MICROCHIP

AN3381

使用温度输入和转速表反馈控制无刷直流风扇的速度

简介

作者：Christopher Best，Microchip Technology Inc.

电子电路以热的形式耗散能量。随着电压强制自由电子运动，电子的动能增加。当电子通过材料时，材料可能会限制电子流动，从而导致电子相互碰撞。当电子相互碰撞时，一些动能以热的形式释放。复杂的电子元件（如微处理器）可能包含数十亿个通过阻性材料互连的晶体管。这些器件不可避免地会由于能量传递而发热。随着元件工作负荷的增加，电阻损耗也会增加，从而导致温度升高。如果内部电路过热，则材料性能下降可能会导致器件失效。解决此问题的一种简单方法是使用无刷直流（brushless DC，BLDC）风扇将热量从器件中带走。

当今许多 BLDC 风扇都被设计为风扇速度随温度升高而加快，目的是为发热元件提供更多的气流，从而带走更多的热量。

本应用笔记讨论了如何使用 PIC18-Q43 的外设，包括：使用具有计算功能的模数转换器（Analog-to-Digital Converter with Computation，ADCC）模块通过热敏电阻监视部件的温度；利用 16 位脉宽调制器（Pulse-Width Modulator，PWM）模块根据温度输入分别调节三个风扇的速度；将捕捉/比较/ PWM（Capture/Compare/PWM，CCP）模块连接至每个风扇的转速计输出，从而确保风扇以预期的速度运行；利用通用异步收发器（Universal Asynchronous Receiver Transmitter，UART）模块将元件的温度和每个风扇的速度发送到串行终端以便查看。

© 2021 Microchip Technology Inc.

目录

简介 1

1. 项目说明和工作原理 3
   - 1.1. 项目说明 3
   - 1.2. 工作原理 11

2. BLDC 基础知识 18
   - 2.1. 转子总成 18
   - 2.2. 定子 19
   - 2.3. 框架 20
   - 2.4. 电机驱动电路 21

3. 结论 23

Microchip 网站 24

产品变更通知服务 24

客户支持 24

Microchip 器件代码保护功能 24

法律声明 24

商标 25

质量管理体系 25

全球销售及服务网点 26

1. 项目说明和工作原理

1.1 项目说明

本项目旨在利用热敏电阻检测温度来单独控制三台轴流风扇的速度，利用三个独立的 PWM 输出分别设置每个风扇的速度，并通过转速计反馈来确保每台风扇在所需速度下运行。

1.1.1 项目元件

本项目使用以下元件：

- Microchip PIC18F57Q43 Curiosity Nano 开发板（部件编号 DM164150）
- Microchip Curiosity Nano Base for Click boards®（部件编号 AC164162）
- Microchip MCP2200 USB 转 UART 分路模块（部件编号 ADM00393）
- Sanyo Denki San Ace 40 4 线轴流风扇（部件编号 9GA0405P6F001）
- DROK 10 kΩ B3950 NTC 温度传感器探头（部件编号 B01MR37GOQ）
- 10 kΩ 通孔电阻（精度 1%）
- 5V 电源
- 跳线

1.1.2 外设概述

本项目利用 PIC18F57Q43 单片机中包含的四个外设：

- 具有计算功能的模数转换器（ADCC）
- 16 位脉宽调制器（PWM）
- 捕捉/比较/PWM（CCP）
- 通用异步收发器（UART）

1.1.2.1 ADCC 概述

PIC18F57Q43 的 ADCC 模块是一个 12 位模数转换器，其中包含一个计算模块，允许对 ADC 转换结果进行数学运算和阈值测试。这些计算模式包括：

- 基本模式
- 累加模式
- 平均模式
- 突发平均模式
- 低通滤波器（Low-Pass Filter，LPF）模式

基本模式可以视为一种传统模式，其操作与标准 ADC 类似。ADC 将执行转换，但不会对结果进行滤波或求平均值。每次转换后都会执行阈值比较。

累加模式允许将多个转换结果累积或相加。每发生一个 ADC 触发事件之后（通过软件或外部源将 GO 位置 1），都会将转换结果添加到 ADC 累加器（ADACC）寄存器中。然后，ADC 计数器（ADCNT）寄存器递增 1，将新的累加器结果右移（右移位数为 ADC 重复计数（ADRPT）寄存器的值）（即将新的累加器结果除以 ADC 重复计数（ADRPT）寄存器的值）。随后，将右移的累加器结果存储在 ADC 滤波器（ADFLTR）寄存器中，并对 ADFLTR 结果执行阈值比较。

