AN957

使用 dsPIC30F2010 控制带传感器的 BLDC 电机

著者： *Stan D' Souza*
Microchip Technology

引言

dsPIC30F2010 是一款专门为嵌入式电机控制应用设计的 28 引脚 16 位 MCU。它主要是为交流感应电机（AC Induction Motor， ACIM）、无刷直流电机 （Brushless DC，BLDC）和普通直流电机这些典型的电机类型而专门设计的。以下是 dsPIC30F2010 的一些主要特性：

- 6 个独立或 3 对互补的电机控制专用 PWM 输出。
- 6 输入、采样速率为 500Ksps 的 ADC，可同时采样最多 4 路输入。
- 多种串行通信：UART、 I^2C™ 和 SPI
- 小型封装：6 × 6 mm QFN，适用于嵌入式控制应用
- DSP 引擎可实现控制环的快速响应。

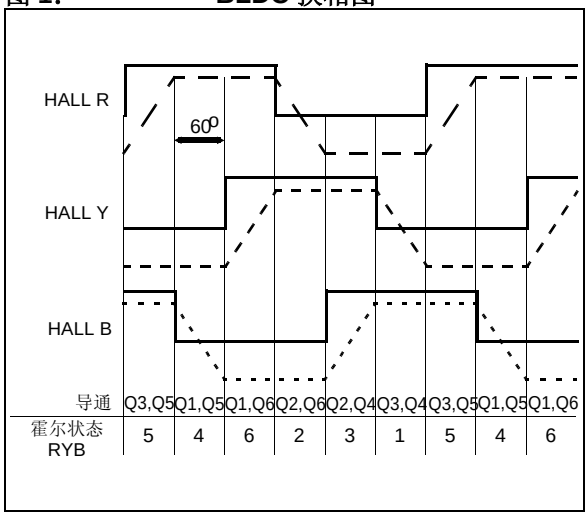
在本应用笔记中，我们将讨论如何使用 dsPIC30F2010 来控制带传感器的 BLDC 电机。欲知 BLDC 的详细工作原理以及 BLDC 电机运行和控制的一般信息，请参阅 AN901_CN。本应用笔记讨论了使用 dsPIC30F2010 控制 BLDC 电机的具体实现，而对 BLDC 电机的细节涉及较少。

BLDC 电机

BLDC 电机基本是内外倒置的直流电机。在一般直流电机中，定子是永磁体。转子上有绕组，对绕组通电。通过使用换向器和电刷将转子中的电流反向来产生旋转的或运动的电场。与之相反，在 BLDC 电机中绕组在定子上而转子是永磁体。“内外倒置的直流电机”这一称谓由此得名。

要使转子转动，必须存在旋转电场。一般来说，三相 BLDC 电机具有 3 相定子，同一时刻为其中的两相通电，以产生旋转电场。此方法相当容易实现，但是为了防止永磁体转子被定子锁住，在知道转子磁体的精确位置的前提下，必须以特定的方式按顺序为定子通电。位置信息通常用霍尔传感器检测转子磁体位置获得，也可采用轴角编码器方式获得。对于典型的三相带传感器的 BLDC 电机，有 6 个不同的工作区间，每个区间中有特定的两相绕组通电。如图 1 所示。

图 1： BLDC 换相图



通过检测霍尔传感器，可以得到一个 3 位编码，编码值的范围从 1 到 6。每个编码值代表转子当前所处的区间。从而提供了需要对哪些绕组通电的信息。因此程序可以使用简单的查表操作来确定要对哪两对特定的绕组通电以使转子转动。

注意状态 “0” 和 “7” 对于霍尔效应传感器而言是无效状态。软件应该检查出这些值并相应地禁止 PWM。

变化通知输入

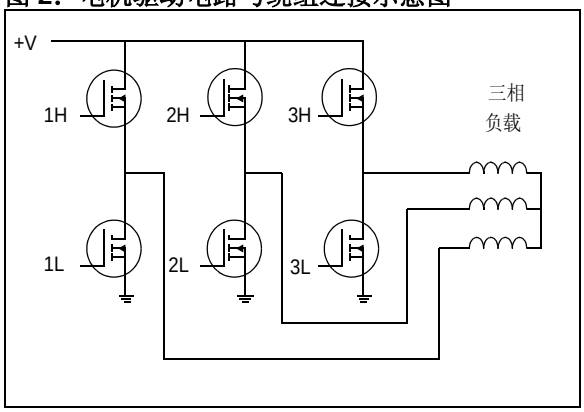
灵活使用上述的技巧，可以将霍尔效应传感器连接到 dsPIC30F2010 的输入引脚上，来检测变化（变化通知（Change Notification， CN）输入）。当这些引脚上的输入电平发生变化时，就会产生中断。在 CN 中断服务程序（Interrupt Service Routine， ISR）中，由用户应用程序读取霍尔效应传感器的值，用以计算偏移量并查表，来正确地驱动 BLDC 电机的绕组。

电机控制脉宽调制 （MCPWM）

使用上面的方法可以使 BLDC 电机全速旋转。然而，为了使 BLDC 电机速度可变，必须在两相绕组的两端加上可变电压。从数字化的语言来讲，就是加在 BLDC 电机绕组上的 PWM 信号的不同占空比可以获得可变电压。

dsPIC30F2010 有六个由 PWM 信号驱动的 PWM 输出。如图 2 所示，通过使用六个开关、IGBT 或 MOSFET，可以将三相绕组驱动为高电平、低电平或根本不通电。例如，当绕组的一端连接到高端驱动器时，就可在低端驱动器上施加占空比可变的 PWM 信号。这与将 PWM 信号加在高端驱动器上，而将低端驱动器连接到 VSS 或 GND 的作用相同。一般更喜欢对低端驱动器施加 PWM 信号。

