**SPC2188 EQEP 使用指南**

# 概述

本应用指南介绍如何有效地利用其内置的增强型正交编码脉冲（Enhanced Quadrature Encoder Pulse, eQEP）模块。该模块是用于精确位置检测和速度检测，尤其适用于工业自动化、机器人技术、伺服驱动器以及其他需要高精度反馈系统的应用场景。

# 目录

# 图片列表

# 表格列表

# 版本历史

| 版本 | 日期 | 作者 | 状态 | 变更 |
|---|---|---|---|---|
| C/0 | 2024-03-11 | 何序清 | Released | 首次发布 |
| | | | | |
| | | | | |

# 术语或缩写

| 术语或缩写 | 描述 |
| --- | --- |
| MCU | Microcontroller Unit，微控制器单元 |
| | |
| | |
| | |
| | |

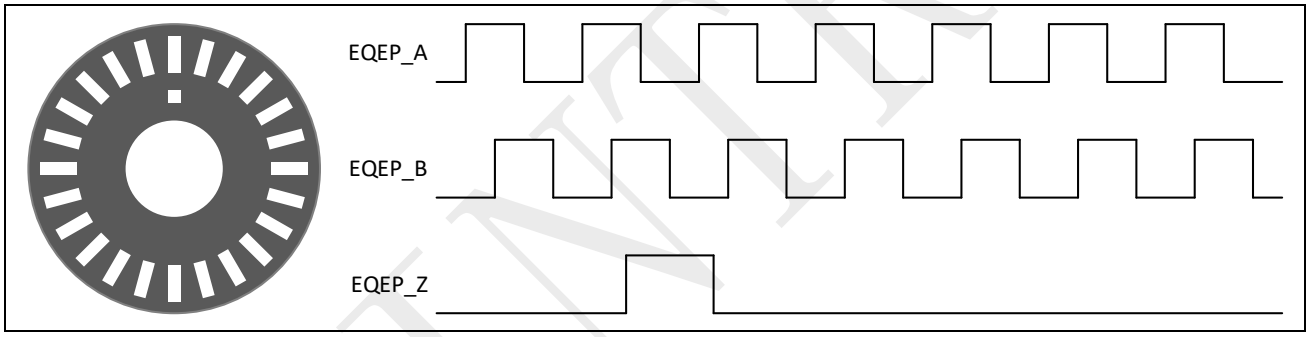
# 1 特性

SPC2188 提供 2 个 EQEP 模块，其特性如下：

- 模块时钟使能控制独立可配，最高频率同 CPU；
- 支持外部 A/B/Z/S 输入滤波和极性选择；
- 支持 4 种 32 位解码计数：正交计数、双向计数、单向计数、顺逆时针计数；
- 灵活的位置计数值加载、复位和锁存，以用于位置测量；
- 边沿捕获和定时器锁存，以用于速度或者频率的测量；
- 提供看门狗定时器，用于停转的检测；

编码器是一种用于测量旋转角度、位置、速度等物理量的装置，常用于机器人、电机、机械设备等领域。如图 1-1 所示，增量式编码器的码盘上通常有内外圈 2 类槽道。当码盘旋转时外圈槽道对应的两个光电元件产生明暗相间且相位差为 90° 的正交信号，记为 EQEP_A 和 EQEP_B；码盘旋转一圈，内圈槽道对应光电元件产生指示绝对零点位置电信号，此信号记为 EQEP_Z；

**图 1-1：光学增量式编码器的码盘和输出信号波形**



# 2 电机测速方法

## 2.1 M 法

M 法又称测角法，即在规定的时间间隔$T_g$内，测量编码器所产生的脉冲数来计算转速。电机每转一圈脉冲发生器发出的脉冲数为 P，而在规定的时间间隔$T_g$内，测得的总脉冲数$m_1$，则电机每分钟转速：

