

概述

QEP (Quadrature Encoder Pulse) 是一种处理编码器正交输出信号的功能模块。SPC2168 内建三个 SIO 单元。使用 SIO 单元的可编程性，实现了 SIO_QEPV2 的功能，能够进行计数、测速等功能。

目录

1	SIO_QEPV2 特性	7
2	SIO_QEPV2 系统概述	9
2.1	正交解码和位置计数.....	9
2.2	测周法测速和边沿事件捕获计时器.....	11
2.3	测频法测速和测速单元定时器.....	12
2.4	看门狗定时器	12
2.5	模块启动和停止.....	13
2.6	中断事件	13
2.7	计数器值和锁存器值的读取.....	13
3	操作方式	14
3.1	配置 SIO 时钟	14
3.2	配置 SIO 为 SIO_QEPV2，配置 PINMUX	14
3.3	位置初始化	15
3.4	获取位置计数，方向.....	16
3.5	测周法测速	16
3.6	测频法测速	17
3.7	中断	17
4	API 函数	19
5	代码示例	21
6	寄存器	22
6.1	SIO_QEPV2 寄存器表	22

图片列表

图 2-1: 基于 SIO 的 QEP 系统架构	9
图 2-2: 位置计数 (以 POSMAX=9 为例)	10
图 2-3: 测周法测速 ($V = X / dt$, 以 X=5 举例)	11
图 2-4: 测频法测速 ($V = dx / T$) (MAX 为 0xFFFFFFFF)	12

SPIN
TROL

表格列表

表 1-1: 管脚分配.....	8
表 4-1: API 函数列表	19
表 6-1: SIO 模块基地址	22

SPIN TROL

版本历史

版本	日期	作者	状态	变更
A/0	2023-12-05	韩伟	Released	设计 V2.10.2: 更换 PIN: QA : SIO0_PIN16 -> GPIO16 QB : SIO0_PIN0 -> GPIO18 QZ : SIO0_PIN17 -> GPIO17

SPIN
TROL

术语或缩写

术语或缩写	描述
SIO	Smart Input Output
QEP	Quadrature Encoder Pulse

SPIN TROL

1 SIO_QEPV2 特性

SIO_QEPV2 支持以下功能:

- 支持 Qa、Qb 正交输入 (Quadrature);
- 支持 Qa、Qb 最快 250KHz (即 Qclk 频率最快为 1MHz);
- 支持 Qz 信号, 且极性可配;
- 支持 32 位的位置计数器 (POSCNT), 计数范围[0: POSMAX];
- 支持软件初始化 POSCNT;
- 支持 Qz 零位事件 (Zevent) 复位 POSCNT, 根据计数方向 (Qdir):
 - Qdir == 1: 检测到 Zevent, POSCNT 锁存到 POSLAT, 并且复位为 0;
 - Qdir == 0: 检测到 Zevent, POSCNT 锁存到 POSLAT, 并且复位为 POSMAX;
- 支持测周法测速 ($v = x / dt$):
 - 正交计数脉冲 (Qclk) 预分频计数器 (QPS) 计数溢出, 产生 Qevent 事件, 并且将 CaptureTimer 计数值锁存到 CTMRLAT;
- 支持测频法测速 ($v = dx / t$):
 - UnitTimer 计时超时, 将速度计数器 (SCNT) 计数值锁存到 SCNTLAT, 然后 SCNT 复位;
 - Qdir == 1: SCNT 向上计数, 计数到 0xFFFFFFFF 溢出或者 UnitTimer 计时超时, 复位为 0x0;
 - Qdir == 0: SCNT 向下计数, 计数到 0x0 溢出或者 UnitTimer 计时超时, 复位为 0xFFFFFFFF;
- 支持启动和停止;
- 支持中断:
 - 计数方向 (Qdir) 改变事件;
 - Qclk 分频事件 (Qevent):
 - 零位事件 (Zevent):
 - POSCNT 上溢和下溢事件:
 - SCNT 上溢和下溢事件:
 - CaptureTimer 超时事件 (CaptureTimer 向上计数, CaptureTimer >= 0x80000000);
 - WatchdogTimer 看门狗超时事件 (WatchdogTimer 向下计数, WDT == 0x0);
 - UnitTimer 超时事件 (UnitTimer 向下计数, UnitTimer == 0);
 - Qa、Qb 信号边沿相位错误:
 - Qa 和 Qb 同时发生边沿事件;
 - 位置匹配事件 (POSCNT == POSCMP);

表 1-1: 管脚分配

SIO 模块名	SIO 管脚编号	GPIO 管脚编号	功能
SIO0	16	GPIO16	QA
SIO0	0	GPIO18	QB
SIO0	17	GPIO17	QZ