图 2：电机驱动电路与绕组连接示意图

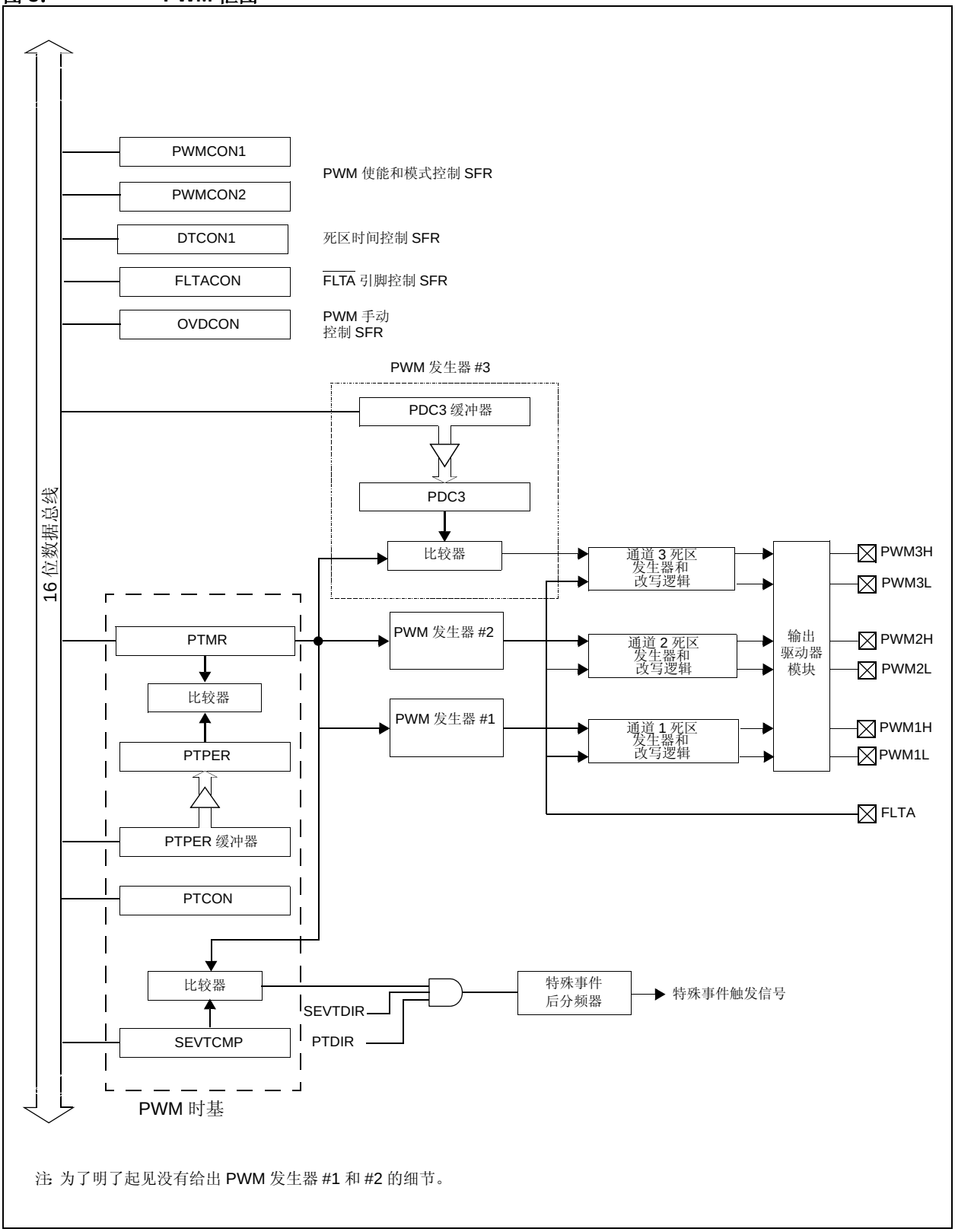


PWM 信号由 dsPIC30F2010 的电机控制（Motor Control，MC）专用 PWM 模块提供。MCPWM 模块是专为电机控制应用而设计的。（阅读本节时，请参阅图 3。）

MCPWM 有一个专用的 16 位 PTMR 时基寄存器。此定时器每隔一个由用户定义的时间间隔进行一次递增计数，该时间间隔最短可以为 TCY。通过选择一个值并将它装入 PTPER 寄存器，用户可以决定所需的 PWM 周期。每个 TCY， PTMR 与 PTPER 作一次比较。当两者匹配时，开始一个新的周期。

控制占空比的方法与此类似，只需在三个占空比寄存器中装入一个值即可。与周期比较不同，每隔 TCY/2 就将占空比寄存器中的值与 PTMR 进行一次比较（即，比较的频率是周期比较的两倍）。如果PTMR的值与PDCx的值相匹配，那么对应的占空比输出引脚就会根据选定的 PWM 模式驱动为低电平或高电平。通过占空比比较产生的三个输出将被分别传输给一对互补的输出引脚，其中一个引脚输出为高电平，而另一个引脚输出为低电平，反之亦然。这两个输出引脚也可以被配置为独立输出模式。当驱动为互补输出时，可以在高电平变低与低电平变高之间插入一段死区。死区是由硬件配置的，最小值为 TCY。插入死区可以防止输出驱动器发生意外的直通现象。

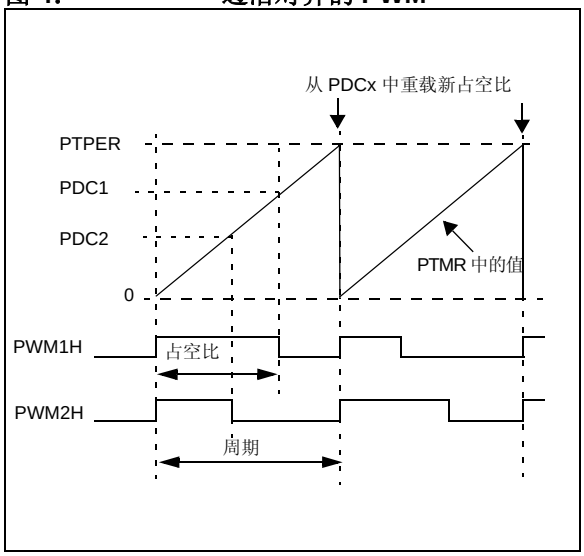
图 3：　　PWM 框图



注：为了明了起见没有给出 PWM 发生器 #1 和 #2 的细节。

可以将 MCPWM 模块配置为多种模式。其中边沿对齐的输出模式可能是最常见的。图 4 描述了边沿对齐的 PWM 的工作原理。在周期开始时，所有输出均驱动为高电平。随着 PTMR 中值的递增，一旦该值与占空比寄存器中的值发生匹配就会导致对应的占空比输出变为低电平，从而表示该占空比结束。PTMR 寄存器的值与 PTPER 寄存器的值匹配导致一个新的周期开始，所有输出变为高电平以开始一个全新的周期。

图 4：　　边沿对齐的 PWM



还可以将 MCPWM 设置为其它模式：中心对齐的 PWM 和单个 PWM。由于它们不用于控制 BLDC 电机，在此将不对这些模式进行讨论。欲知有关这些模式的详细信息，请参阅《*dsPIC30F 系列参考手册*》（DS70046C_CN）。

