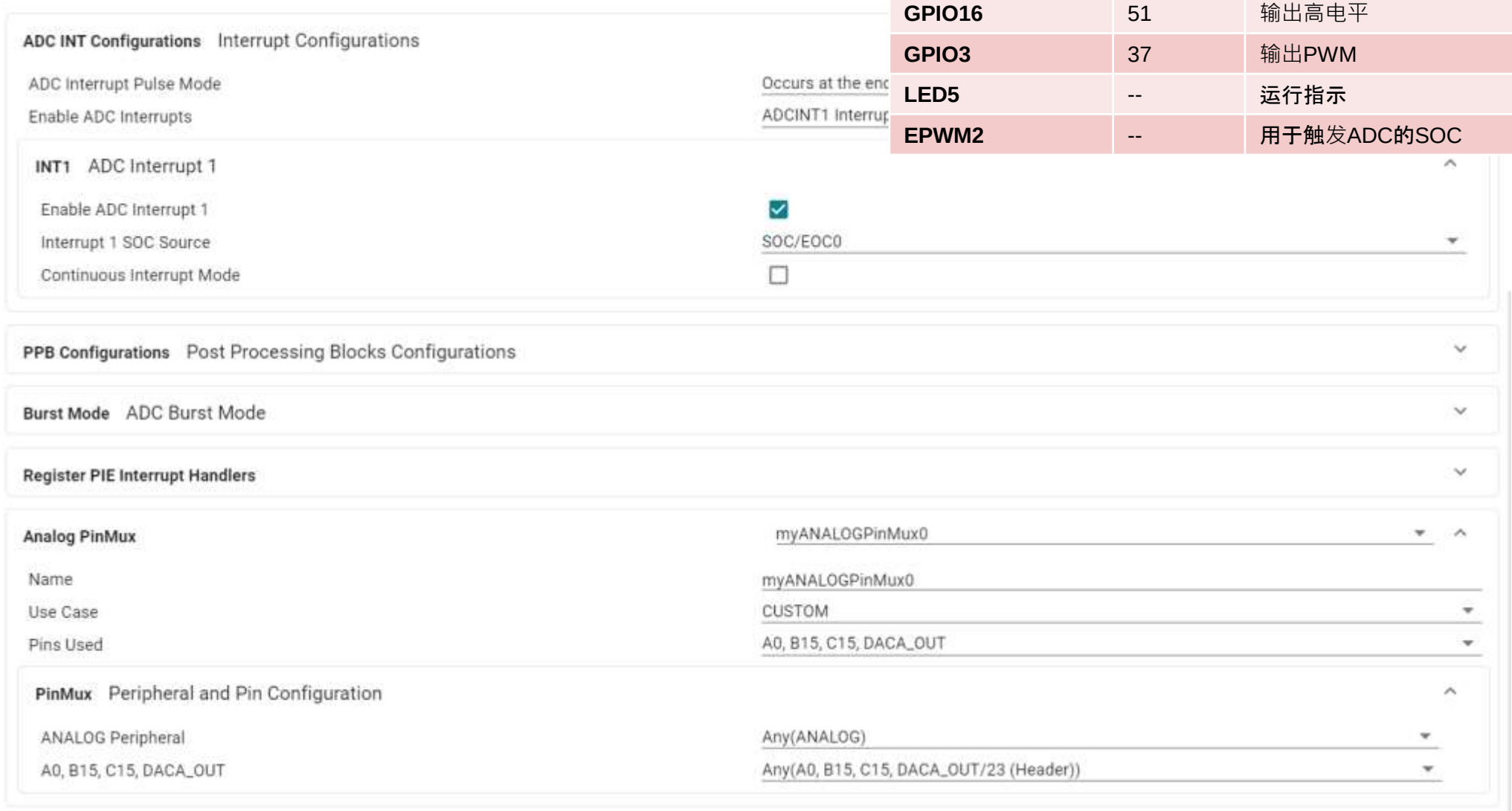
# Sysconfig配置ADCA0

| GPIO | PIN脚 | 用途 |
|---|---|---|
| ADCINA0 | 70 | AD采样端口 |
| GPIO16 | 51 | 输出高电平 |
| GPIO3 | 37 | 输出PWM |
| LED5 | -- | 运行指示 |
| EPWM2 | -- | 用于触发ADC的SOC |



# Sysconfig配置ADCA0

| GPIO | PIN脚 | 用途 |
|---|---|---|
| ADCINA0 | 70 | AD采样端口 |
| GPIO16 | 51 | 输出高电平 |
| GPIO3 | 37 | 输出PWM |
| LED5 | -- | 运行指示 |
| EPWM2 | -- | 用于触发ADC的SOC |



```c
#include "board.h"



#define ADC_BUF_LEN         50
uint16_t DEBUG_TOGGLE = 1;    // Used for real-time mode
uint16_t AdcBuf[ADC_BUF_LEN];  // ADC buffer allocation


 //
 // Function Declarations
 //
 __interrupt void INT_myADCA_1_ISR(void);



void main(void)
{
    // CPU Initialization
    Device_init();
    Interrupt_initModule();
    Interrupt_initVectorTable();

    // Configure the GPIOs/ADC/PWM through SysConfig generated files
    Board_init();

    // Enable global interrupts and real-time debug
    EINT;
    ERTM;

    // Main Loop
    while(1){}

}
```

```c
interrupt void INT_myADCA_1_ISR(void)
{
    static uint16_t *AdcBufPtr = AdcBuf;
    uint16_t LED_count = 0;

    // Read the ADC Result
    *AdcBufPtr++ = ADC_readResult(myADCA_RESULT_BASE, myADCA_SOC0);

    // Brute Force the circular buffer
    if (AdcBufPtr == (AdcBuf + ADC_BUF_LEN))
    {
        AdcBufPtr = AdcBuf;
    }

    // Toggle the pin
    if(DEBUG_TOGGLE == 1)
    {
        GPIO_togglePin(myGPIOToggle);
    }

    if(LED_count++ > 25000)                    // Toggle slowly to see the LED blink
    {
        GPIO_togglePin(myBoardLED0_GPIO);                  // Toggle the pin
        LED_count = 0;                         // Reset the counter
    }
    Interrupt_clearACKGroup(INT_myADCA_1_INTERRUPT_ACK_GROUP);
    ADC_clearInterruptStatus(myADCA_BASE, ADC_INT_NUMBER1);
} // End of ADC ISR
```