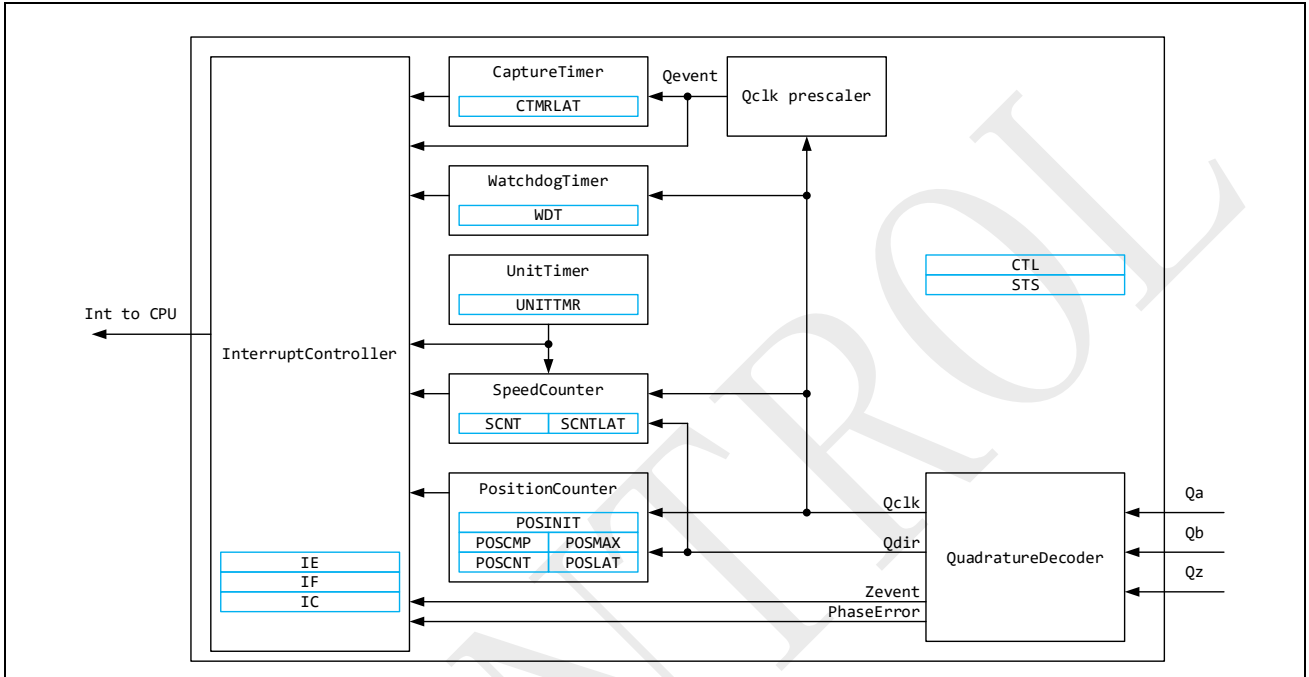
注意事项:

- SIO 时钟频率 $F_{SIO} \leq 100\text{MHz}$;
- 不支持通过修改 SIO 配置文件里的 SIO 管脚数组来实现管脚的重新分配。需要不同的配置文件，请联系 Spintrol 工程师。

2 SIO_QEPV2 系统概述

如图 2-1 所示，SIO_QEPV2 模块由正交解码、位置计数器（PositionCounter）、速度计数器（SpeedCounter）、边沿事件捕获计时器（CaptureTimer）、测速单元定时器（UnitTimer）、边沿事件看门狗定时器（WatchdogTimer）、中断控制等组成。

图 2-1: 基于 SIO 的 QEP 系统架构



2.1 正交解码和位置计数

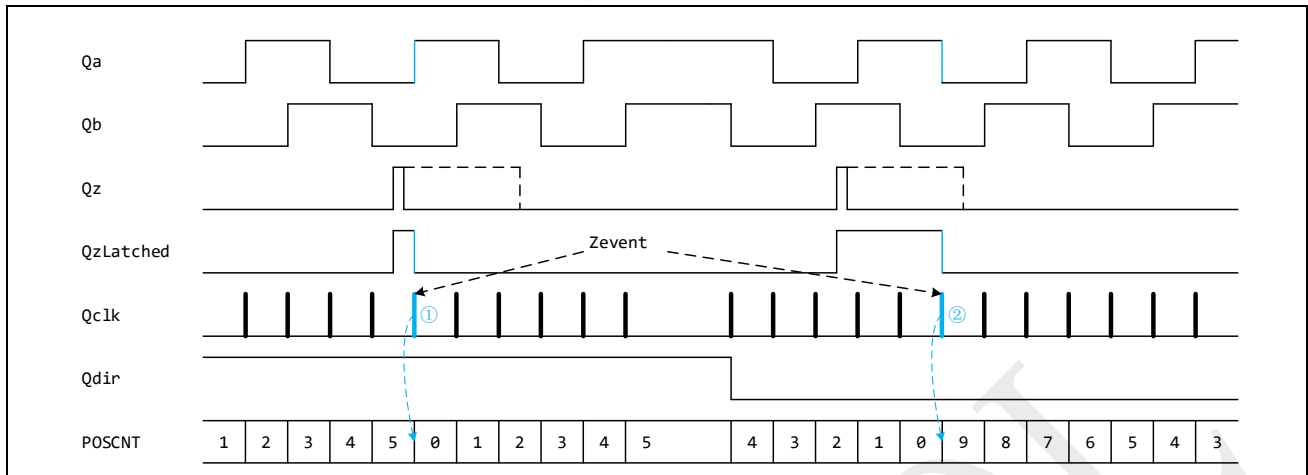
正交解码模块输入为编码器 Qa、Qb、Qz 信号。Qa、Qb 为正交模式输入信号，Qz 为零位（ZeroIndex）信号。不支持输入信号滤波功能。解码模块输出为：