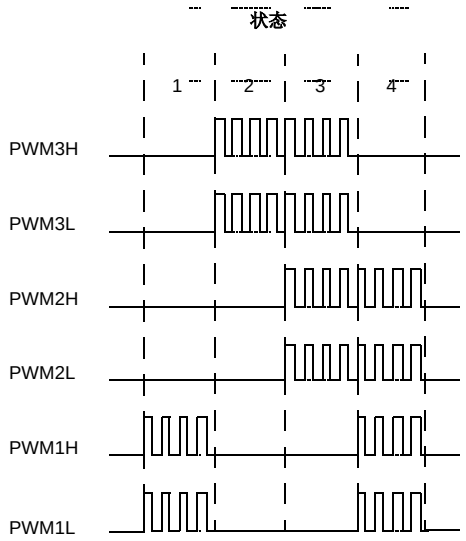
改写是本应用中使用的 MCPWM 的一个重要特征。改写控制是 MCPWM 模块的最后级。它允许用户直接写入 OVDCON 寄存器并控制输出引脚。OVDCON 寄存器中有两个 6 位字段。这两个字段中的每一位对应于一个输出引脚。 OVDCON 寄存器的高字节部分确定对应的输出引脚是由 PWM 信号驱动（当置为 1 时），还是由 OVDCON 寄存器低字节部分中的相应位驱动为有效/无效（当置为 0 时）。此功能允许用户使用 PWM 信号，但是并不驱动所有输出引脚。对于 BLDC 电机，相同的值被写入所有 PDCx 寄存器。

根据 OVDCON 寄存器中的值，用户可以选择哪个引脚获得 PWM 信号以及哪个引脚被驱动为有效或无效。控制带传感器的 BLDC 时，必须根据由霍尔传感器的值所指定的转子位置对两相绕组通电。在 CN 中断服务程序中，首先读霍尔传感器，然后将霍尔传感器的值用作查找表中的偏移量，以找到对应的将要装入 OVDCON 寄存器的值。表 1和图 5说明了如何根据转子所处的区间将不同的值装入 OVDCON 寄存器，从而确定需要对哪些绕组通电。

表 1：　　PWM 输出改写示例

状态	OVDCON<15:8>	OVDCON<7:0>
1	00000011b	00000000b
2	00110000b	00000000b
3	00111100b	00000000b
4	00001111b	00000000b

图 5：　　PWM 输出改写示例

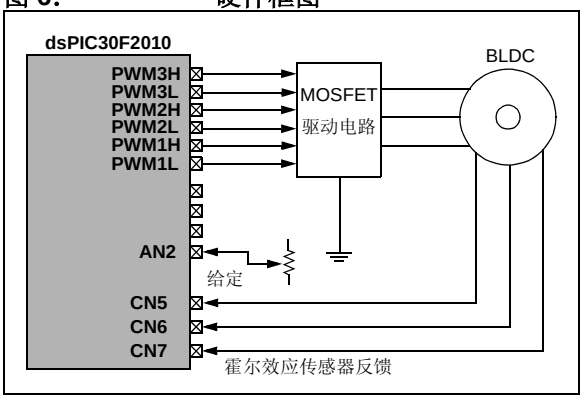


注： 在状态 1- 4 之间切换的时间由用户软件控制。通过向 OVDCON 写入新值控制状态切换。本例中 PWM 输出工作在独立模式。

硬件描述

图 6 中的框图说明了如何使用 dsPIC30F2010 驱动 BLDC 电机。如需详细的原理图，请参阅**附录 C**。

图 6：　　硬件框图



6 个 MCPWM 输出连接到 3 对 MOSFET 驱动器（IR2101S），最终连接到 6 个 MOSFET（IRFR2407）。这些 MOSFET 以三相桥式连接到 3 相 BLDC 电机绕组。在当前实现中，MOSFET 的最大电压为 70V，最大电流为 18A。

注意在使用最大功率时必须提供充分的散热，这一点很重要。MOSFET 驱动器也需要一个较高的电压（15V）来运行，因此需要提供这么高的电平。该电机是 24V BLDC 电机，因此 DC+ 到 DC- 母线电压为 24V。需要提供 5V 的稳压电源来驱动 dsPIC30F2010。3 个霍尔效应传感器的输出信号连接到与变化通知电路相连的输入引脚，使能输入的同时也使能相应中断。若这 3 个引脚中的任何一个发生了电平变化，就会产生中断。为了提供速度给定，将一个电位计连接到 ADC 输入（RB2）。

在 RC14 上提供了一个按钮开关，用于起动和停止电机。为了向电机提供一些电流反馈，在 DC 母线负电压与地或 Vss 之间连接了一个低阻值电阻（25 毫欧）。由此电阻产生的电压被一个外部运放 （MCP6002）放大并反馈到 ADC 输入 （RB1）。

固件描述

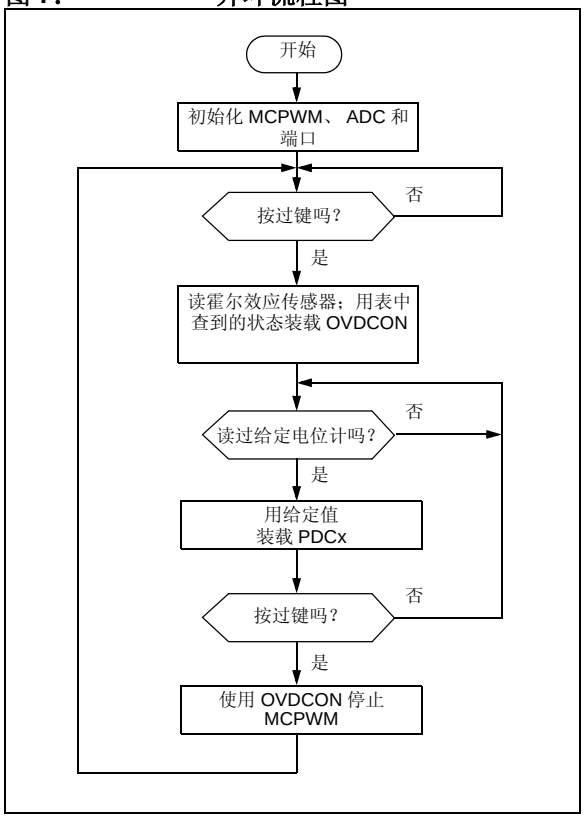
附录 A 和**附录 B** 包含了两个固件程序来举例说明此应用笔记中描述的方法。一个程序使用开环速度控制。另一个使用比例和积分反馈来实现闭环速度控制。

对于实际应用而言，开环方式通常是不实用的。此处介绍它主要是为了阐明 BLDC 电机的驱动方法。

开环控制

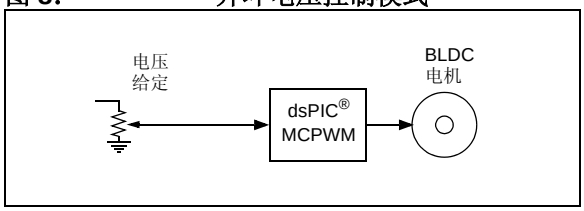
在开环控制中，MCPWM 根据来自速度电位计的电压输入直接控制电机速度。初始化 MCPWM、ADC、端口和变化通知输入之后，程序将等待一个激活信号 （例如，按一个键）来表示开始 （参见图 7）。按下键后，程序将读霍尔传感器。根据读到的值，从表中取出对应的值并将它写入 OVDCON。此时电机开始旋转。

