Application Note

基于 MSPM0 的有传感器有刷直流电机控制

Jingguo Wang and Eason Zhou

摘要

本应用手册介绍了基于 MSPM0C1104 的有传感器有刷直流电机控制。在该解决方案中，MSPM0C1104 利用其高级计时器并使用 PI 控制器通过电机的速度和电流信息来控制有刷直流电机。

工程配套资料包含软件工程和硬件设计，可从以下 URL 下载：https://www.ti.com/cn/lit/zip/slaaem1。您也可以在 SDK 下找到该配套资料，其地址为：`C:\ti\mspm0_sdk_x_xx_xx_xx\examples\nortos\LP_MSPM0C1104\demos\motor_control_bdc_sensor`。

内容

1 引言....2
2 系统架构简介....2
3 硬件设计简介....3
3.1 电源电路....3
3.2 驱动电路....4
3.3 采样电路....4
3.4 主控制器电路....5
3.5 控制系统简介....5
4 软件设计简介....7
4.1 参数初始化....7
4.2 方向设置....8
4.3 计时器中断....8
4.4 闭环控制器....9
5 评估....10

插图清单

图 2-1. 系统方框图....2
图 3-1. 硬件电路原理图....3
图 3-2. 电源电路....3
图 3-3. 驱动电路....4
图 3-4. 采样电路....4
图 3-5. 主控制器电路....5
图 3-6. 电机驱动原理....6
图 3-7. 控制方框图....6
图 4-1. 软件流程....7
图 4-2. 参数初始化....7
图 4-3. 主函数....8
图 4-4. 计时器中断....8
图 4-5. 电流 PID....9
图 4-6. 速度 PID....10
图 5-1. 电机控制板....11

表格清单

表 3-1. 硬件连接....5

表 5-1. JGB37-520E 电机参数..... 10

商标

1 引言

有刷直流电机因其简易性、易控制性和成本效益而闻名，是注重性能、成本效益和可靠性的应用的优先选择。MSPM0C1104 属于 MSP 高度集成的超低功耗 32 位微控制器单元 (MCU) 系列，基于增强型 Arm® Cortex®-M0+ 内核平台，工作频率高达 24MHz，可提供先进的电机控制功能，因此非常适合驱动和控制有刷直流电机。本文档介绍了一种由 MSPM0C1104 控制的分立式电机驱动解决方案，该解决方案可以降低系统成本并提高系统设计灵活性，适用于各种工业和商业应用。