- Qdir: 解码后的计数方向信号；
- Qclk: Qa、Qb 的有效计数脉冲信号；
- Zevent: 零位事件；

位置计数器 POSCNT 主要根据 Qdir 和 Qclk 进行计数，计数范围为[0:POSMAX]，并且具有位置比较功能。当 Zevent 发生时候，会将 POSCNT 锁存到位置锁存寄存器 POSLAT，并且根据 Qdir 复位 POSCNT。支持软件初始化 POSCNT 值为 POSINIT 值。

关于位置计数器 POSCNT 和 Zevent 的功能，如图 2-2 所示。

图 2-2: 位置计数 (以 POSMAX=9 为例)



Qa、Qb 的每个边沿都会产生 Qclk，触发 SCNT 根据当前方向 Qdir 进行计数。计数方向 Qdir 根据 Qa、Qb 的相位关系定义为：

- Qa 超前 Qb: Qdir 为 1, POSCNT 向上计数。当 POSCNT==POSMAX, Qclk 触发时候, 计数向上溢出, 且 POSCNT 复位为 0;
- Qa 滞后 Qb: Qdir 为 0, POSCNT 向下计数。当 POSCNT==0, Qclk 触发时候, 计数向下溢出, 且 POSCNT 复位为 POSMAX;

定义第一次 Zevent 发生时候, 锁存的方向和 Qa、Qb 为 FirstIndexLatchedQdirection、FirstIndexLatchedQA、FirstIndexLatchedQB, 该事件的标志为 FirstIndexLatchedFlag。则零位事件 Zevent 的定义为:

- 检测 Qz 信号的上升沿事件, QzLatched 信号置 1;
- 启动后, Qclk 第一次检测到 QzLatched 为高, FirstIndexLatchedFlag 设置为 1, 且记录当前 Qa、Qb 的边沿信息 FirstIndexLatchedQA、FirstIndexLatchedQB 和当前方向 FirstIndexLatchedQdirection;
- 当 Qdir == FirstIndexLatchedQdirection, Qclk 检测到 QzLatched 为高, 且 Qa、Qb 边沿信息和第一次记录的信息 FirstIndexLatchedQA、FirstIndexLatchedQB 相同, 当前 Qclk 为 Zevent 事件, 如图 2-2 中标记①所示;
- 当 Qdir != FirstIndexLatchedQdirection, Qclk 检测到 QzLatched 为高, 且 Qa、Qb 边沿信息和第一次记录的信息 FirstIndexLatchedQA、FirstIndexLatchedQB 相对 (例如, 第一次记录时, Qa 为上升沿, 则反转相对位置为 Qa 下降沿, 且此时 Qb 电平和 FirstIndexLatchedQB 相同), 当前 Qclk 为 Zevent 事件, 如图 2-2 中标记②所示;
- Zevent 触发后, QzLatched 信号清 0;

由于 Zevent 导致的 POSCNT 复位:

- Qdir 为 1: 检测到 Zevent, POSCNT 锁存到 POSLAT, 并且复位为 0;
- Qdir 为 0: 检测到 Zevent, POSCNT 锁存到 POSLAT, 并且复位为 POSMAX;