图 7：　　开环流程图



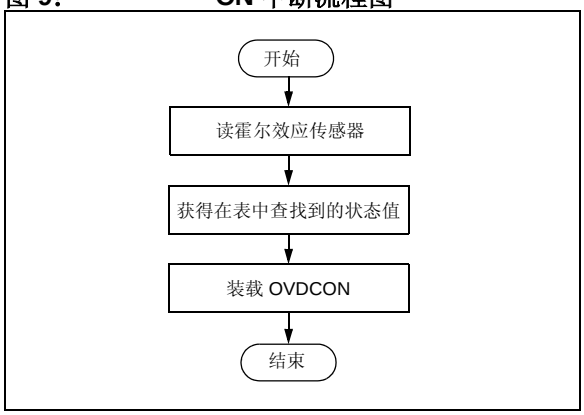
最初占空比值保持在默认值 50%。但是，在主程序的第一个循环，将读电位计并将其值（即正确的给定值）作为占空比插入。该值决定电机的速度。占空比值越高，电机转得越快。图 8 所示电机速度由电位计控制。

图 8：　　开环电压控制模式



霍尔效应传感器连接到变化通知引脚。允许 CN 中断。当转子旋转时，转子磁体的位置发生变化，从而使转子进入不同的区间。 CN 中断表示转子进入每个新位置。在 CN 中断程序（如图 9 所示）中，读霍尔效应传感器的值，并根据该值得到一个表查找值，并将它写入 OVDCON 寄存器。此操作将确保在正确的区间对正确的绕组通电，从而使电机继续旋转。

图 9：　　CN 中断流程图



相位超前

欲知有关相位超前以及实现方式的详细信息，请参阅 AN901_CN。

闭环控制

在闭环控制固件版本中，主要的不同是使用电位计来设定速度给定。控制环提供了对速度的比例和积分（Proportional and Integral， PI）控制。要测量实际速度，可以使用 TMR3 作为定时器来选通一个完整的电周期。由于我们使用的是 10 极电机，因此一个机械周期将由 5 个电周期构成。如果 T（秒）是一个电周期的时间，那么速度 S = 60/ （P/2*T） rpm，其中 P 是电机的极数。控制如图 10 所示。闭环控制流程图如图 11 所示。

图 10：　　闭环电压控制模式

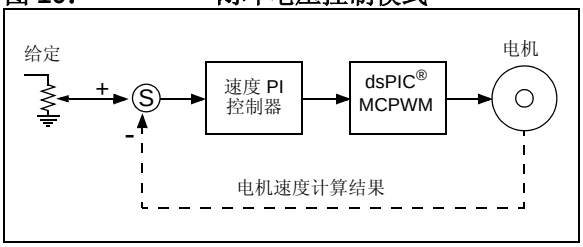
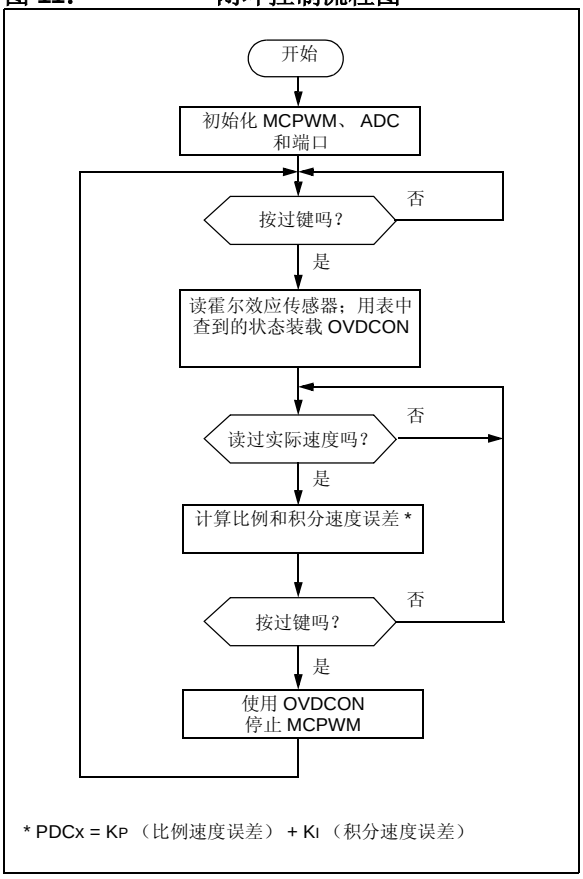


图 11：　　闭环控制流程图



结论

dsPIC30F2010 非常适合对带传感器的 BLDC 电机进行闭环控制。外设和 DSP 引擎为带传感器的 BLDC 应用提供了足够宽的带宽，并为客户的应用程序提供了充足的代码空间。

参考书目

- AN885—Brushless DC （BLDC） Motor Fundamentals
- AN901_CN—dsPIC30F 在无传感器 BLDC 控制中的应用
- AN857— Brushless DC Motor Control Made Easy
- AN899 — Brushless DC Motor Control Using PIC18FXX31 MCUs