2 系统架构简介

图 2-1 显示了电机驱动器解决方案的系统方框图。除 MSPM0C1104 和电源环路之外，系统还主要包含三个组成部分：电源电路、驱动器电路和采样电路。

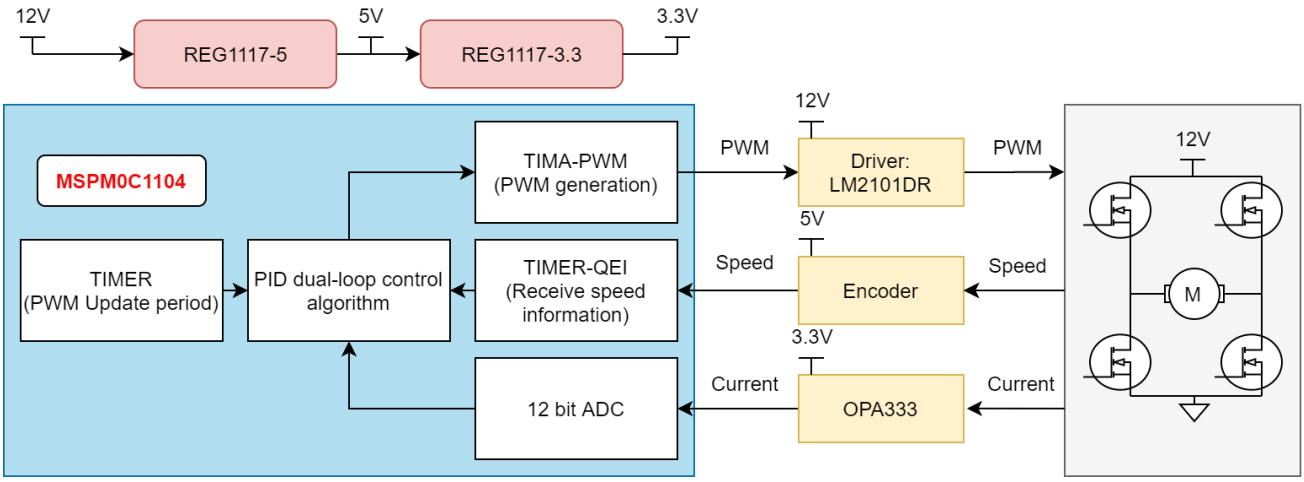

图 2-1. 系统方框图

采样电路通过编码器和运算放大器收集电机的电流和速度信息，然后将其发送到 MCU。MCU 通过 PID 闭环计算占空比，并根据实时速度和电流信息生成 PWM 信号。MOSFET 驱动器电路将 MCU 生成的 PWM 信号转换为驱动信号，以控制 MOSFET 开关。电源电路为系统中的所有元件（例如 MCU、编码器、驱动器、OPA 等）提供适当的电源轨。下面将详细说明硬件电路和部分软件控制代码。

3 硬件设计简介

图 3-1 展示了整个硬件电路的原理图。下一节将对电路的每个部分进行全面的说明。

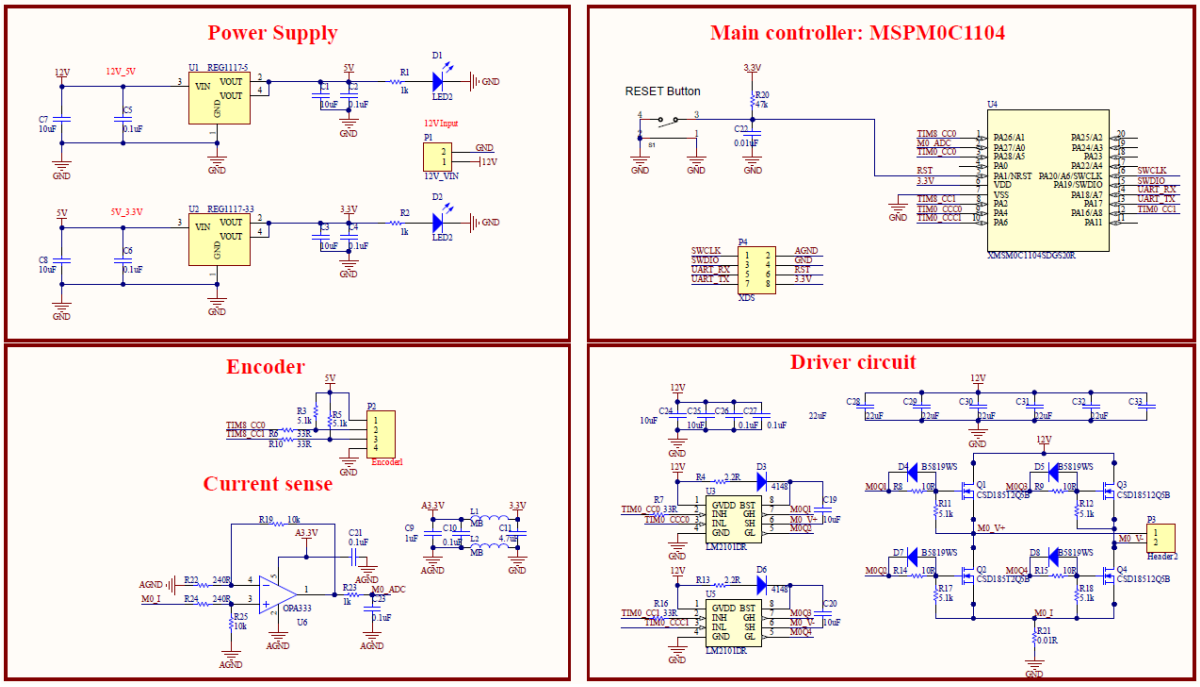

图 3-1. 硬件电路原理图

3.1 电源电路

节 3.1 展示了电源电路，该电路为编码器电路提供 5V 电源轨，为 MSPM0C1104 提供 3.3V 电源轨。该电路采用两个固定输出 LDO，可降低噪声并获得稳定的输出，同时利用两个 LED 灯指示电源轨是否正常运行。