注意：

正常情况下，POSLAT 的值应当等于 0 或者 POSMAX；否则的话，可能由于 Qa 或者 Qb 信号有毛刺导致计数错误。

Qz 为高期间，Qdir 不断变化的情况下，最多只有 1 次 Zevent 发生。

2.2 测周法测速和边沿事件捕获计时器

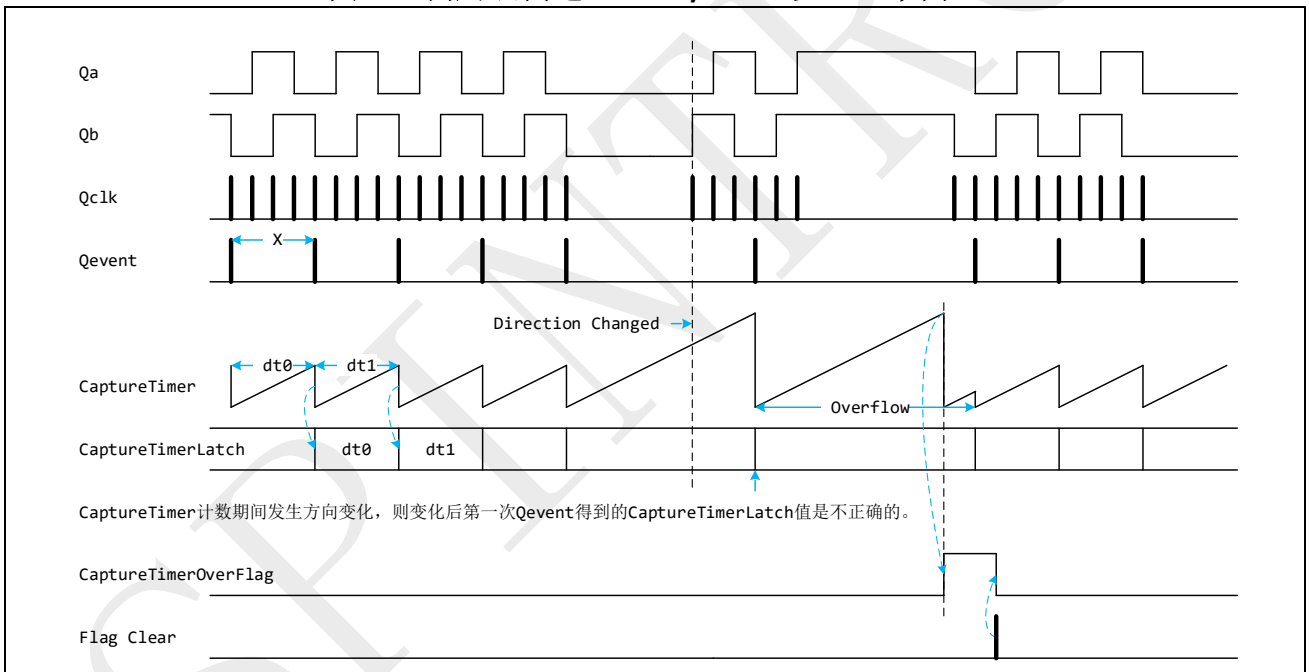
当被测信号频率较低时候，考虑到计数器的计数误差，一般使用测周法来进行频率测量，其公式为：

$$V = \frac{x}{dt} \tag{1}$$

V 表示速度；x 表示固定距离；dt 表示走过固定距离所需要的时间。

如图 2-3 所示。Qclk 经过预分频器产生 Qevent 事件，预分频系数为 x。每当 Qevent 发生时候，将 CaptureTimer 当前的计数值锁存到 CaptureTimerLatch (dt)，然后复位重新计数。因此速度值可以由上述公式计算出。

图 2-3:测周法测速 (V = x / dt, 以 X=5 举例)



注意：

要得到正确的速度值，要确保 CaptureTimer 没有发生计数溢出 (CaptureTimer >= 0x80000000)，且计数方向未发生改变。

2.3 测频法测速和测速单元定时器

当被测信号频率较高时候，一般使用测频法来进行频率测量，其公式为：

$$V = \frac{dx}{t} \quad (2)$$

V 表示速度；dx 表示时间 t 内走过的距离；t 表示设置的时间。

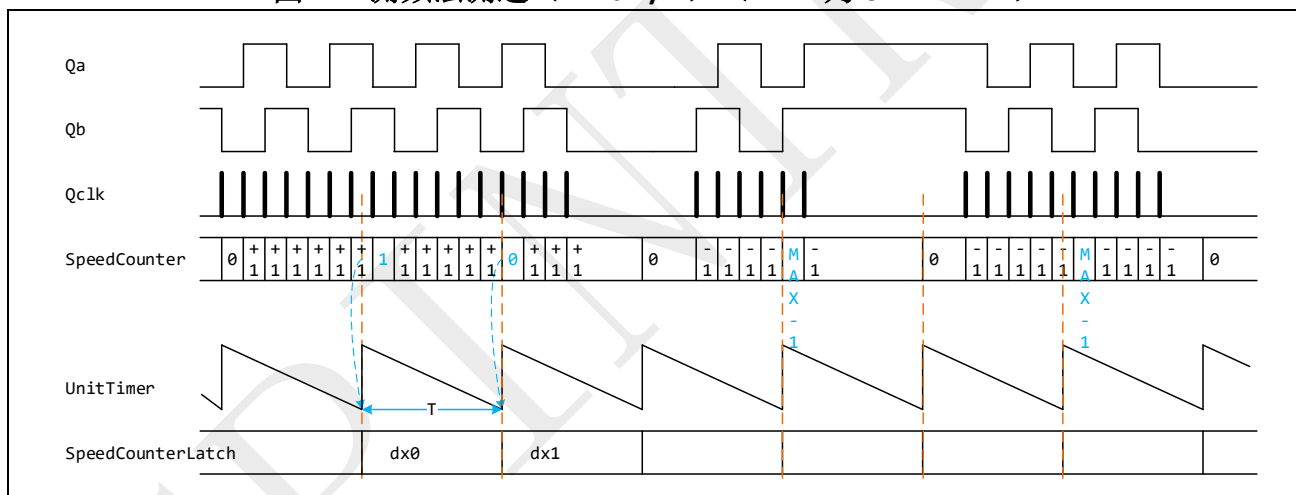
如图 2-4 所示。SpeedCounter 根据 Qdir 进行计数：

- Qdir 为 1：向上计数，计数到 0xFFFFFFFF，复位为 0 重新计数；
- Qdir 为 0：向下计数，计数到 0，复位为 0xFFFFFFFF 重新计数。