附录 A： 开环控制的源代码清单

此附录包含了开环控制的源代码清单。


```
//------------------------------------------------------------------------------
//                        Software License Agreement
//
// The software supplied herewith by Microchip Technology Incorporated
// (the "Company") is intended and supplied to you, the Company's customer,
// for use solely and exclusively with products manufacture by the Company.
// The software is owned by the Company and/or its supplier, and is protected under
// applicable copyright laws. All rights are reserved. Any use in violation of the
// foregoing restrictions may subject the user to criminal sanctions under applicable
// laws, as well as to civil liability for the breach of the terms and conditions of
// this license.
//
// THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES, WHETHER EXPRESS,
// IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
// MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
// THE COMPANY SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR
// CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
//------------------------------------------------------------------------------
//  文件: ClosedLoopSenBLDC.c
//
//  编写者: Stan D'Souza, Microchip Technology
//
// 下列文件应该包含在 MPLAB 项目中:
//
//     ClosedLoopSenBLDC.c——主源代码文件
//     p30f2010.gld——链接描述文件

//
//
//------------------------------------------------------------------------
//
// 版本历史
//
// 10/01/04——第一版
//------------------------------------------------------------------------
/*************************************************************
    以下是低端驱动器表。在此 StateLoTable 中，
    在低端驱动器施加 PWM 信号，而高端驱动器为 “导通”或 “截止”状态。
    在本练习中使用此表。
/*************************************************************/

unsigned int StateLoTable[] = {0x0000, 0x0210, 0x2004, 0x0204,
                               0x0801, 0x0810, 0x2001, 0x0000};
/*************************************************************
以下是变化通知引脚 CN5、 CN6 和 CN7 的中断向量。
当霍尔传感器改变状态时，将引起中断，指令执行将转到下面的子程序。
然后用户必须读端口 B 的第 3 位、第 4 位和第 5 位，
对读到的值进行移位和调节以使之读作 1、 2……6。
然后将调整后的值用作查找表 StateLoTable 中的偏移量
以确定装入 OVDCON 寄存器的值。
*************************************************************/
```

```
void _ISR _CNInterrupt(void)
{
    IFS0bits.CNIF = 0;                  // 清零标志
    HallValue = PORTB & 0x0038;         // 屏蔽其它位，保留 RB3、 RB4 和 RB5
    HallValue = HallValue >> 3;         // 执行 3 次右移
    OVDCON = StateLoTable[HallValue];
}

/*********************************************************************
ADC 中断用给定的电位计值装载 PDCx 寄存器。
仅在电机运行时执行此操作。

*********************************************************************/

void _ISR _ADCInterrupt(void)
{
    IFS0bits.ADIF = 0;
    if (Flags.RunMotor)
    {
        PDC1 = ADCBUF0;                         // 赋值……
        PDC2 = PDC1;                            // 并装载所有的三个 PWM……
        PDC3 = PDC1;                            // 占空比寄存器
    }
}

int main(void)
{
    LATE = 0x0000;
    TRISE = 0xFFC0;                             // 设置为输出 PWM 信号
    CNEN1 = 0x00E0;                             // 使能 CN5、 CN6 和 CN7
    CNPU1 = 0x00E0;                             // 使能内部上拉
    IFS0bits.CNIF = 0;                          // 清零 CNIF
    IEC0bits.CNIE = 1;                          // 允许 CN 中断
    InitMCPWM();
    InitADC10();
    while(1)
    {       while (!S2);                        // 等待按开始键
        while (S2)                              // 等待直到释放按键
            DelayNmSec(10);
        // 在 PORTB 上读霍尔位置传感器
        HallValue = PORTB & 0x0038;             // 屏蔽其它位，保留 RB3、 RB4 和 RB5
        HallValue = HallValue >> 3;             // 右移以获得值 1、 2……6
        OVDCON = StateLoTable[HallValue]; // 装载改写控制寄存器
        PWMCON1 = 0x0777;                       // 使能 PWM 输出
        Flags.RunMotor = 1;                     // 将标志置 1
        while (Flags.RunMotor)                  // 当电机运行时
            if (S2)                             // 如果按下 S2
            {
                PWMCON1 = 0x0700;               // 禁止 PWM 输出
                OVDCON = 0x0000;                // 将 PWM 改写为低电平
                Flags.RunMotor = 0;             // 复位运行标志
                while (S2)                      // 等待释放按键
                    DelayNmSec(10);
            }
    }   // while (1) 结束


}
```

```
/*********************************************************************
       以下代码用于设置 ADC 寄存器，该代码可实现下列功能：
       1. 1 个通道转换（本例中，该通道为 RB2/AN2）
       2. PWM 触发信号启动转换
       3. 电位计连接到 CH0 和 RB2
       4. 手动停止采样和启动转换
       5. 手动检查转换完成

*********************************************************************/
void InitADC10(void)
{

 ADPCFG = 0xFFF8;                  // 将端口 B 的 RB0 到 RB2 配置为模拟引脚；将其它引脚配置为数字引脚
 ADCON1 = 0x0064;                  // PWM 启动转换
 ADCON2 =  0x0200;                 // 同时采样 4 个通道
 ADCHS = 0x0002;                   // 将 RB2/AN2 作为 CH0 连接到电位计……
                                   // ch1 连接母线电压、 Ch2 连接电机， Ch3 连接电位计
 ADCON3 = 0x0080;                  // Tad 来源于内部 RC （4uS）
 IFS0bits.ADIF = 0;
 IEC0bits.ADIE = 1;

 ADCON1bits.ADON = 1;              // 启动 ADC
}

/*********************************************************************
InitMCPWM，对 PWM 做以下初始化：
1. FPWM = 16000 hz
2. 独立的 PWM
3. 使用 OVDCON 控制输出
4. 用从电位计读取的 ADC 值设置占空比
5. 将 ADC 设置为由 PWM 特殊触发信号触发
*********************************************************************/

void InitMCPWM(void)
{
    PTPER = FCY/FPWM - 1;

    PWMCON1 = 0x0700;              // 禁止 PWM
    OVDCON = 0x0000;               // 允许使用 OVD 控制
    PDC1 = 100;                    // 将 PWM1、 PWM2 和 PWM3 初始化为 100
    PDC2 = 100;
    PDC3 = 100;
    SEVTCMP = PTPER;
    PWMCON2 = 0x0F00;              // 后分频比设为 1:16
    PTCON = 0x8000;                // 启动 PWM

}

//--------------------------------------------------------------------------
// 这是普通的 1 ms 延迟程序，用于提供 1 mS 到 65.5 秒的延迟。
// 如果 N = 1，则延迟为 1 mS；如果 N = 65535，则延迟为 65,535 mS。
// 注意 FCY 用于计算。
// 请根据上述定义语句做出必要的更改（PLLx4 或 PLLx8 等）
// 以计算出正确的 FCY。

void DelayNmSec(unsigned int N)
{
unsigned int j;
while(N--)
    for(j=0;j < MILLISEC;j++);
}
```