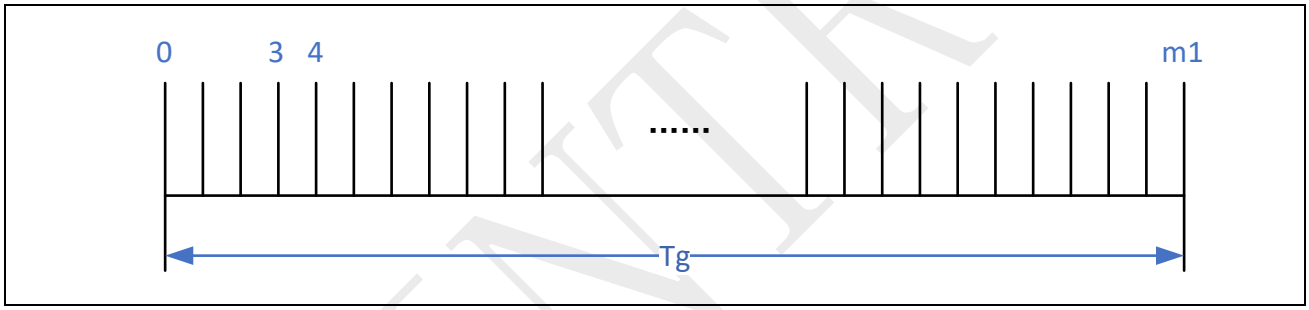
$$n = 60 * \frac{m_1}{P * T_g}(r/min)$$

P：电机每转一圈脉冲发生器发出脉冲数；

$T_g$：测量时间（单位：秒）

$m_1$：在时间间隔$T_g$内，脉冲发生器发出总脉冲数；

**图 2-1：M 法**




**注意：** **$T_g$时间内测量$m_1$值越大，测量转速越准确，M 法适合测量高速。**

## 2.2 T 法

T 法也称之为测周法，即测量相邻两个脉冲的时间间隔来确定被测速度的方法，T 法的实现需要用到一已知频率为$f_c$的高频脉冲，在两个相邻脉冲之间（相邻脉冲之间周期为 T）计算高频脉冲数$m_1$。电机每转一圈脉冲发生器发出的脉冲数为 P，则电机每分钟转速：

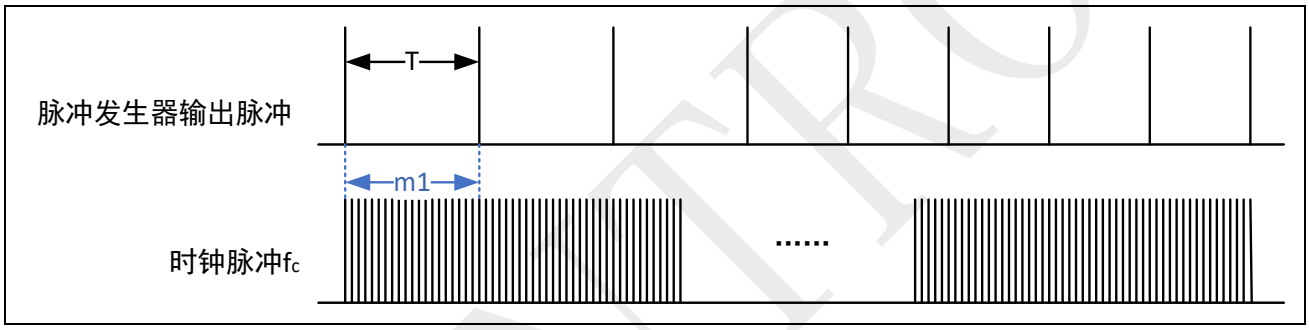
$$n = 60 * \frac{f_c}{P * m_1} \quad （r/min）$$

P：电机每转一圈脉冲发生器发出脉冲数；

$f_c$：高频时钟 1s 产生的脉冲数；

$m_1$：脉冲发生器一个周期 T，高频时钟的脉冲数；

**图 2-2：T 法**




**注意：　　一个周期 T 时间内测量$m_1$值越大，测量转速越准确，T 法适合测量慢速。**

## 2.3 M/T 法

M/T 法测速是将 M 法和 T 法两种方法结合在一起使用，即测量编码器脉冲数又测量一定时间内的高频脉冲数，则电机每分钟转速：

$$n = 60 * \frac{m_1 * f_c}{P * m_2}$$

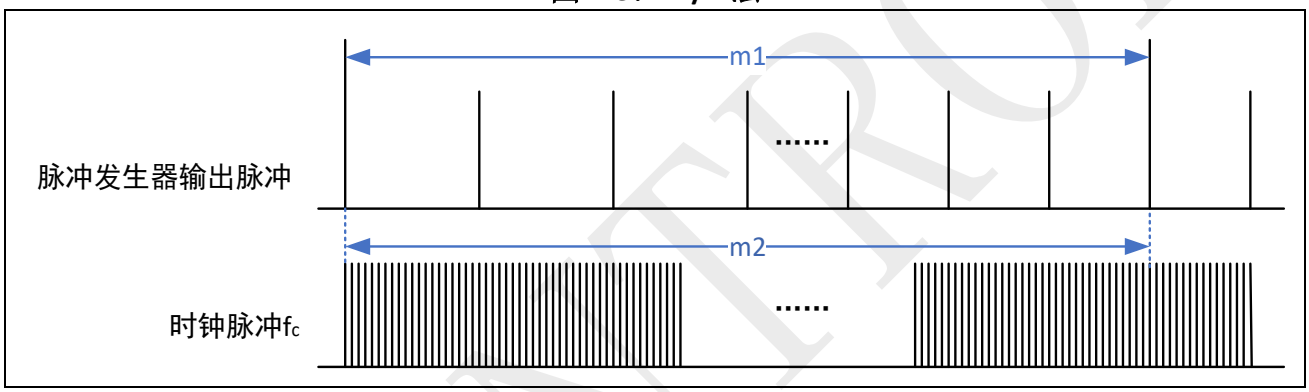
P：电机每转一圈脉冲发生器发出脉冲数；

$f_c$：高频时钟 1s 产生的脉冲数；

$m_1$：脉冲发生器发出总脉冲数；

$m_2$：高频时钟的脉冲数；

**图 2-3：M/T 法**



# 3 实例

## 3.1 测量转速信息

实际应用当中一般都采用正交计数来获取速度信息，在正交计数模式时，QA 和 QB 的边沿都会影响位置计数值和方向变化。其 QA 和 QB 对应的真值表和状态机以及时序图如下所示。