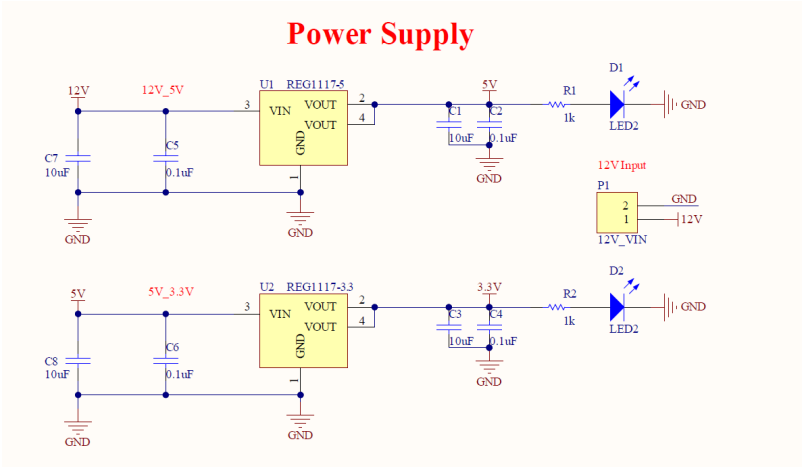

图 3-2. 电源电路

3.2 驱动电路

驱动电路中的栅极驱动器和 MOSFET 均来自德州仪器 (TI)。LM2101 是一款紧凑型高压栅极驱动器，专为驱动采用同步降压或半桥配置的高侧和低侧 N 沟道 MOSFET 而设计。CSD18512Q5B 是一款 40V、1.3mΩ、5mm × 6mm NexFET™ 功率 MOSFET，旨在更大限度地降低功率转换应用中的损耗。电路参数是根据 *LM2101 具有 8V UVLO 的 107V、0.5A、0.8A 半桥驱动器数据表* 中的应用电路设计的。

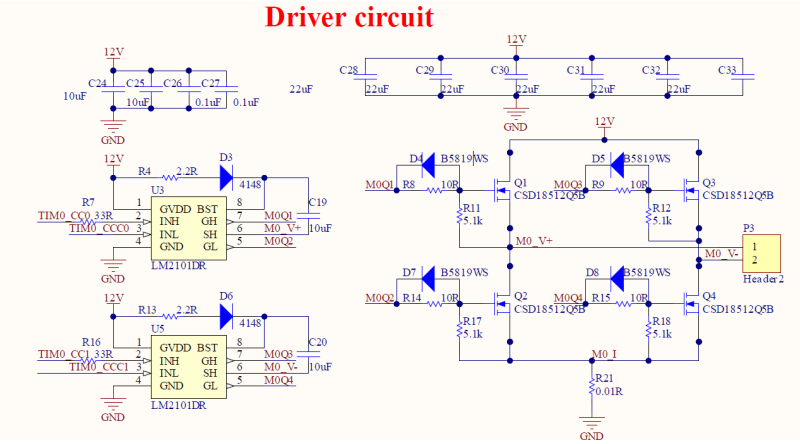

图 3-3. 驱动电路

3.3 采样电路

采样电路主要分为速度采样和电流采样两部分。速度采样由编码器实现，编码器可以向 MCU 发送包含速度信息的方波信号。对于电流反馈电路，电流通过检测电阻器转换为电压。然后电压通过放大器电路被放大至模数转换器 (ADC) 的采样范围内。