附录 B： 闭环控制的源代码清单

此附录包含了闭环控制的源代码清单。

```
/*******************************************************************************
//                        Software License Agreement
//
// The software supplied herewith by Microchip Technology Incorporated
// (the "Company") is intended and supplied to you, the Company's customer,
// for use solely and exclusively with products manufacture by the Company.
// The software is owned by the Company and/or its supplier, and is protected under
// applicable copyright laws. All rights are reserved. Any use in violation of the
// foregoing restrictions may subject the user to criminal sanctions under applicable
// laws, as well as to civil liability for the breach of the terms and conditions of
// this icense.
//
// THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES, WHETHER EXPRESS,
// IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
// MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
// THE COMPANY SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR
// CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
//*******************************************************************************
// 文件：ClosedLoopSenBLDC.c
//
// 编写者：Stan D'Souza， Microchip Technology
//
// 下列文件应该包含在 MPLAB 项目中：
//
//     ClosedLoopSenBLDC.c——主源代码文件
//     p30f2010.gld——链接描述文件
//
//-------------------------------------------------------------------------
// 版本历史
//
// 10/01/04——第一版
//-------------------------------------------------------------------------
//*************************************************************************
ClosedLoopSenBLDC.c 用于对带传感器的 BLDC 电机进行闭环控制。
它的任务包括：
将霍尔传感器的传感变化信号输出到 CN5、 CN6 和 CN7 引脚 （端口 B）
在 CN 中断期间，通过读端口 B 读取来自传感器的输入信号
分析并确定位置的状态 1、 2……6。
使用查找表 StateLoTable，该表用于确定
OVDCON 的值。将在表中找到的值装入 OVDCON。

将 PWM 初始化为产生独立连续的 PWM 信号。
电位计参考电压值用来确定给定 （即要求的）
电机速度。然后使用要求的速度值和
实际的速度值来确定比例速度误差和
积分速度误差。有了这两个值，就可以使用下面的公式计算出新的占空比：
NewDutyCycle （新的占空比） = Kp*（比例速度误差） + Ki*（积分速度误差）
然后将 10 位的 NewDutyCycle （新占空比值）装入所有的 3 个 PWM 占空比寄存器中。

FPWM = 16000hz

设置 ADC，使通过 PWM 触发信号来启动转换。
*************************************************************************/
```

```
#define __dsPIC30F2010__
#include "c:\pic30_tools\support\h\p30F2010.h"


#define FCY  10000000// xtal = 5.0Mhz； PLLx8
#define MILLISEC FCY/10000// 1 mS 延迟常数
#define FPWM 16000
#define Ksp1200
#define Ksi10
#define RPMConstant60*(FCY/256)

#define S2!PORTCbits.RC14


void InitTMR3(void);
void InitADC10(void);
void AverageADC(void);
void DelayNmSec(unsigned int N);
void InitMCPWM(void);
void CalculateDC(void);
void GetSpeed(void);

struct {
          unsigned RunMotor : 1;
          unsigned Minus : 1;
          unsigned unused : 14;
       }  Flags;

unsigned int HallValue;
int Speed;
unsigned int Timer3;
unsigned char Count;
unsigned char SpeedCount;
int DesiredSpeed;
int ActualSpeed;
int SpeedError;
int DutyCycle;
int SpeedIntegral;

//************************************************************
    以下是低端驱动器表。在此 StateLoTable 中，
    在低端驱动器上施加 PWM 信号，而高端驱动器为 “导通”或“截止”状态。
    在本练习中使用此表。
************************************************************/

unsigned int StateLoTable[] = {0x0000, 0x1002, 0x0420, 0x0402,
                               0x0108, 0x1008, 0x0120, 0x0000};
/*************************************************************
以下是变化通知引脚 CN5、 CN6 和 CN7 的中断向量。
当霍尔传感器改变状态时，将引起中断，指令执行将转到下面的子程序。
然后用户必须读端口 B 的第 3 位、第 4 位和第 5 位，
对读到的值进行移位和调节以使之读作 1、 2……6。
然后将调整后的值用作查找表 StateLoTable 中的偏移量
以确定装入 OVDCON 寄存器的值。
*************************************************************/
```

```
void _ISR _CNInterrupt(void)
{
    IFS0bits.CNIF = 0;                  // 清零标志
    HallValue = PORTB & 0x0038;         // 屏蔽其它位，保留 RB3、 RB4 和 RB5
    HallValue = HallValue >> 3;         // 执行 3 次右移
    OVDCON = StateLoTable[HallValue];// 装载改写控制寄存器
}

/************************************************************************
ADC 中断用给定的电位计值装载 DesiredSpeed 变量。
然后用该值来确定速度误差。当电机不运行时，
使用来自电位计的直接给定值作为 PDC 值。
************************************************************************/
void _ISR _ADCInterrupt(void)
{
    IFS0bits.ADIF = 0;
    DesiredSpeed = ADCBUF0;
    if (!Flags.RunMotor)
        {
        PDC1 = ADCBUF0;                 // 赋值……
        PDC2 = PDC1;                    // 并装载所有的三个 PWM……
        PDC3 = PDC1;                    // 占空比寄存器
        }
}

/************************************************************************
该主程序控制初始化，按键以起动
和停止电机。
************************************************************************/

int main(void)
{
    LATE = 0x0000;
    TRISE = 0xFFC0;                 // 设置为输出 PWM 信号
    CNEN1 = 0x00E0;                 // 使能 CN5、 CN6 和 CN7
    CNPU1 = 0x00E0;                 // 使能内部上拉
    IFS0bits.CNIF = 0;              // 清零 CNIF
    IEC0bits.CNIE = 1;              // 允许 CN 中断
    SpeedError = 0;
    SpeedIntegral = 0;
    InitTMR3();
    InitMCPWM();
    InitADC10();
    while(1)
    {