**表 3-1：正交计数真值表**

| 上个边沿 | 当前边沿 | 方向（QDIR） | 位置计数值（QPOSCNT） |
|---|---|---|---|
| QA ↑ | QB ↑ | 正 | 增加 |
| | QB ↓ | 负 | 减小 |
| | QA ↓ | 正/负→负/正 | 减小/增加 |
| QA ↓ | QB ↓ | 正 | 增加 |
| | QB ↑ | 负 | 减小 |
| | QA ↑ | 正/负→负/正 | 减小/增加 |
| QB ↑ | QA ↑ | 正 | 减小 |
| | QA ↑ | 负 | 增加 |
| | QB ↓ | 正/负→负/正 | 减小/增加 |
| QB ↓ | QA ↓ | 正 | 减小 |
| | QA ↑ | 负 | 增加 |
| | QB ↑ | 正/负→负/正 | 减小/增加 |

**图 3-1：正交解码状态机**

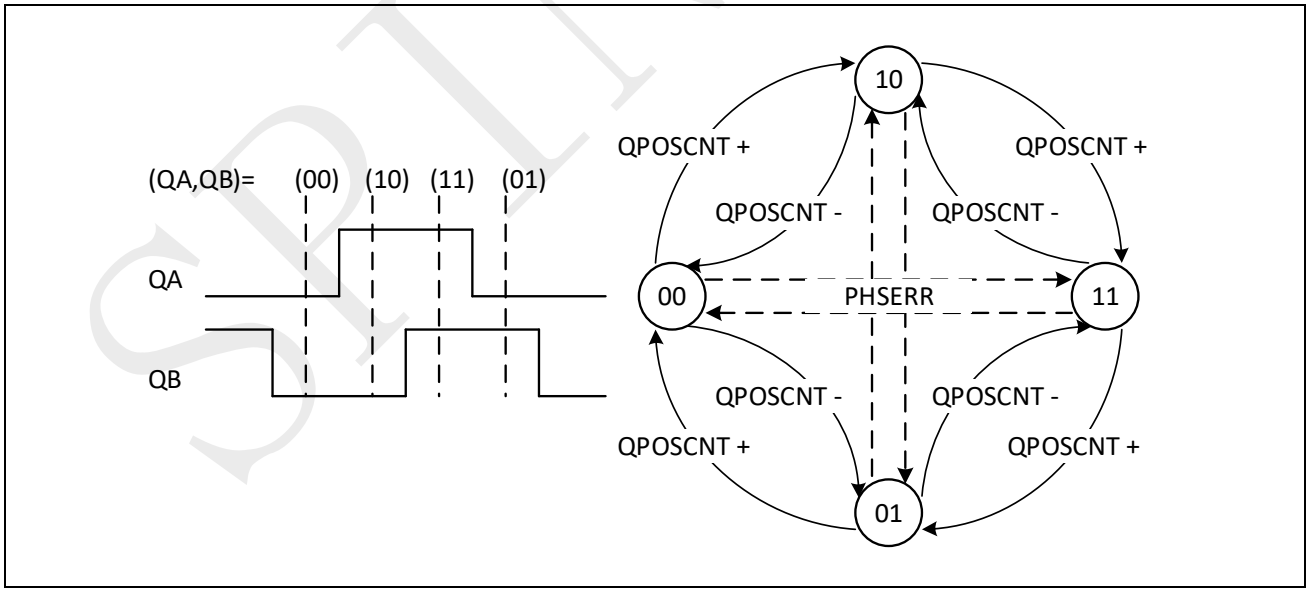
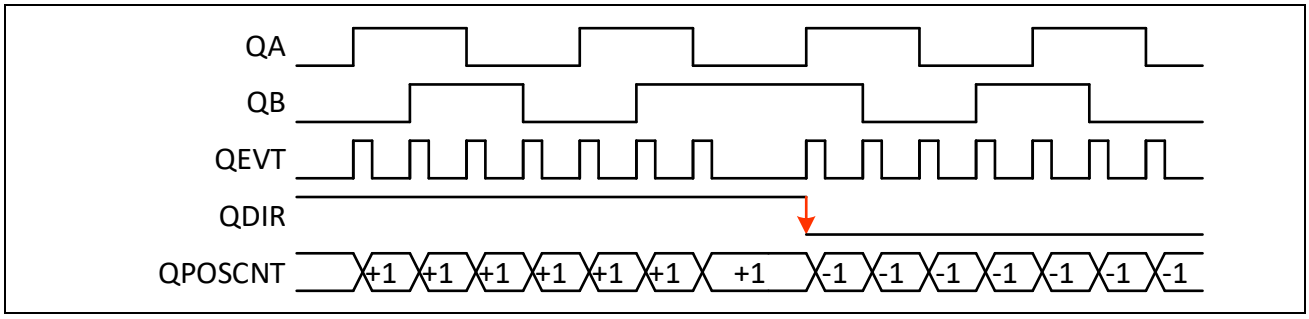