设置定时时间为 t，每当定时器 UnitTimer 计时超时的时候，将 SpeedCounter 当前的计数值锁存到 SpeedCounterLatch (dx)，然后根据方向复位重新计数。因此速度值可以由上述公式计算出。其中 SpeedCounter 复位为：

- Qdir 为 1：当 UnitTimer 计时超时和 Qclk 同时发生，则 SpeedCounter 复位为 1；
- Qdir 为 0：当 UnitTimer 计时超时和 Qclk 同时发生，则 SpeedCounter 复位为 0xFFFFFFFF；
- 当 UnitTimer 计时超时和 Qclk 未同时发生，SpeedCounter 复位为 0；

图 2-4: 测频法测速 ($V = dx / T$) (MAX 为 0xFFFFFFFF)



注意：

- 要得到正确的速度值，要确保 SpeedCounter 没有发生计数溢出，且计数方向未发生改变。
- 当 SpeedCounter 计数值的绝对值在 [0:0x7FFFFFFF] 范围内，可以把最高位当作符号位，从而获取带符号的速度值。

2.4 看门狗定时器

Qclk 事件会复位 WatchdogTimer 的计数器，因此当 WatchdogTimer 超时，表示 Qa、Qb 的信号输入电平一直没有变化。

2.5 模块启动和停止

启动时候复位所有状态寄存器和所有功能。

停止后，模块停止工作，当前寄存器值保持不变。

2.6 中断事件

- 计数方向 (Qdir) 改变事件
- Qclk 分频事件 (Qevent)
- 零位事件 (Zevent)
- POSCNT 上溢和下溢事件
 - Qdir == 1: Qclk 发生时候, POSCNT == POSMAX, POSCNT 向上溢出, 复位为 0;
 - Qdir == 0: Qclk 发生时候, POSCNT == 0, POSCNT 向下溢出, 复位为 POSMAX。
- SCNT 上溢和下溢事件
 - Qdir == 1: Qclk 发生时候, SCNT == 0xFFFFFFFF, SCNT 向上溢出, 复位为 1;
 - Qdir == 0: Qclk 发生时候, SCNT == 0, SCNT 向下溢出, 复位为 0xFFFFFFFF。
- CaptureTimer 超时事件
 - CaptureTimer 向上计数, CaptureTimer >= 0x80000000 时候发生超时。
- WatchdogTimer 看门狗超时事件
 - WatchdogTimer 向下计数, WDT == 0x0 时候发生超时。
- UnitTimer 超时事件
 - UnitTimer 向下计数, UnitTimer == 0 时候发生超时。
- Qa、Qb 信号边沿相位错误
 - Qa 和 Qb 同时发生边沿事件。
- 位置匹配事件
 - POSCNT == POSCMP。

2.7 计数器值和锁存器值的读取

SpeedCounter (SCNT)、SpeedCounterLatch (SCNTLAT)、CaptureTimerLatch (CTMRLAT)、PositionLatch (POSLAT) 等 32 位寄存器必须使用如下方式才能获取正确值:

- 必须在相应锁存事件发生后读取锁存的数据;
- 读取数据突变后再读取一次。
 - 32 位寄存器是由两个 16 位寄存器拼接而成的, SIO 逻辑先更新低 16 位, 后更新高 16 位。当低 16 位溢出时候, 会有 1 个时钟周期的值错误, 此时低 16 位突变为 0 (Qdir=0) 或者 0xFFFF (Qdir=1), 且高 16 位保持之前的值不变。

3 SIO_QEPV2 操作方式

Spintrol 提供了相应的软件库来简化该系统的使用。

3.1 配置 SIO 时钟

用户可以通过 SIOCLKCTL 寄存器来配置 SIO 时钟，包括时钟的使能和分频比。具体可参见《SPC2168 Technical Reference Manual》的第 3 章。当 SIO 被配置用作 SIO_QEPV2 时，所允许的 SIO 模块时钟最高频率可达 100MHz（对于部分型号芯片，最高频率会达不到 100MHz，具体参考 TRM）。

