FAQ0097

常见问题解答

AT32F421_BLDC电机六步驱动方式设置说明

Questions：AT32F421 BLDC 电机六步驱动方式设置说明？

Answer：

1. TMR1 设置方式如下：

```
/* time base configuration */
tmr_base_init(TMR1, 4095, 0);
tmr_cnt_dir_set(TMR1, TMR_COUNT_UP);

/* channel 1, 2 and 3 configuration in output mode */
tmr_output_default_para_init(&tmr_output_struct);
tmr_output_struct.oc_mode = TMR_OUTPUT_CONTROL_OFF;
tmr_output_struct.oc_output_state = TRUE;
tmr_output_struct.oc_polarity = TMR_OUTPUT_ACTIVE_HIGH;
tmr_output_struct.oc_idle_state = TRUE;
tmr_output_struct.occ_output_state = TRUE;
tmr_output_struct.occ_polarity = TMR_OUTPUT_ACTIVE_HIGH;
tmr_output_struct.occ_idle_state = TRUE;

/* channel 1 */
tmr_output_channel_config(TMR1, TMR_SELECT_CHANNEL_1, &tmr_output_struct);
tmr_channel_value_set(TMR1, TMR_SELECT_CHANNEL_1, 2047);
/* channel 2 */
tmr_output_channel_config(TMR1, TMR_SELECT_CHANNEL_2, &tmr_output_struct);
tmr_channel_value_set(TMR1, TMR_SELECT_CHANNEL_2, 1023);
/* channel 3 */
tmr_output_channel_config(TMR1, TMR_SELECT_CHANNEL_3, &tmr_output_struct);
tmr_channel_value_set(TMR1, TMR_SELECT_CHANNEL_3, 511);

/* automatic output enable, break, dead time and lock configuration */
tmr_brkdt_default_para_init(&tmr_brkdt_config_struct);
tmr_brkdt_config_struct.brk_enable = TRUE;
tmr_brkdt_config_struct.auto_output_enable = TRUE;
tmr_brkdt_config_struct.deadtime = 0;
tmr_brkdt_config_struct.fcsodis_state = TRUE;
tmr_brkdt_config_struct.fcsoen_state = TRUE;
tmr_brkdt_config_struct.brk_polarity = TMR_BRK_INPUT_ACTIVE_HIGH;
```

```
tmr_brkdt_config_struct.wp_level = TMR_WP_OFF;
tmr_brkdt_config(TMR1, &tmr_brkdt_config_struct);
tmr_channel_buffer_enable(TMR1, TRUE);

/* hall interrupt enable */
tmr_interrupt_enable(TMR1, TMR_HALL_INT, TRUE);
/* tmr1 hall interrupt nvic init */
nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);
nvic_irq_enable(TMR1_BRK_OVF_TRG_HALL_IRQn, 0, 0);
/* tmr1 output enable */
tmr_output_enable(TMR1, TRUE);
/* enable tmr1 */
tmr_counter_enable(TMR1, TRUE);
```

2. 互补 PWM 设置说明：

表 14-15 带刹车功能的互补输出通道CxOUT和CxCOUT的控制位

控制位					输出状态（1）	
OEN 位	FCSODIS 位	FCSOEN 位	CxEN 位	CxCEN 位	CxOUT 输出状态	CxCOUT 输出状态
1	X	0	0	0	输出禁止（与定时器断开）CxOUT=0，Cx_EN=0	输出禁止（与定时器断开）CxCOUT=0，CxCEN=0
		0	0	1	输出禁止（与定时器断开）CxOUT=0，Cx_EN=0	CxORAW + 极性，CxCOUT= CxORAW xor CxCP，CxCEN=1
		0	1	0	CxORAW+极性，CxOUT= CxORAW xor CxP，Cx_EN=1	输出禁止（与定时器断开）CxCOUT=0，CxCEN=0
		0	1	1	CxORAW+极性+死区，Cx_EN=1	CxORAW 反相+极性+死区，CxCEN=1
		1	0	0	输出禁止（与定时器断开）CxOUT=CxP，Cx_EN=0	输出禁止（与定时器断开） CxCOUT=CxCP，CxCEN=0
		1	0	1	关闭状态（输出使能且为无效电平）CxOUT=CxP，Cx_EN=1	CxORAW + 极性，CxCOUT= CxORAW xor CxCP，CxCEN=1
		1	1	0	CxORAW + 极性，CxOUT= CxORAW xor CxP，Cx_EN=1	关闭状态（输出使能且为无效电平）CxCOUT=CxCP，CxCEN=1
		1	1	1	CxORAW+极性+死区，Cx_EN=1	CxORAW 反相+极性+死区，CxCEN=1

A：CxEN=0,CxCEN=0 时，TMR1 不接管其所对应的 IO，此时 TMR1 对应的 IO 为 Floating 态，此时对应的 IO 输出电平由外部 IO 上拉或下拉决定；

B：使用六步方式时，需要强制此时状态（CxEN=0,CxCEN=0）对应 IO 为无效电平状态；

如：

```
/*强迫 OC3 的上下桥为无效电平状态*/
tmr_force_output_set(TMR1, TMR_SELECT_CHANNEL_3, TMR_FORCE_OUTPUT_LOW);
/*需要设置其对应的上或下桥其中一个为使能*/
tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_3, FALSE);
tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_3C, TRUE);
```