图 3-2：正交计数时序图




### 3.1.1 M 法测速

本示例演示 EQEP 使用 EQEP_A 和 EQEP_B 引脚采集一对正交信号，使用 M 法的方法计数电机转速，例如 A、B 引脚输入一对正交信号是 500K，进行模拟测量电机的转速，其配置程序配置流程如下：

- 初始化系统时钟，UART 调试口以及初始化 EQEP 的 GPIO；
- 初始化 EQEP，并配置正交计数；
- 配置 EQEP 的计数值为 0；
- 使能 EQEP 的定时器功能，并设置进行 5s 计数；
- 使能 EQEP 定时器计数超时 counter 计数器进行复位；
- 使能 EQEP 定时器超时中断；
- 使能 SDFM；

**M 法测速**

```
#include "spc2188.h"

#include <stdio.h>

#define RESOLUTION      500     /* The number of pulses generated by one
revolution of the encoder */
#define MEASURE_TIME    5       /* s(Units of time) */


uint32_t u32EQEPModuleClk = 0;  /* EQEP CLK */
int32_t  i32EQEPCount = 0;      /* EQEP total count */
int32_t  i32RotateSpeed = 0;    /* rotate speed */


int main(void)
{
    uint32_t u32ReloadValue = 0;

    CLOCK_InitWithRCO(240000000U);
    Delay_Init();

    PIN_SetChannel(PIN_GPIO62, PIN_GPIO62_UART0_TXD);
    PIN_SetChannel(PIN_GPIO63, PIN_GPIO63_UART0_RXD);
    UART_Init(UART0, 38400);
```

```
    printf("EQEP test.....\n");

    /* Config EQEP GPIO */
    PIN_SetChannel(PIN_GPIO52, PIN_GPIO52_EQEP0_A);
    PIN_SetChannel(PIN_GPIO53, PIN_GPIO53_EQEP0_B);
    PIN_SetChannel(PIN_GPIO54, PIN_GPIO54_EQEP0_Z);

    /* Init EQEP */
    EQEP_Init(EQEP0, EQEP_MODE_QUADRATURE_COUNT);

    /* Set IO invert input or normall input */
    EQEP_SetQa(EQEP0, EQEP_INPUT_ORIGINAL);
    EQEP_SetQb(EQEP0, EQEP_INPUT_ORIGINAL);
    EQEP_SetQz(EQEP0, EQEP_INPUT_ORIGINAL);

    /* Init position counter */
    EQEP_SetPositionCounterValue(EQEP0, 0);

    /* Set maximum position */
    EQEP_SetMaxPosition(EQEP0, 0xFFFFFFFF);

    /* Set position counter reload value */
    EQEP_SetPositionReloadValue(EQEP0, 0);

    /* Set Unit-timer period */
    u32EQEPModuleClk = CLOCK_GetModuleClock(EQEP0_MODULE);

    /* Set Unit-timer count value */
    u32ReloadValue = u32EQEPModuleClk * MEASURE_TIME;
    EQEP_SetUnitTimerReloadValue(EQEP0, u32ReloadValue);
    EQEP_SetUnitTimerCounterValue(EQEP0, u32ReloadValue);

    /* EQEP Set counter timer timeout mode */
    EQEP_SetCounterResetMode(EQEP0, EQEP_RESET_ON_UNIT_TIMER_TIMEOUT);

    /* Enable timer Timeout int */
    EQEP_EnableInt (EQEP0, EQEP_INT_UNIT_TIMER_TIMEOUT);
    NVIC_EnableIRQ( EQEP0_IRQn ) ;

    /* Enable Unit-timer */
    EQEP_EnableUnitTimer(EQEP0) ;

    /* Enable EQEP */
    EQEP_Enable(EQEP0) ;


    while (1)
    {

    }
}


void EQEP0_IRQHandler(void)
{
    int64_t i64Temp = 0;

    if ( EQEP_GetIntFlag(EQEP0, EQEP_INT_UNIT_TIMER_TIMEOUT))
    {
        i32EQEPCount = EQEP_GetLatchedPositionOnUnitTimerTimeout(EQEP0);
        i64Temp = i32EQEPCount;
        i64Temp = i64Temp * 60;
        i32RotateSpeed = i64Temp / (RESOLUTION * 4 * MEASURE_TIME);
```

```
        printf("Measure Rotational Speed : %d(r/min)\n", i32RotateSpeed);

        EQEP_ClearInt(EQEP0, EQEP_INT_UNIT_TIMER_TIMEOUT);
    }

    EQEP_ClearInt(EQEP0, EQEP_INT_GLOBAL);
}
```