        while (!S2);                // 等待按开始键
        while (S2)                  // 等待直到释放按键
            DelayNmSec(10);
        // 通过端口 B 读来自霍尔位置传感器的信号
        HallValue = PORTB & 0x0038;         // 屏蔽其它位，保留 RB3、 RB4 和 RB5
        HallValue = HallValue >> 3;         // 右移以获得值 1、 2……6
        OVDCON = StateLoTable[HallValue];// 装载改写控制寄存器
        PWMCON1 = 0x0777;               // 使能 PWM 输出
        Flags.RunMotor = 1;             // 将标志置 1
        T3CON = 0x8030;                 // 启动 TMR3
        while (Flags.RunMotor)          // 当电机运行时
            if (!S2)                    // 如果未按下 S2
```

```
        {
            if (HallValue == 1)         // 如果位于区间 1
            {
            HallValue = 0xFF;               // 强制一个新值作为区间值
            if (++Count == 5)               // 对于 10 极电机，将此代码段执行 5 个电周期 （即 1 个
                                            // 机械周期）
                {
                Timer3 = TMR3;// 读 tmr3 的最新值
                TMR3 = 0;
                Count = 0;
                GetSpeed();// 确定速度
                }
            }
        }
        else          // 如果按下 S2，停止电机
        {
            PWMCON1 = 0x0700;// 禁止 PWM 输出
            OVDCON = 0x0000; // 将 PWM 改写为低电平。
            Flags.RunMotor = 0;// 复位运行标志
            while (S2)// 等待释放按键
                DelayNmSec(10);
        }
    }  // while (1) 结束

}

/*********************************************************************
以下代码用于设置 ADC 寄存器，该代码可实现下列功能：
       1. 1 个通道转换 ( 本例中，该通道为 RB2/AN2)
       2. PWM 触发信号启动转换
       3. 电位计连接到 CH0 和 RB2
       4. 手动停止采样和启动转换
       5. 手动检查转换完成
*********************************************************************/
void InitADC10(void)
{


 ADPCFG = 0xFFF8;                   // 将端口 B 的 RB0 到 RB2 配置为模拟引脚；将其它引脚配置为数字引脚
 ADCON1 = 0x0064;                   // PWM 启动转换
 ADCON2 =  0x0200;                  // 采样 CH0 通道
 ADCHS = 0x0002;                    // 将 RB2/AN2 作为 CH0 连接到电位计。
 ADCON3 = 0x0080;                   // Tad 来源于内部 RC （4uS）
 IFS0bits.ADIF = 0;                 // 清零标志
 IEC0bits.ADIE = 1;                 // 允许中断

 ADCON1bits.ADON = 1;               // 启动 ADC
}
```

```
/**********************************************************************
InitMCPWM，对 PWM 做以下初始化：
1. FPWM = 16000 hz
2. 独立的 PWM
3. 使用 OVDCON 控制输出
4. 使用 PI 算法和速度误差设置占空比
5. 将 ADC 设置为由 PWM 特殊触发信号触发
**********************************************************************/

void InitMCPWM(void)
{
    PTPER = FCY/FPWM - 1;

    PWMCON1 = 0x0700;           // 禁止 PWM
    OVDCON = 0x0000;            // 允许使用 OVD 控制
    PDC1 = 100;                 // 将 PWM1、 PWM2 和 PWM3 初始化为 100
    PDC2 = 100;
    PDC3 = 100;
    SEVTCMP = PTPER;            // 特殊触发值等于 16 个周期值
    PWMCON2 = 0x0F00;           // 后分频比设为 1:16
    PTCON = 0x8000;             // 启动 PWM

}

/**********************************************************************
Tmr3 用于确定速度，因此它被设置为使用 Tcy/256 作为时钟周期进行计数。

**********************************************************************/

void InitTMR3(void)
{
    T3CON = 0x0030;             // 内部 Tcy/256 时钟
    TMR3 = 0;
    PR3 = 0x8000;
}

/**********************************************************************
GetSpeed，通过使用每个机械周期内 TMR3 中的值确定
电机的精确速度。
**********************************************************************/

void GetSpeed(void)
{
    if (Timer3 > 23000)                    // 如果 TMR3 值很大，则忽略此次读取
        return;
    if (Timer3 > 0)
        Speed = RPMConstant/(long)Timer3;// 获得以 RPM 为单位的速度
    ActualSpeed += Speed;
    ActualSpeed = ActualSpeed >> 1;
    if (++SpeedCount == 1)
        {SpeedCount = 0;CalculateDC();}
}
```

```
/*************************************************************************
CalculateDC，使用 PI 算法来计算新的 DutyCycle（占空比）值，
该值将被载入 PDCx 寄存器。
*************************************************************************/

void CalculateDC(void)
{

    DesiredSpeed = DesiredSpeed*3;
    Flags.Minus = 0;
    if (ActualSpeed > DesiredSpeed)
    SpeedError = ActualSpeed - DesiredSpeed;
    else
        {
        SpeedError = DesiredSpeed - ActualSpeed;
        Flags.Minus = 1;
        }
    SpeedIntegral += SpeedError;
    if (SpeedIntegral > 9000)
        SpeedIntegral = 0;
    DutyCycle = (((long)Ksp*(long)SpeedError + (long)Ksi*(long)SpeedIntegral) >> 12);
    DesiredSpeed = DesiredSpeed/3;
    if (Flags.Minus)
    DutyCycle = DesiredSpeed + DutyCycle;
    else DutyCycle = DesiredSpeed - DutyCycle;
    if (DutyCycle < 100)
        DutyCycle = 100;
    if (DutyCycle > 1250)
        {DutyCycle = 1250;SpeedIntegral = 0;}
    PDC1 = DutyCycle;
    PDC2 = PDC1;
    PDC3 = PDC1;

}

//-----------------------------------------------------------------------
// 这是通用的 1 ms 延迟程序，用于提供 1 mS 到 65.5 秒的延迟。
// 如果 N = 1，则延迟为 1 mS；如果 N = 65535，则延迟为 65,535 mS。
// 注意 FCY 会用于计算。
// 请根据上述定义语句作必要的更改（PLLx4 或 PLLx8 等）
// 以计算出正确的 FCY。

void DelayNmSec(unsigned int N)
{
unsigned int j;
while(N--)
    for(j=0;j < MILLISEC;j++);
}
```