3. 换向设置方式 Demo 如下：

```
void TMR1_BRK_OVF_TRG_HALL_IRQHandler(void)
{
  /* clear tmr1 com pending bit */
  tmr_flag_clear(TMR1, TMR_HALL_FLAG);
  if(step == 1)
  {
    /* next step: step 2 configuration ---------------------------- */
    /*  channel3 configuration */
    tmr_force_output_set(TMR1, TMR_SELECT_CHANNEL_3, TMR_FORCE_OUTPUT_LOW);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_3, FALSE);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_3C, TRUE);
    /*  channel1 configuration */
    tmr_output_channel_mode_select(TMR1, TMR_SELECT_CHANNEL_1, TMR_OUTPUT_CONTROL_PWM_MODE_A);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_1, TRUE);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_1C, FALSE);
    /*  channel2 configuration */
    tmr_output_channel_mode_select(TMR1, TMR_SELECT_CHANNEL_2, TMR_OUTPUT_CONTROL_PWM_MODE_A);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_2, FALSE);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_2C, TRUE);
    step++;
  }
  else if(step == 2)
  {
    /* next step: step 3 configuration ---------------------------- */
    /*  channel2 configuration */
    tmr_output_channel_mode_select(TMR1, TMR_SELECT_CHANNEL_2, TMR_OUTPUT_CONTROL_PWM_MODE_A);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_2, FALSE);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_2C, TRUE);
    /*  channel3 configuration */
    tmr_output_channel_mode_select(TMR1, TMR_SELECT_CHANNEL_3, TMR_OUTPUT_CONTROL_PWM_MODE_A);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_3, TRUE);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_3C, FALSE);
    /*  channel1 configuration */
    tmr_force_output_set(TMR1, TMR_SELECT_CHANNEL_1, TMR_FORCE_OUTPUT_LOW);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_1, FALSE);
```

```
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_1C, TRUE);
    step++;
  }
  else if(step == 3)
  {
    /* next step: step 4 configuration ---------------------------- */
    /*   channel3 configuration */
    tmr_output_channel_mode_select(TMR1, TMR_SELECT_CHANNEL_3, TMR_OUTPUT_CONTROL_PWM_MODE_A);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_3, TRUE);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_3C, FALSE);
    /*   channel2 configuration */
    tmr_force_output_set(TMR1, TMR_SELECT_CHANNEL_2, TMR_FORCE_OUTPUT_LOW);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_2, FALSE);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_2C, TRUE);
    /*   channel1 configuration */
    tmr_output_channel_mode_select(TMR1, TMR_SELECT_CHANNEL_1, TMR_OUTPUT_CONTROL_PWM_MODE_A);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_1, FALSE);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_1C, TRUE);
    step++;
  }
  else if(step == 4)
  {
    /* next step: step 5 configuration ---------------------------- */
    /*   channel3 configuration */
    tmr_force_output_set(TMR1, TMR_SELECT_CHANNEL_3, TMR_FORCE_OUTPUT_LOW);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_3, FALSE);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_3C, TRUE);
    /*   channel1 configuration */
    tmr_output_channel_mode_select(TMR1, TMR_SELECT_CHANNEL_1, TMR_OUTPUT_CONTROL_PWM_MODE_A);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_1, FALSE);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_1C, TRUE);
    /*   channel2 configuration */
    tmr_output_channel_mode_select(TMR1, TMR_SELECT_CHANNEL_2, TMR_OUTPUT_CONTROL_PWM_MODE_A);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_2, TRUE);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_2C, FALSE);
    step++;
  }
  else if(step == 5)
  {
    /* next step: step 6 configuration ---------------------------- */
    /*   channel3 configuration */
    tmr_output_channel_mode_select(TMR1, TMR_SELECT_CHANNEL_3, TMR_OUTPUT_CONTROL_PWM_MODE_A);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_3, FALSE);
```

```
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_3C, TRUE);
    /*   channel1 configuration */
    tmr_force_output_set(TMR1, TMR_SELECT_CHANNEL_1, TMR_FORCE_OUTPUT_LOW);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_1, FALSE);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_1C, TRUE);
    /*   channel2 configuration */
    tmr_output_channel_mode_select(TMR1, TMR_SELECT_CHANNEL_2, TMR_OUTPUT_CONTROL_PWM_MODE_A);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_2, TRUE);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_2C, FALSE);
    step++;
  }
  else
  {
    /* next step: step 1 configuration ---------------------------- */
    /*   channel1 configuration */
    tmr_output_channel_mode_select(TMR1, TMR_SELECT_CHANNEL_1, TMR_OUTPUT_CONTROL_PWM_MODE_A);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_1, TRUE);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_1C, FALSE);
    /*   channel3 configuration */
    tmr_output_channel_mode_select(TMR1, TMR_SELECT_CHANNEL_3, TMR_OUTPUT_CONTROL_PWM_MODE_A);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_3, FALSE);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_3C, TRUE);
    /*   channel2 configuration */
    tmr_force_output_set(TMR1, TMR_SELECT_CHANNEL_2, TMR_FORCE_OUTPUT_LOW);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_2, FALSE);
    tmr_channel_enable(TMR1, TMR_SELECT_CHANNEL_2C, TRUE);
    step = 1;
  }
}
```

类型：MCU 应用

适用型号：AT32F421

主功能：TMR1

次功能：Motor Control

文档版本历史

日期	版本	变更
2022.2.24	2.0.0	最初版本

重要通知 - 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途（及其依据任何司法管辖区的法律的对应情况），或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：（A）对安全性有特别要求的应用，例如：生命支持、主动植入设备或对产品功能安全有要求的系统；（B）航空应用；（C）航天应用或航天环境；（D）武器，且/或（E）其他可能导致人身伤害、死亡及财产损害的应用。如果采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险及法律责任仍将由采购商单独承担，且采购商应独力负责在前述应用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。