### 3.1.2 T 法测速

本示例演示 EQEP 使用 EQEP_A 和 EQEP_B 引脚采集一对正交信号，使用 T 法的方法计数电机转速，例如 A、B 引脚输入一对正交信号是 500K，进行模拟测量电机的转速，其配置程序配置流程如下：

- 初始化系统时钟，UART 调试口以及初始化 EQEP 的 GPIO；
- 初始化 EQEP，并配置正交计数；
- 配置 EQEP 的计数值为 0；
- 使能 EQEP 的事件捕获功能；
- 设置 EQEP 的事件捕获分频比；
- 使能 EQEP 事件捕获中断；
- 使能 SDFM；

**T 法测速**

```
#include "spc2188.h"

#include <stdio.h>

#define RESOLUTION      500     /* The number of pulses generated by one
revolution of the encoder */


uint32_t u32EQEPModuleClk = 0;  /* EQEP CLK */
int32_t  i32EQEPCount = 0;      /* EQEP total count */
int32_t  i32RotateSpeed = 0;    /* rotate speed */
uint32_t u32CaptureEventDiv = 0;/* Capture Event Div */

int main(void)
{
    CLOCK_InitWithRCO(240000000U);
    Delay_Init();

    PIN_SetChannel(PIN_GPIO62, PIN_GPIO62_UART0_TXD);
    PIN_SetChannel(PIN_GPIO63, PIN_GPIO63_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("EQEP test.....\n");

    /* Config EQEP GPIO */
    PIN_SetChannel(PIN_GPIO52, PIN_GPIO52_EQEP0_A);
    PIN_SetChannel(PIN_GPIO53, PIN_GPIO53_EQEP0_B);

    /* Init EQEP */
    EQEP_Init(EQEP0, EQEP_MODE_QUADRATURE_COUNT);

    /* Set IO invert input or normall input */
    EQEP_SetQa(EQEP0, EQEP_INPUT_ORIGINAL ) ;
    EQEP_SetQb(EQEP0, EQEP_INPUT_ORIGINAL ) ;


    /* Init position counter */
    EQEP_SetPositionCounterValue(EQEP0, 0 ) ;
```

```
    /* Set maximum position */
    EQEP_SetMaxPosition(EQEP0, 0xFFFFFFFF) ;

    /* Set position counter reload value */
    EQEP_SetPositionReloadValue(EQEP0, 0 ) ;

    /* Get EQEP Clock */
    u32EQEPModuleClk = CLOCK_GetModuleClock(EQEP0_MODULE);

    /* Set capture count value */
    EQEP_SetCaptureTimerCount(EQEP0, 0);

    /* Set capture Timer clock divider */
    EQEP_SetCaptureTimerClockDiv(EQEP0, EQEP_CTIMER_CLK_DIV_1);

    /* Set capture Timer event divider */
    EQEP_SetCaptureEventDiv(EQEP0, EQEP_QEVT_DIV_4);
    u32CaptureEventDiv = EQEP_GetCaptureEventDiv(EQEP0);
    u32CaptureEventDiv = 1 << u32CaptureEventDiv;

    /* Enable capture module run */
    EQEP_EnableCapture(EQEP0);

    /* Enable Capture Event int */
    EQEP_EnableInt (EQEP0, EQEP_INT_CAPTURE_EVENT );
    NVIC_EnableIRQ( EQEP0_IRQn );

    /* Enable EQEP */
    EQEP_Enable(EQEP0);


    while (1)
    {

    }
}


void EQEP0_IRQHandler(void)
{
    int64_t i64Temp1 = 0;
    int64_t i64Temp2 = 0;

    if ( EQEP_GetIntFlag(EQEP0, EQEP_INT_CAPTURE_EVENT) )
    {
        i32EQEPCount = EQEP_GetCaptureTimerPeriod(EQEP0);

        i64Temp1 = u32EQEPModuleClk;
        i64Temp1 = i64Temp1 * 60;
        i64Temp2 = i32EQEPCount;
        i64Temp2 = i64Temp2 * RESOLUTION;
        i32RotateSpeed = i64Temp1 / i64Temp2;
        printf("Measure Rotational Speed : %d(r/min)\n", i32RotateSpeed);

        EQEP_ClearInt(EQEP0, EQEP_INT_CAPTURE_EVENT) ;
    }

    EQEP_ClearInt(EQEP0, EQEP_INT_GLOBAL) ;
}
```