附录 C： 原理图

此附录包含了使用 dsPIC30F2010 来控制带传感器的 BLDC 电机的原理图。

图 C-1： 电机控制原理图 1

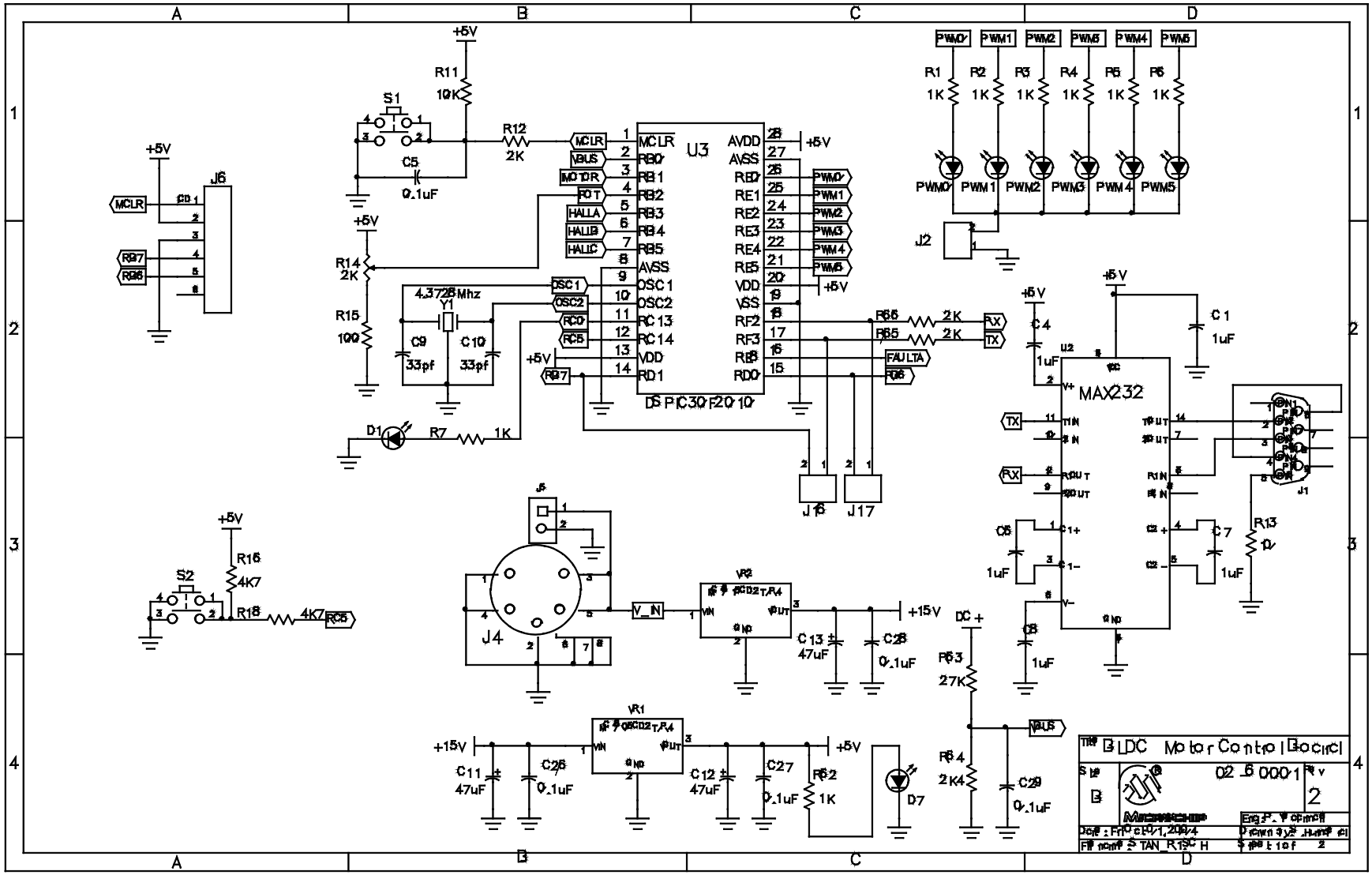
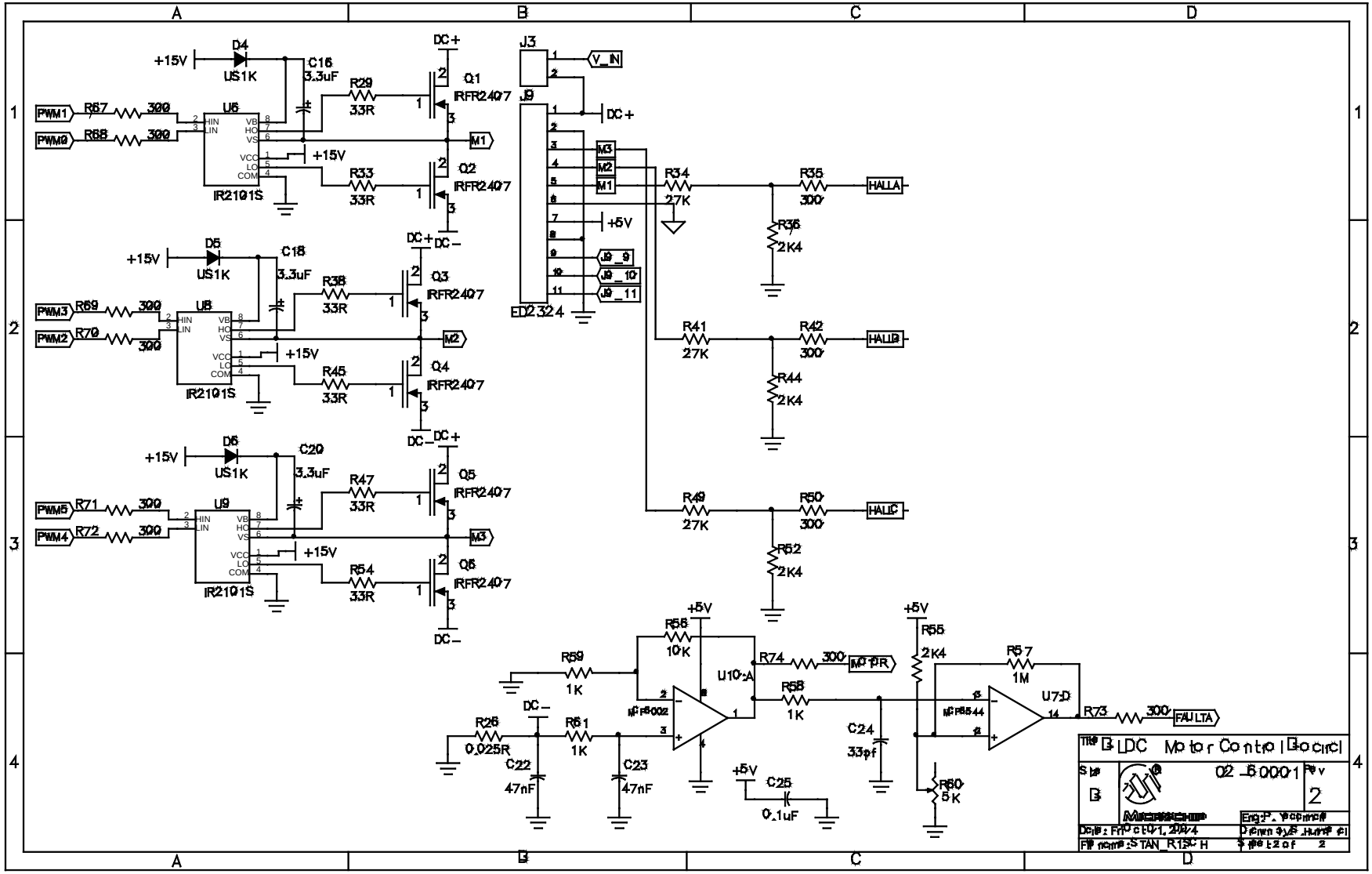


图 C-2： 电机控制原理图 2



请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其它半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案（Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其它受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 及位于加利福尼亚州 Mountain View 的全球总部、设计中心和晶圆生产厂均于 2003 年 10 月通过了 ISO/TS-16949:2002 质量体系认证。公司在 PICmicro® 8 位单片机、 KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品方面的质量体系流程均符合 ISO/TS-16949:2002。此外， Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://support.microchip.com
网址：www.microchip.com

亚特兰大 Atlanta
Alpharetta, GA
Tel: 1-770-640-0034
Fax: 1-770-640-0307

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

科科莫 Kokomo
Kokomo, IN
Tel: 1-765-864-8360
Fax: 1-765-864-8387

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

圣何塞 San Jose
Mountain View, CA
Tel: 1-650-215-1444
Fax: 1-650-961-0286

加拿大多伦多 Toronto
Mississauga, Ontario,
Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

中国 – 北京
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

中国 – 成都
Tel: 86-28-8676-6200
Fax: 86-28-8676-6599

中国 – 福州
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

中国 – 香港特别行政区
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 – 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 – 深圳
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

中国 – 顺德
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

中国 – 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 – 武汉
Tel: 86-27-5980-5330
Fax: 86-27-5980-5118

台湾地区 – 高雄
Tel: 886-7-536-4818
Fax: 886-7-536-4803

台湾地区 – 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

台湾地区 – 新竹
Tel: 886-3-572-9526
Fax: 886-3-572-6459

亚太地区

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-2229-0061
Fax: 91-80-2229-0062

印度 India - New Delhi
Tel: 91-11-5160-8631
Fax: 91-11-5160-8632

日本 Japan - Kanagawa
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Penang
Tel:011-604-646-8870
Fax:011-604-646-5086

菲律宾 Philippines - Manila
Tel: 011-632-634-9065
Fax: 011-632-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

欧洲

奥地利 Austria - Weis
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

丹麦 Denmark - Ballerup
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 France - Massy
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Ismaning
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

英国 England - Berkshire
Tel: 44-118-921-5869
Fax: 44-118-921-5820

06/08/05