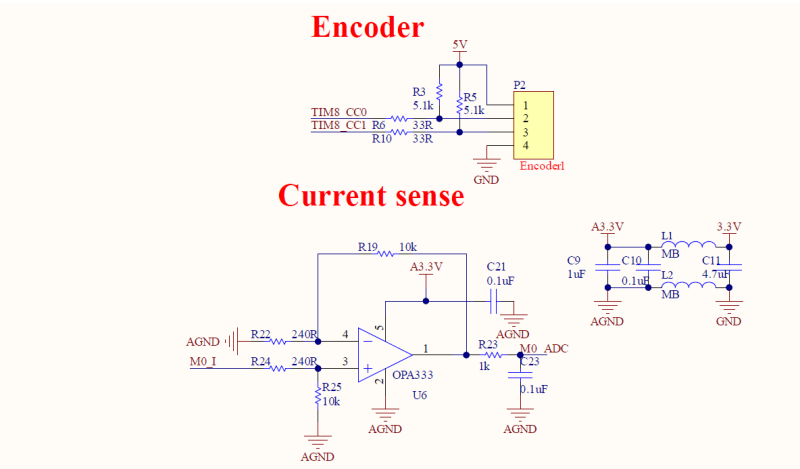

图 3-4. 采样电路

3.4 主控制器电路

MSPM0C1104 的外设电路主要由复位电路和程序烧录接口组成，如图 3-5 所示。

在该解决方案中，有关 MSPM0C1104 引脚功能的分配，请参阅表 3-1。TIM0_CC0 和 TIM0_CCC0 是互补 PWM 信号，TIM0_CC1 和 TIM0_CCC1 是互补 PWM 信号。保留 UART 接口是为了方便调试，只需四个信号即可进行烧录，包括 SWCLK、SWDIO、RST 和 GND。

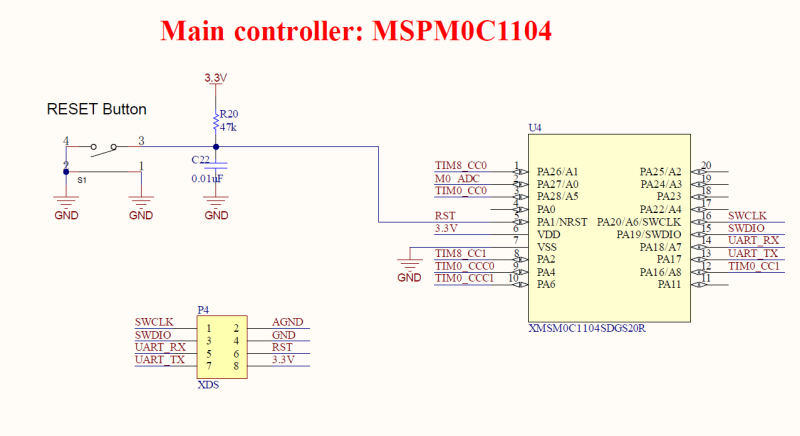

图 3-5. 主控制器电路

表 3-1. 硬件连接

连接类型	连接网络	LP-MSPM0LC1104 引脚编号：引脚名称
电源输入	3.3V	引脚 6：VDD
	GND	引脚 7：VSS
PWM 输出	TIM0_CC0	引脚 3：PA28
	TIM0_CCC0	引脚 9：PA4
	TIM0_CC1	引脚 12：PA16
	TIM0_CCC1	引脚 10：PA6
ADC 输入	M0_ADC	引脚 2：PA27
编码器输入	TIM8_CC0	引脚 1：PA26
	TIM8_CC1	引脚 8：PA2
程序烧录接口	SWCLK	引脚 16：PA20
	SWDIO	引脚 15：PA19
	UART_RX	引脚 14：PA18
	UART_TX	引脚 13：PA17
	RST	引脚 5：PA1

3.5 控制系统简介

3.5.1 驱动方法

直流有刷电机具有多种驱动方法。在该解决方案中，使用了一种相对简单的驱动方法，如图 3-6 所示。该方法只需要一对互补驱动信号即可运行半桥。另外一个半桥中的一个 MOSFET 保持恒定导通状态，而另一个 MOSFET 保持恒定关断状态，具体取决于所需的旋转方向。