### 3.1.3 M/T 法测速

本示例演示 EQEP 使用 EQEP_A 和 EQEP_B 引脚采集一对正交信号，使用 M/T 法的方法计数电机转速，例如 A、B 引脚输入一对正交信号是 500K，进行模拟测量电机的转速，其配置程序配置流程如下：

- 初始化系统时钟，UART 调试口以及初始化 EQEP 的 GPIO；
- 初始化 EQEP，并配置正交计数；
- 配置 EQEP 的计数值为 0；
- 使能 EQEP 的定时器功能，并设置进行 5s 计数；
- 使能 EQEP 定时器计数超时 counter 计数器进行复位；
- 使能 EQEP 定时器超时中断；
- 使能 SDFM；

**M/T 法测速**

```
#include "spc2188.h"

#include <stdio.h>

#define RESOLUTION      500     /* The number of pulses generated by one
revolution of the encoder */
#define MEASURE_TIME    1       /* s(Units of time) */


uint32_t u32EQEPModuleClk = 0;  /* EQEP CLK */
int32_t  i32EQEPCount = 0;      /* EQEP total count */
int32_t  i32RotateSpeed = 0;    /* rotate speed */
uint32_t u32CaptureEventDiv = 0;/* Capture Event Div */
uint32_t u32ReloadValue = 0;    /* Number of loading pulses */


int main(void)
{
    CLOCK_InitWithRCO(240000000U);
    Delay_Init();

    PIN_SetChannel(PIN_GPIO62, PIN_GPIO62_UART0_TXD);
    PIN_SetChannel(PIN_GPIO63, PIN_GPIO63_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("EQEP test.....\n");

    /* Config EQEP GPIO */
    PIN_SetChannel(PIN_GPIO52, PIN_GPIO52_EQEP0_A);
    PIN_SetChannel(PIN_GPIO53, PIN_GPIO53_EQEP0_B);

    /* Init EQEP */
    EQEP_Init(EQEP0, EQEP_MODE_QUADRATURE_COUNT);

    /* Set IO invert input or normall input */
    EQEP_SetQa(EQEP0, EQEP_INPUT_ORIGINAL ) ;
    EQEP_SetQb(EQEP0, EQEP_INPUT_ORIGINAL ) ;
```

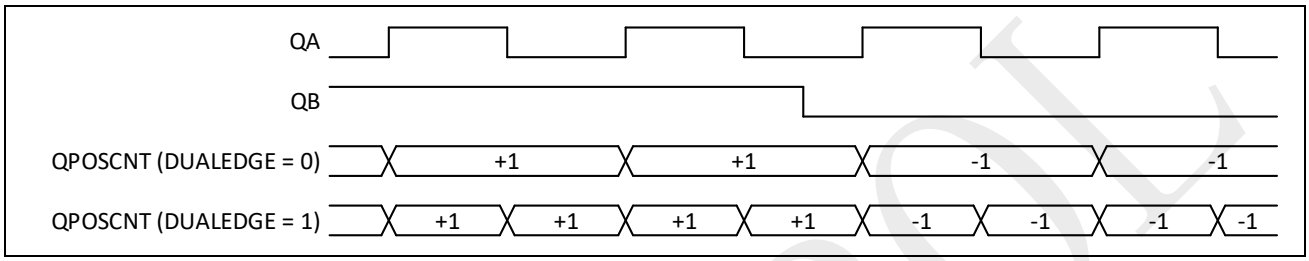
```
    /* Init position counter */
    EQEP_SetPositionCounterValue(EQEP0, 0 ) ;

    /* Set maximum position */
    EQEP_SetMaxPosition(EQEP0, 0xFFFFFFFF) ;

    /* Set position counter reload value */
    EQEP_SetPositionReloadValue(EQEP0, 0 ) ;

    /* Get EQEP Clock */
    u32EQEPModuleClk = CLOCK_GetModuleClock(EQEP0_MODULE);

    /* Set Unit-timer count value */
    u32ReloadValue = u32EQEPModuleClk * MEASURE_TIME;
    EQEP_SetUnitTimerReloadValue(EQEP0, u32ReloadValue);
    EQEP_SetUnitTimerCounterValue(EQEP0, u32ReloadValue);

    /* EQEP Set counter timer timeout mode */
    EQEP_SetCounterResetMode(EQEP0, EQEP_RESET_ON_UNIT_TIMER_TIMEOUT);

    /* Enable Capture Event int */
    EQEP_EnableInt (EQEP0, EQEP_INT_UNIT_TIMER_TIMEOUT);
    NVIC_EnableIRQ( EQEP0_IRQn );


    /* Enable Unit-timer */
    EQEP_EnableUnitTimer(EQEP0) ;

    /* Enable EQEP */
    EQEP_Enable(EQEP0);


    while (1)
    {
    }
}


void EQEP0_IRQHandler(void)
{
    int64_t i64Temp = 0;

    if ( EQEP_GetIntFlag(EQEP0, EQEP_INT_UNIT_TIMER_TIMEOUT))
    {
        i32EQEPCount = EQEP_GetLatchedPositionOnUnitTimerTimeout(EQEP0);
        i64Temp = i32EQEPCount;
        i64Temp = i64Temp * 60 * u32EQEPModuleClk;
        i32RotateSpeed = i64Temp / RESOLUTION / u32ReloadValue / 4;

        printf("Measure Rotational Speed : %d(r/min)\n", i32RotateSpeed);

        EQEP_ClearInt(EQEP0, EQEP_INT_UNIT_TIMER_TIMEOUT);
    }

    EQEP_ClearInt(EQEP0, EQEP_INT_GLOBAL) ;
}
```