3.2 配置 SIO 为 SIO_QEPV2，配置 PINMUX

提供了 SIO_QEPV2_Program() 函数，配置 SIO 模块，将其初始化成 SIO_QEPV2；SIO_QEPV2_Init() 使能 SIO 运行且配置 PINMUX，将引脚切换至 SIO 通道。用户只需要在代码中直接调用这些函数即可。

注意：

- SIO_QEPV2 的引脚重定义需要联系 Spintrol 工程师进行重新配置，目前暂不支持客户自定义。
- 当使用多个 SIO 模块，需要先调用所有的 SIO_XXX_Program() 函数，确保所有 SIO 模块的 uCode 和 PLA 配置完成后，再调用所有的 SIO_XXX_Init() 函数，使能各个模块及配置 IO 管脚。

SIO_QEPV2 初始化代码如下：

示例代码 3-1: SIO_QEPV2 初始化

```
/* Config Flash Timing for 200 MHz */
FLASH_WALLOW();
FLASH_SetTiming(200000000);
FLASH_WDIS();

/* Clock Init */
CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

/* Configure SIO Clock, maximum clock is 100MHz */
CLOCK_SetModuleDiv(SIO0_MODULE, 2);
CLOCK_EnableModule(SIO0_MODULE);

/* SIO program to SIO_QEPV2 function */
SIO_QEPV2_Program(SIOx);

/* SIO_QEPV2 Init */
SIO_QEPV2_Init(SIOx);
```

示例代码 3-1: SIO_QEPV2 初始化

```
/* Parameters setting */
SIO_QEPV2_SetQzPolarity( SIOx, eQzPol );
SIO_QEPV2_SetComparePosition( SIOx, u32CmpPosition );
SIO_QEPV2_SetMaximumPosition( SIOx, u32MaxPosition );
SIO_QEPV2_SetInitialPosition( SIOx, u32InitPosition );
SIO_QEPV2_EnableInitPosition( SIOx );
SIO_QEPV2_SetQclkPrescaler( SIOx, u16QclkPrescaler );
SIO_QEPV2_SetUnitTimerPeriod( SIOx, u16UnitTimerCntLow, u16UnitTimerCntHigh );
SIO_QEPV2_SetWatchdogTimerPeriod( SIOx, u16WatchdogTimerCntLow,
u16WatchdogTimerCntHigh );

/* Enable interrupt */
SIO_QEPV2_EnableInt( SIOx, u16IntSel );
NVIC_EnableIRQ(SIO0A_IRQn);

/* Enable unit tmier */
SIO_QEPV2_EnableUnitTimer( SIOx );

/* Enable run */
SIO_QEPV2_Enable( SIOx );
```

3.3 位置初始化**示例代码 3-2: SIO_QEPV2 位置初始化**

```
SIO_QEPV2_SetInitialPosition( SIOx, u32InitPosition );
SIO_QEPV2_EnableInitPosition( SIOx );
```

3.4 获取位置计数，方向

示例代码 3-3: SIO_QEPV2 读取位置计数器和计数方向

```
/* Get position and direction */
u32Data = SIO_QEPV2_GetPosition(SIOx) ;
u16Dir  = SIO_QEPV2_GetStatus(SIOx, SIOQEPV2_STS_DIRECTION) ? 1 : 0 ;
PRINT_INFO("D: %d; Pos: %u", u16Dir, u32Data );
/* Get speed counter */
i32SpeedCnt = SIO_QEPV2_GetSpeedCounter(SIOx) ;
PRINT_INFO("SCNT: %d", i32SpeedCnt );
```

3.5 测周法测速

示例代码 3-4: SIO_QEPV2 测周法测速

```
/* Get speed by suvery cycle */
u16QclkPrescaler      = SIO_QEPV2_GetQclkPrescaler( SIOx );
u32LatchedCaptureTimer = SIO_QEPV2_GetCaptureTimerLatchValue( SIOx );
/* */
f64Speed0      = 1.0e9 * u16QclkPrescaler / ( u32LatchedCaptureTimer * ( 1.0e9 /
f64Clock ) ) ;
f64Speed0_rpm = f64Speed0 / 4 * 60 / ( u32MaxPosition + 1 ) ;
PRINT_INFO("CSpeed is %f Hz ( %f rpm )", f64Speed0, f64Speed0_rpm );
```


3.6 测频法测速

示例代码 3-5: SIO_QEPV2 测频法测速

```
/* Get speed by suvery frequency */
u32UnitTimerCnt      = SIO_QEPV2_GetUnitTimerPeriod( SIOx );
u32UnitTimerPeriod  = u32UnitTimerCnt * ( 1.0e9 / f64Clock );
i32LatchedSpeedCounter  = SIO_QEPV2_GetSpeedCounterLatchValue( SIOx );
/* */
f64Speed1 = 1.0e9 * i32LatchedSpeedCounter / u32 UnitTimerPeriod ;
f64Speed1_rpm = f64Speed1 / 4 * 60 / ( u32MaxPosition + 1 ) ;
PRINT_INFO("FSpeed is %f Hz ( %f rpm )", f64Speed1, f64Speed1_rpm );
```