当左半桥中的上部晶体管导通时，电源为电机供电，电机电流从左向右流动。当上部晶体管关断时，电机继续通过两个半桥的下部晶体管传导。

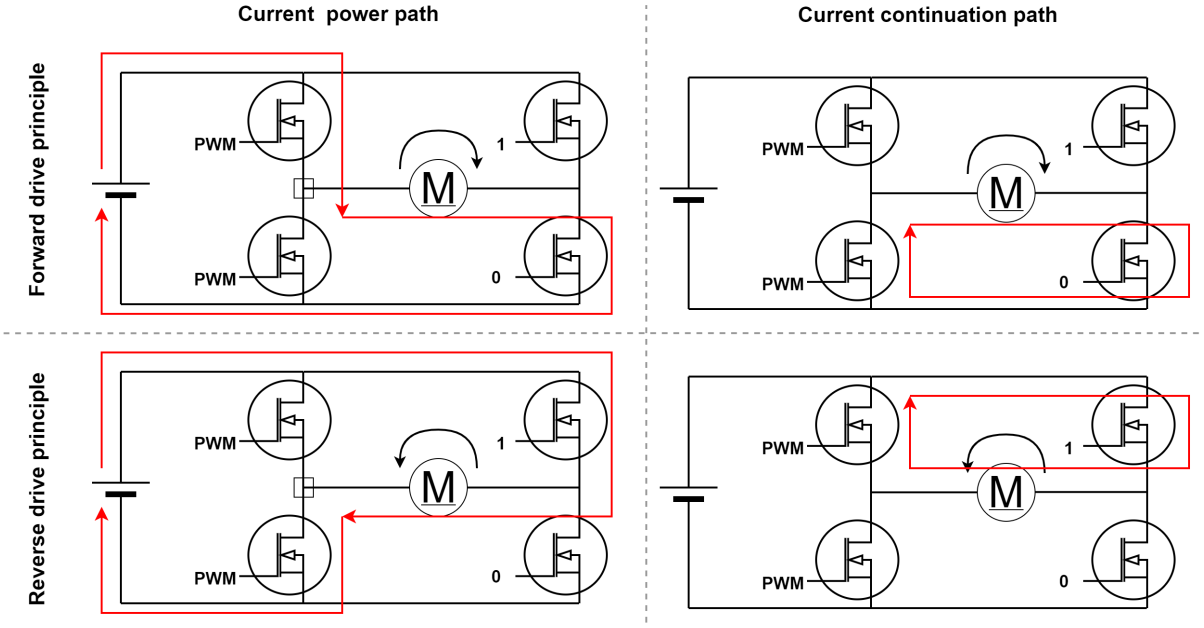

图 3-6. 电机驱动原理

3.5.2 控制方法

在该解决方案中，对速度和电流采用了双环路控制策略，如图 3-7 所示。

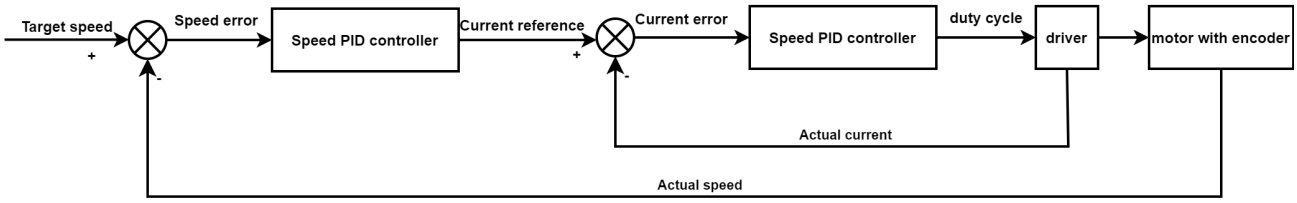

图 3-7. 控制方框图

在速度控制环路中，确定目标速度并测量实际速度。然后，使用目标速度与实际速度之差生成误差信号。该误差信号由控制算法处理以生成控制命令，该命令用于调节电机的输入电压或 PWM 信号以调节电机转速。

在电流控制环路中，对电机电流进行监测和调节，以确保在任何负载条件下都能提供足够的扭矩。电流控制环路还可以为电机提供过载和短路保护。

通过采用该速度外部环路和电流内部环路控制方法，可以实现对有刷直流电机的高精度和高性能控制。因此，该方法广泛用于需要精确可靠控制的应用，例如工业自动化、机器人和各种运动控制应用。