## 3.2 测量位置信息

### 3.2.1 双向计数

本示例演示 EQEP 使用双向计数模式获取电机位置信息，当 QB 为高时，进行正向计数（正转），当 QB 为低时，进行反向计数（反转），每当经过过零点时计数器进行复位（编码器每转一圈都过产生一次零点信号）。

图 3-3：双向计数




示例会时刻输出位置信息，其配置程序配置流程如下：

- 初始化系统时钟，UART 调试口以及初始化 EQEP 的 GPIO；
- 初始化 EQEP，并配置双向计数；
- 配置 EQEP 采用上升沿计数；
- 配置 EQEP 的计数值为 0；
- 设置 EQEP 计数器复位模式为每经过零点时进行复位；
- 使能 SDFM；

双向计数

```
#include "spc2188.h"

#include <stdio.h>

int main(void)
{
    uint32_t u32ReloadValue = 0;

    CLOCK_InitWithRCO(240000000U);
    Delay_Init();

    PIN_SetChannel(PIN_GPIO62, PIN_GPIO62_UART0_TXD);
    PIN_SetChannel(PIN_GPIO63, PIN_GPIO63_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("EQEP test.....\n");

    /* Config EQEP GPIO */
    PIN_SetChannel(PIN_GPIO52, PIN_GPIO52_EQEP0_A);
    PIN_SetChannel(PIN_GPIO53, PIN_GPIO53_EQEP0_B);
    PIN_SetChannel(PIN_GPIO54, PIN_GPIO54_EQEP0_Z);
```

```
    /* Init EQEP */
    EQEP_Init(EQEP0, EQEP_MODE_DIRECTION_COUNT);

    /* Set direction up count event mode */
    EQEP_SetCountMode(EQEP0, EQEP_COUNT_ON_RISING_EDGE);

    /* Set IO invert input or normall input */
    EQEP_SetQa(EQEP0, EQEP_INPUT_ORIGINAL);
    EQEP_SetQb(EQEP0, EQEP_INPUT_ORIGINAL);
    EQEP_SetQz(EQEP0, EQEP_INPUT_ORIGINAL);

    /* Init position counter */
    EQEP_SetPositionCounterValue(EQEP0, 0);

    /* Set maximum position */
    EQEP_SetMaxPosition(EQEP0, 0xFFFFFFFF);

    /* Set position counter reload value */
    EQEP_SetPositionReloadValue(EQEP0, 0);

    /* EQEP Set counter reset mode */
    EQEP_SetCounterResetMode(EQEP0, EQEP_RESET_ON_EACH_ZERO_MARKER);

    /* Enable EQEP */
    EQEP_Enable(EQEP0) ;


    while (1)
    {
        printf("EQEP Count : %d\n", EQEP_GetPositionCounterValue(EQEP0));
    }
}
```

### 3.2.2 单向计数

本示例演示 EQEP 使用单向计数模式获取电机位置信息，QA 用作位置计数事件，当 QDECCTL.SGLCNTDIR 为 1 时，进行正向计数（正转），当 QDECCTL.SGLCNTDIR 为 0 时，进行反向计数(反转),每当经过过零点时计数器进行复位(编码器每转一圈都过产生一次零点信号)。

示例会时刻输出位置信息，其配置程序配置流程如下：

- 初始化系统时钟，UART 调试口以及初始化 EQEP 的 GPIO；
- 初始化 EQEP，并配置向上计数；
- 配置 EQEP 的计数值为 0；
- 设置 EQEP 计数器复位模式为每经过零点时进行复位；
- 使能 SDFM；