3.7 中断

示例代码 3-6: SIO_QEPV2 读取位置计数器和计数方向

```
uint16_t u16Flag ;

uint16_t u16IntSel = 0xFFFF;

u16Flag = SIO_QEPV2_GetIntFlag(SIOx, u16IntSel);
SIO_QEPV2_ClearInt(SIOx, u16Flag);

if ( u16Flag & SIOQEPV2_INT_DIRECTION_CHANGED )
{
    SIO_QEPV2_DirectionChangedIntCallback(SIOx) ;
}

if ( u16Flag & SIOQEPV2_INT_PHASE_ERROR )
{
    SIO_QEPV2_PhaseErrorIntCallback(SIOx) ;
}

if ( u16Flag & SIOQEPV2_INT_POSITION_MATCH )
{
    SIO_QEPV2_PositionCompareMatchIntCallback(SIOx) ;
}

if ( u16Flag & SIOQEPV2_INT_QEVENT )
{
    SIO_QEPV2_QeventIntCallback(SIOx) ;
}

if ( u16Flag & SIOQEPV2_INT_ZEVENT )
```

示例代码 3-6: SIO_QEPV2 读取位置计数器和计数方向

```
{
    SIO_QEPV2_ZeventIntCallback(SIOx) ;
}

if ( u16Flag & SIOQEPV2_INT_UNIT_TIMER_TIMEOUT      )
{
    SIO_QEPV2_UnitTimerTimeoutIntCallback(SIOx) ;
}

if ( u16Flag & SIOQEPV2_INT_CAPTURE_TIMER_TIMEOUT   )
{
    SIO_QEPV2_CaptureTimerTimeoutIntCallback(SIOx) ;
}

if ( u16Flag & SIOQEPV2_INT_WATCHDOG_TIMER_TIMEOUT )
{
    SIO_QEPV2_WatchdogTimerTimeoutIntCallback(SIOx) ;
}

if ( u16Flag & SIOQEPV2_INT_POSITION_COUNTER_UNDERFLOW )
{
    SIO_QEPV2_PositionCounterUnderflowIntCallback(SIOx) ;
}

if ( u16Flag & SIOQEPV2_INT_POSITION_COUNTER_OVERFLOW )
{
    SIO_QEPV2_PositionCounterOverflowIntCallback(SIOx) ;
}

if ( u16Flag & SIOQEPV2_INT_SPEED_COUNTER_UNDERFLOW   )
{
    SIO_QEPV2_SpeedCounterUnderflowIntCallback(SIOx) ;
}

if ( u16Flag & SIOQEPV2_INT_SPEED_COUNTER_OVERFLOW   )
{
    SIO_QEPV2_SpeedCounterOverflowIntCallback(SIOx) ;
}
```

4 API 函数

注意：函数详细功能请参考驱动代码注释。

表 4-1: API 函数列表

函数名称	说明
<code>void SIO_QEPV2_Program(SIOx)</code>	SIO 编程为 SIO_QEPV2 功能
<code>void SIO_QEPV2_Init(SIOx)</code>	SIO_QEPV2 初始化
<code>void SIO_QEPV2_Enable(SIOx)</code>	使能 SIO_QEPV2 运行
<code>void SIO_QEPV2_Disable(SIOx)</code>	停止运行
<code>uint16_t SIO_QEPV2_IsEnable(SIOx)</code>	读取判断是否正在运行
<code>void SIO_QEPV2_EnableInitPosition(SIOx)</code>	使能位置初始化功能
<code>void SIO_QEPV2_DisableInitPosition(SIOx)</code>	停止位置初始化功能
<code>uint16_t SIO_QEPV2_IsEnableInitPosition(SIOx)</code>	读取判断是否使能位置初始化
<code>void SIO_QEPV2_EnableUnitTimer(SIOx)</code>	使能 UnitTimer
<code>void SIO_QEPV2_DisableUnitTimer(SIOx)</code>	停止运行 UnitTimer
<code>uint16_t SIO_QEPV2_IsEnableUnitTimer(SIOx)</code>	读取判断是否使能 UnitTimer
<code>SIO_QEPV2_SetQzPolarity(SIOx, ePol)</code>	设置 Qz 极性
<code>SIOQEPV2_QzPolarityEnum SIO_QEPV2_GetQzPolarity(SIOx)</code>	读取 Qz 极性设置
<code>void SIO_QEPV2_SetInitialPosition(SIOx, u32Pos)</code>	设置初始位置
<code>uint32_t SIO_QEPV2_GetInitialPosition(SIOx)</code>	读取初始位置
<code>void SIO_QEPV2_SetQclkPrescaler(SIOx, u16Prescaler)</code>	设置 Qclk 预分频系数
<code>uint16_t SIO_QEPV2_GetQclkPrescaler(SIOx)</code>	读取 Qclk 预分频系数
<code>void SIO_QEPV2_SetComparePosition(SIOx, u32Pos)</code>	设置比较位置
<code>uint32_t SIO_QEPV2_GetComparePosition(SIOx)</code>	读取比较位置
<code>void SIO_QEPV2_SetMaximumPosition(SIOx, u32Pos)</code>	设置最大位置
<code>uint32_t SIO_QEPV2_GetMaximumPosition(SIOx)</code>	读取最大位置
<code>uint32_t SIO_QEPV2_GetPosition(SIOx)</code>	读取当前位置
<code>void SIO_QEPV2_SetUnitTimerPeriod(SIOx, u16PrdLow, u16PrdHigh)</code>	设置 UnitTimer 周期。周期为： $(PrdLow+1) * (PrdHigh+1) * SIO_CLK_Period$ 。其中 PrdLow 必须大于 0xFF。
<code>uint32_t SIO_QEPV2_GetUnitTimerPeriod(SIOx)</code>	获取 UnitTimer 周期
<code>uint32_t SIO_QEPV2_GetPositionLatchValue(SIOx)</code>	获取 Zevent 触发时候锁存的位置值
<code>uint32_t SIO_QEPV2_GetCaptureTimerLatchValue(SIOx)</code>	获取 Qevent 出发时候锁存的 CaptureTimer 计数值
<code>void SIO_QEPV2_SetWatchdogTimerPeriod(SIOx, u16PrdLow, u16PrdHigh)</code>	设置 WDT 周期。周期为： $(PrdLow+1) * (PrdHigh+1) * SIO_CLK_Period$ 。其中 PrdLow 必须大于 0xFFF。
<code>uint32_t SIO_QEPV2_GetWatchdogTimerPeriod(SIOx)</code>	获取 WDT 周期
<code>uint32_t SIO_QEPV2_GetSpeedCounter(SIOx)</code>	获取用于测速的计数器值