4 软件设计简介

示例代码可以分为三个主要部分：参数初始化、设置旋转方向以及计算计时器中断期间的占空比输出。图 4-1 说明了软件执行逻辑。

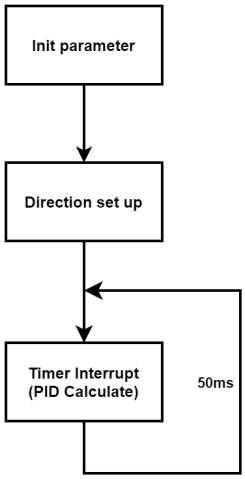

图 4-1. 软件流程

4.1 参数初始化

本节主要涉及三个方面的初始化：电机配置参数、速度环路参数和电流环路参数，如图 4-2 所示。配置设置包括定义目标速度、运行方向（向上或向下）、编码器分辨率、记录时间、最大和最小占空比以及最大电流。为速度和电流控制环路指定了包括 P、I 和 D 增益在内的 PID 控制器参数。此外，代码声明各种易失性变量，用于捕获计数，记录距离，存储实际速度和电流，定义目标电流以及捕获 ADC 读数。

```
// motor configure
#define target_speed              (150)    // speed (MAX:333)
#define run_direction             (1)      // 1:up 0:down
#define ENCODER_RESOLUTION        (1320)   // 11*4*30
#define RECORD_TIME               (0.05)   // record time(s)
#define CC_value_MAX              (475)    // D=97%
#define CC_value_MIN              (25)     // D=3%
#define current_MAX               (0.98)

// Speed PID controller parameters
#define PIDSPEED_KP     (0.01)
#define PIDSPEED_KI     (0.0005)
#define PIDSPEED_KD     (0)
volatile float speed_sum_error=0.0;
volatile float speed_error_last=0.0;
volatile int Capture_count = 0;
volatile int Last_count = 0;
volatile int distance=0;
volatile float actual_speed=0;

// current PID controller parameters
#define PIDCURRENT_KP   (250)
#define PIDCURRENT_KI   (100)
#define PIDCURRENT_KD   (0)
volatile float current_sum_error=0.0;
volatile float current_error_last=0.0;
volatile float actual_current=0;
volatile float target_current=0;
volatile int PRE_ADC=0;
volatile int CC_value=25;
```

图 4-2. 参数初始化

4.2 方向设置

初始化参数后，必须根据设定的参数配置电机旋转，这是通过主函数中的 GPIO 输出高电平或低电平实现的，如图 4-3 所示。之后，启用中断，并开始计算和更新占空比。

```
//main function
int main(void)
{
    SYSCFG_DL_init();
    switch (run_direction)
            {
                case 1: // UP
                    DL_GPIO_clearPins(GPIO_GRP_0_PORT, GPIO_GRP_0_PIN_0_PIN);
                    DL_GPIO_setPins(GPIO_GRP_0_PORT, GPIO_GRP_1_PIN_1_PIN);
                    DL_TimerA_setCaptureCompareValue(PWM_0_INST, 500, DL_TIMER_CC_0_INDEX);
                break;
                case 0: // DOWM
                    DL_GPIO_setPins(GPIO_GRP_0_PORT, GPIO_GRP_0_PIN_0_PIN);
                    DL_GPIO_clearPins(GPIO_GRP_0_PORT, GPIO_GRP_1_PIN_1_PIN);
                    DL_TimerA_setCaptureCompareValue(PWM_0_INST, 0, DL_TIMER_CC_0_INDEX);
                break;
             }
    NVIC_EnableIRQ(TIMER_0_INST_INT_IRQN);
}
```