单向计数

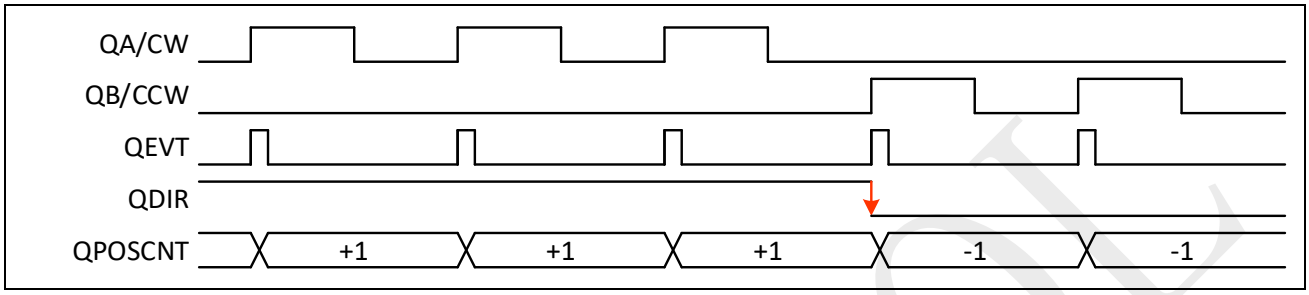
```
#include "spc2188.h"

#include <stdio.h>

int main(void)
{
    CLOCK_InitWithRCO(240000000U);
    Delay_Init();

    PIN_SetChannel(PIN_GPIO62, PIN_GPIO62_UART0_TXD);
    PIN_SetChannel(PIN_GPIO63, PIN_GPIO63_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("EQEP test.....\n");

    /* Config EQEP GPIO */
    PIN_SetChannel(PIN_GPIO52, PIN_GPIO52_EQEP0_A);

    /* Init EQEP */
    EQEP_Init(EQEP0, EQEP_MODE_UP_COUNT);

    /* Set IO invert input or normall input */
    EQEP_SetQa(EQEP0, EQEP_INPUT_ORIGINAL ) ;

    /* Init position counter */
    EQEP_SetPositionCounterValue(EQEP0, 0 ) ;

    /* Set maximum position */
    EQEP_SetMaxPosition(EQEP0, 0xFFFFFFFF) ;

    /* Set position counter reload value */
    EQEP_SetPositionReloadValue(EQEP0, 0 ) ;

    /* EQEP Set counter reset mode */
    EQEP_SetCounterResetMode(EQEP0, EQEP_RESET_ON_EACH_ZERO_MARKER);

    /* Enable EQEP */
    EQEP_Enable(EQEP0);


    while (1)
```

```
    {
        printf("EQEP Count : %d\n", EQEP_GetPositionCounterValue(EQEP0));
    }
}
```

### 3.2.3 顺逆时针计数

本示例演示 EQEP 使用顺逆计数模式获取电机位置信息，当 QA 上升沿表示计数方向为正向计数（正转），当 QB 上升沿表示计数方向为反向计数（反转），每当经过过零点时计数器进行复位（编码器每转一圈都过产生一次零点信号）。

图 3-4：顺逆时针计数




示例会时刻输出位置信息，其配置程序配置流程如下：

- 初始化系统时钟，UART 调试口以及初始化 EQEP 的 GPIO；
- 初始化 EQEP，并配置顺逆计数；
- 配置 EQEP 的计数值为 0；
- 设置 EQEP 计数器复位模式为每经过零点时进行复位；
- 使能 SDFM；

顺逆时针计数

```
#include "spc2188.h"

#include <stdio.h>

int main(void)
{
    CLOCK_InitWithRCO(240000000U);
    Delay_Init();

    PIN_SetChannel(PIN_GPIO62, PIN_GPIO62_UART0_TXD);
    PIN_SetChannel(PIN_GPIO63, PIN_GPIO63_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("EQEP test.....\n");

    /* Config EQEP GPIO */
    PIN_SetChannel(PIN_GPIO52, PIN_GPIO52_EQEP0_A);
    PIN_SetChannel(PIN_GPIO53, PIN_GPIO53_EQEP0_B);
    PIN_SetChannel(PIN_GPIO54, PIN_GPIO54_EQEP0_Z);

    /* Init EQEP */
    EQEP_Init(EQEP0, EQEP_MODE_CW_CCW);

    /* Set IO invert input or normall input */
    EQEP_SetQa(EQEP0, EQEP_INPUT_ORIGINAL ) ;
    EQEP_SetQb(EQEP0, EQEP_INPUT_ORIGINAL ) ;
    EQEP_SetQz(EQEP0, EQEP_INPUT_ORIGINAL);
```

```
    /* Init position counter */
    EQEP_SetPositionCounterValue(EQEP0, 0 ) ;

    /* Set maximum position */
    EQEP_SetMaxPosition(EQEP0, 0xFFFFFFFF) ;

    /* Set position counter reload value */
    EQEP_SetPositionReloadValue(EQEP0, 0 ) ;

    /* EQEP Set counter reset mode */
    EQEP_SetCounterResetMode(EQEP0, EQEP_RESET_ON_EACH_ZERO_MARKER);

    /* Enable EQEP */
    EQEP_Enable(EQEP0);


    while (1)
    {
        printf("EQEP Count : %d\n", EQEP_GetPositionCounterValue(EQEP0));
    }
}
```