函数名称	说明
<code>uint32_t SIO_QEPV2_GetSpeedCounterLatchValue(SIOx)</code>	获取 UnitTimer 超时时候锁存的测速计数器值
<code>void SIO_QEPV2_EnableInt(SIOx, u16IntSel)</code>	使能中断
<code>void SIO_QEPV2_DisableInt(SIOx, u16IntSel)</code>	停止中断
<code>uint16_t SIO_QEPV2_IsEnableInt(SIOx, u16IntSel)</code>	读取判断是否使能中断
<code>uint16_t SIO_QEPV2_GetIntFlag(SIOx, u16IntSel)</code>	获取中断标志
<code>void SIO_QEPV2_ClearInt(SIOx, u16IntSel)</code>	清除中断标志
<code>uint16_t SIO_QEPV2_IsClearInt(SIOx)</code>	判断中断清除是否结束
<code>uint16_t SIO_QEPV2_GetStatus(SIOx, u16Query)</code>	获取状态
<code>void SIO_QEPV2_IRQHandler(SIOx)</code>	中断服务程序
<code>void SIO_QEPV2_DirectionChangedIntCallback(SIOx)</code>	中断响应事件回调函数
<code>void SIO_QEPV2_PhaseErrorIntCallback(SIOx)</code>	中断响应事件回调函数
<code>void SIO_QEPV2_PositionCompareMatchIntCallback(SIOx)</code>	中断响应事件回调函数
<code>void SIO_QEPV2_QeventIntCallback(SIOx)</code>	中断响应事件回调函数
<code>void SIO_QEPV2_ZeventIntCallback(SIOx)</code>	中断响应事件回调函数
<code>void SIO_QEPV2_UnitTimerTimeoutIntCallback(SIOx)</code>	中断响应事件回调函数
<code>void SIO_QEPV2_CaptureTimerTimeoutIntCallback(SIOx)</code>	中断响应事件回调函数
<code>void SIO_QEPV2_WatchdogTimerTimeoutIntCallback(SIOx)</code>	中断响应事件回调函数
<code>void SIO_QEPV2_PositionCounterUnderflowIntCallback(SIOx)</code>	中断响应事件回调函数
<code>void SIO_QEPV2_PositionCounterOverflowIntCallback(SIOx)</code>	中断响应事件回调函数
<code>void SIO_QEPV2_SpeedCounterUnderflowIntCallback(SIOx)</code>	中断响应事件回调函数
<code>void SIO_QEPV2_SpeedCounterOverflowIntCallback(SIOx)</code>	中断响应事件回调函数

5 代码示例

以 SIO0 为例子，参考 demos 目录下的例程。

SPIN TROL

6 寄存器

6.1 SIO_QEPV2 寄存器表

表 6-1: SIO 模块基地址

外设模块	基地址
SIO0	0x4000B000
SIO1	0x4000C000
SIO2	0x4000D000

注意 对 SPC2168, SIO 的 FIFO 深度为 8, 宽度为 16-bits。

SIO_QEPV2 的寄存器和功能定义请参考相应驱动函数中的描述, 此处不再赘述。