图 4-3. 主函数

4.3 计时器中断

计时器被配置为向下计数模式，在其达到零时触发中断。计数周期被设置为 50ms，这意味着占空比每 50ms 更新一次。在该中断函数内，首先使用速度环路控制器计算电流基准值，然后钳制电流。随后，在考虑死区时间的同时使用电流环路控制器确定占空比。最后，占空比信息根据电机的实际方向进行更新。

```
void TIMER_0_INST_IRQHandler(void)
{

   switch (DL_TimerG_getPendingInterrupt(TIMER_0_INST))
   {
   case DL_TIMER_IIDX_ZERO:
          //speed loop
          target_current=speed_PID_realize(PIDSPEED_KP,PIDSPEED_KI,PIDSPEED_KD);
          //current loop
          target_current = (target_current > current_MAX) ? current_MAX : target_current; // limit max current
          target_current = (target_current < 0) ? 0 : target_current;   // limit min current
          switch (DL_TimerG_getQEIDirection(QEI_0_INST))
                   {
                        case 1: // UP
                            CC_value=500-current_PID_realize(target_current, PIDCURRENT_KP,PIDCURRENT_KI,PIDCURRENT_KD);
                        break;
                        case 0: // DOWM
                            CC_value=current_PID_realize(target_current, PIDCURRENT_KP,PIDCURRENT_KI,PIDCURRENT_KD);
                        break;
                    }
          //limit D according to dead time
          CC_value = (CC_value > CC_value_MAX) ? CC_value_MAX : CC_value;
          CC_value = (CC_value < CC_value_MIN) ? CC_value_MIN : CC_value;
          //update duty cycle
          DL_TimerA_setCaptureCompareValue(PWM_0_INST, CC_value, DL_TIMER_CC_0_INDEX);
       break;
   default:
       break;
   }
}
```

图 4-4. 计时器中断

4.4 闭环控制器

电流和速度闭环控制器如图 4-5 和图 4-6 所示。在该解决方案中，实现了位置 PID 算法。因此，除限制输出范围之外，还必须限制积分值以避免积分饱和。由于 ADC 分辨率的限制，需要限制初始误差值，以防止闭环死区引起系统振荡。ADC 的采样电流也需要进行均值滤波，以尽可能减少噪声对采样结果的影响并提高系统的稳定性。

MSPM0C1104 具有内置死区时间生成功能的可配置计时器 (TIMA)，可以根据 PID 计算结果方便地生成所需的 PWM 波形。

```
float current_PID_realize(float target_current, float Kp,float Ki, float Kd)
 {
     float PID_out;
     float current_error=0;
     int SUM_ADC=0;
     //get current from ADC
     for(int i=0; i<30; i++)
     {
     DL_ADC12_startConversion(ADC12_0_INST);
     SUM_ADC += DL_ADC12_getMemResult(ADC12_0_INST, DL_ADC12_MEM_IDX_0);
     DL_ADC12_enableConversions(ADC12_0_INST);
     }
     //average filter
     PRE_ADC=SUM_ADC/30;
     //10 bit ADC number convert to real current value
     actual_current=3.3*2.5*PRE_ADC/1024;
     current_error = target_current-actual_current;
     //limiting closed-loop deadband
     current_error = (current_error/target_current <0.1 && current_error >-0.1) ? 0 : current_error;
     current_sum_error+=current_error;     // error integrate
     //Preventing integral saturation
     if (current_sum_error > 10)
         current_sum_error = 10;
     else if (current_sum_error < -10)
         current_sum_error = -10;
     //PID calculate
     PID_out=Kp*current_error+Ki*current_sum_error+Kd*(current_error-current_error_last);     // PID calculation
     current_error_last=current_error;     // Error propagationA
     return PID_out;
 }
```

图 4-5. 电流 PID

```
float speed_PID_realize(float Kp,float Ki, float Kd)
    {
       float PID_out=0;
       float speed_error=0;
       // get speed from encoder
                switch (DL_TimerG_getQEIDirection(QEI_0_INST))
                        {
                              case 1: // UP
                                     Capture_count =(QEI_0_INST->COUNTERREGS.CTR);
                                     distance=Capture_count-Last_count;
                                     distance = (distance <0) ? (0xFFFF+distance) : distance;
                                     actual_speed=(float)distance*60 / (ENCODER_RESOLUTION*RECORD_TIME);
                              break;
                              case 0: // DOWM
                                     Capture_count =(QEI_0_INST->COUNTERREGS.CTR);
                                     distance=Last_count-Capture_count;
                                     distance = (distance <0) ? (0xFFFF+distance) : distance;
                                     actual_speed=(float)distance*60 / (ENCODER_RESOLUTION*RECORD_TIME);
                              break;
                         }
                         Last_count=Capture_count;
        speed_error = target_speed-actual_speed;
       //limiting closed-loop deadband
        speed_error = (speed_error <3 && speed_error >-3) ? 0 : speed_error;
       // PID realize
        speed_sum_error+=speed_error; // error integrate
       //Preventing integral saturation
       if (speed_sum_error > 6000)
           speed_sum_error = 6000;
       else if (speed_sum_error < -6000)
           speed_sum_error = -6000;
       PID_out=Kp*speed_error+Ki*speed_sum_error+Kd*(speed_error-speed_error_last);    // PID calculation
       speed_error_last=speed_error;     // Error propagation
       return PID_out;
    }
```

