Forum: 16-bit PIC24/dsPIC

Topic: 關於dsPIC33EP256MC506使用MCC產出的程式問題

Subject: 關於dsPIC33EP256MC506使用MCC產出的程式問題

作者: DengKai

2018年04月02日 16:08:42

今天要使用dsPIC33EP256MC506的PWM弁鉈承A利用MCC產出程式後發現在pwm.h中所提供的一個弁鄖蝳’鹵欞~。

這個弁鄖蝳＇0要設定IOCONx暫存器中的OVRDAT<1:0>的值，紅色的部分應該要改成overrideData & 0x0003，分享給大家~

```
/**
  @Summary
    Updates PWM override data bits with the requested value for a specific instance.

  @Description
    This routine is used to updates PWM override data bits with the requested value for a
specific instance.

  @Param
    genNum          - PWM generator instance number.
    overrideData    - Override data

  @Returns
    None

  @Example
    <code>
    PWM_GENERATOR genNum;
    uint16_t overrideData;

    overrideData = 0x01;

    genNum = PWM_GENERATOR_1;
    PWM_OverrideDataSet(genNum, overrideData);
    </code>
*/
inline static void PWM_OverrideDataSet(PWM_GENERATOR genNum,uint16_t overrideData)
{
    switch(genNum) {
        case PWM_GENERATOR_1:
                overrideData = ((overrideData & 0xFFFC)<<6);
                __builtin_write_PWMSFR(&IOCON1, (IOCON1 | overrideData), &PWMKEY);

                break;
        case PWM_GENERATOR_2:
                overrideData = ((overrideData & 0xFFFC)<<6);
```

```
                __builtin_write_PWMSFR(&IOCON2, (IOCON2 | overrideData), &PWMKEY);

                break;
        case PWM_GENERATOR_3:
                overrideData = ((overrideData & 0xFFFC)<<6);
                __builtin_write_PWMSFR(&IOCON3, (IOCON3 | overrideData), &PWMKEY);

                break;
        default:break;
    }
}
```

附加檔案:

---

register.jpg(209.43 KB)

寄存器 14-13： IOCONx：PWM I/O 控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PENH	PENL	POLH	POLL	PMOD<1:0>		OVRENH	OVRENL
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OVRDAT<1:0>		FLTDAT<1:0>(1,2)		CLDAT<1:0>		SWAP	OSYNC
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15 **PENH：** PWMxH 输出引脚所有权位
1 = PWM 模块控制 PWMxH 引脚
0 = GPIO 模块控制 PWMxH 引脚

bit 14 **PENL：** PWMxL 输出引脚所有权位
1 = PWM 模块控制 PWMxL 引脚
0 = GPIO 模块控制 PWMxL 引脚

bit 13 **POLH：** PWMxH 输出引脚极性位
1 = PWMxH 引脚为低电平有效
0 = PWMxH 引脚为高电平有效

bit 12 **POLL：** PWMxL 输出引脚极性位
1 = PWMxL 引脚为低电平有效
0 = PWMxL 引脚为高电平有效

bit 11-10 **PMOD<1:0>：** PWM # I/O 引脚模式位
11 = PWM I/O 引脚对处于真正独立 PWM 输出模式 (3)
10 = PWM I/O 引脚对处于推挽输出模式
01 = PWM I/O 引脚对处于冗余输出模式
00 = PWM I/O 引脚对处于互补输出模式

bit 9 **OVRENH：** PWMxH 引脚改写使能位
1 = OVRDAT<1> 为 PWMxH 引脚上的输出提供数据
0 = PWM 发生器为 PWMxH 引脚提供数据

bit 8 **OVRENL：** PWMxL 引脚改写使能位
1 = OVRDAT<0> 为 PWMxL 引脚上的输出提供数据
0 = PWM 发生器为 PWMxL 引脚提供数据

bit 7-6 **OVRDAT<1:0>：** 使能改写时 PWMxH 和 PWMxL 引脚状态 (2) 位
如果 OVERENH = 1，则 OVRDAT<1> 提供 PWMxH 的数据
如果 OVERENL = 1，则 OVRDAT<0> 提供 PWMxL 的数据

code.jpg(92.38 KB)

```
    overrideData    - Override data

  @Returns
    None

  @Example
    <code>
    PWM_GENERATOR genNum;
    uint16_t overrideData;

    overrideData = 0x01;

    genNum = PWM_GENERATOR_1;
    PWM_OverrideDataSet(genNum, overrideData);
    </code>
*/
inline static void PWM_OverrideDataSet(PWM_GENERATOR genNum,uint16_t overrideData)
{
    switch(genNum) {
        case PWM_GENERATOR_1:
                overrideData = ((overrideData & 0xFFFC)<<6);
                __builtin_write_PWMSFR(&IOCON1, (IOCON1 | overrideData), &PWMKEY);
                break;
        case PWM_GENERATOR_2:
                overrideData = ((overrideData & 0xFFFC)<<6);
                __builtin_write_PWMSFR(&IOCON2, (IOCON2 | overrideData), &PWMKEY);
                break;
        case PWM_GENERATOR_3:
                overrideData = ((overrideData & 0xFFFC)<<6);
                __builtin_write_PWMSFR(&IOCON3, (IOCON3 | overrideData), &PWMKEY);
                break;
        default:break;
    }
}
```