图 4-6. 速度 PID

5 评估

要评估该解决方案，需要使用以下硬件元件：

- 安装了 Windows® 7 或更高版本和 .NET Framework 4.5 的计算机
- 12V 直流电源
- XDS110 调试探针
- 用于驱动器板和调试器通信的连接器
- 用于 PC 和驱动器板通信的 USB
- 带编码器的有刷直流电机。（ATK-JGB37-520E，规格如表 5-1 所示）

表 5-1. JGB37-520E 电机参数

项目	值
额定电压	直流 12V
电流	0.2-0.5A
怠速	333RPM（峰值）
传动比	30:1

- 电机控制板如图 5-1 所示。

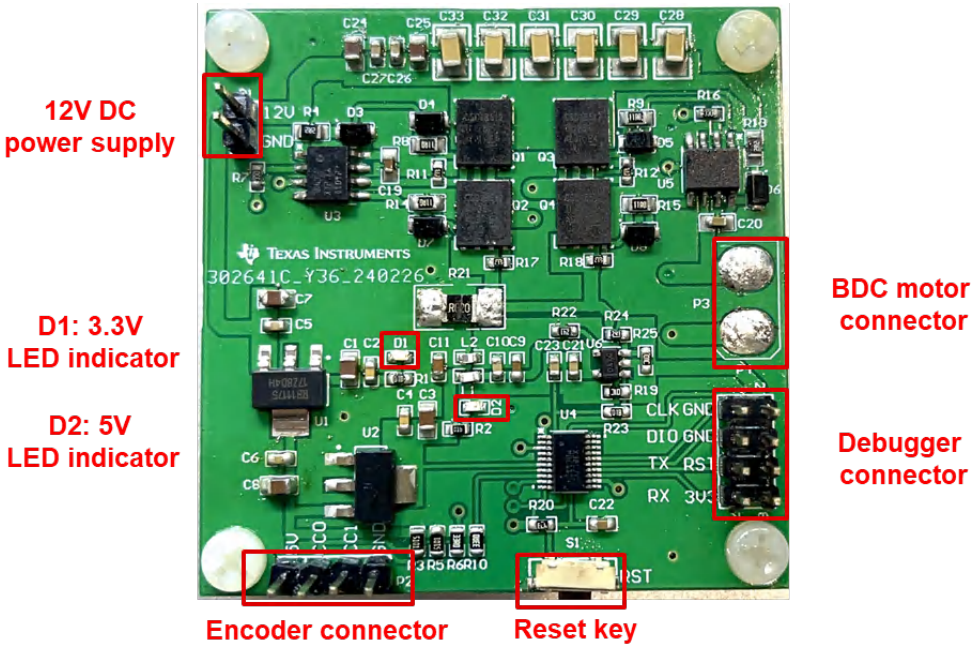

图 5-1. 电机控制板

- 节 4 显示了示例代码。

准备好必要的硬件和代码后，可根据上述说明进行硬件连接。之后，可以通过 Code Composer Studio™ (CCS) 软件将示例代码烧写到芯片中，并对解决方案进行评估。本文基于 TI MSPM0C1104 为有刷直流电机提供了一种经济灵活的驱动解决方案，可显著降低系统成本，同时确保整体性能。客户可以根据自身的实际需求基于该解决方案灵活地进行开发。

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265