平均模式的工作方式与累加模式类似。但是，当累积采样数与 ADRPT 的值匹配时，将对 ADFLTR 值执行阈值比较。在发生下一个 ADC 触发事件时，累加器清零。

突发平均模式的工作方式与平均模式非常相似，只是在突发平均模式下，单个 ADC 触发事件会导致 ADC 累积采样，直到采样数与 ADRPT 值匹配为止。当发生 ADC 触发事件时，累加器清零，GO 位将保持置 1 状态，直到累积采样数达到所需数量为止。一旦 ADCNT 与 ADRPT 匹配，硬件会将 GO 位清零，累加器右移（右移位数为 ADRPT 的值），并对新的 ADFLTR 值执行阈值比较。

低通滤波器模式的工作方式与平均模式类似，直到 ADCNT 等于 ADRPT，然后硬件对 ADFLTR 值执行阈值比较。该模式不会在下一个 ADC 触发事件时将累加器清零，而是将新的转换结果添加到当前累加器值中，然后向右移位，并执行另一个阈值测试。只要有新的触发事件发生，或直至观察到超出阈值之前，滤波过程都会持续进行。

本项目将使用突发平均模式来监视热敏电阻。将 ADRPT 设置为每次发送触发事件时取 32 个采样的平均值。 例 1-1 显示了用于该项目的 ADCC 初始化程序。

例 1-1. ADCC 初始化代码

```
void ADCC_Initialize(void)
{
    ADLTHL = 0x32;                  // 下限阈值设为 50（0x0032）
    ADLTHH = 0x00;
    ADUTHL = 0xA0;                  // 上限阈值设为 4000（0x0FA0）
    ADUTHH = 0x0F;
    ADSTPTL = 0x00;                 // 设定点置 0
    ADSTPTH = 0x00;
    ADACCU = 0x00;
    ADRPT = 0x20;                   // 设置为 32 个采样
    ADPCH = 0x00;
    ADCON1 = 0x00;                  // DSEN 禁止；
    ADCON2 = 0x53;                  // CRS 5；突发平均模式
    ADCON3 = 0x57;                  // CALC 滤波值与设定点比较
    ADSTAT = 0x00;
    ADREF = 0x00;                   // NREF VSS; PREF VDD
    ADACT = 0x05;
    ADCLK = 0x3F;                   // CS FOSC/128;
    ADCON0 = 0x84;                  // FM 右移；ON 使能；CS FOSC/ADCLK

    PIR2bits.ADTIF = 0;             // 清除 ADC 阈值中断标志
    PIE2bits.ADTIE = 1;             // 允许 ADCC 阈值中断

    ADCC_SetADTIInterruptHandler(ADCC_DefaultInterruptHandler);
}
```

1.1.2.2 16 位脉宽调制器（PWM）概述

PWM 模块提供多个输出和比较功能。PIC18F57Q43 单片机包含三个 PWM 模块。每个模块提供一个单输出片段，该片段由两个 PWM 输出通道组成。两个输出通道具有独立的占空比，可以通过各自的参数寄存器进行设置，但是共享相同的工作模式。五种主要工作模式包括：

- 左对齐模式
- 右对齐模式
- 中心对齐模式
- 可变对齐模式
- 比较模式

左对齐、右对齐、可变对齐和比较模式的 PWM 周期可以使用公式 1-1 计算。

公式 1-1. PWM 周期

$$PWM\ 周期 = \frac{(PWMxPR + 1)}{f_{PWMxCLK}}$$

中心对齐模式的 PWM 周期可以使用公式 1-2 计算。

公式 1-2. PWM 周期（中心对齐模式）

$$PWM\ 周期 = \frac{(PWMxPR + 1) \times 2}{f_{PWMxCLK}}$$

所有模式的 PWM 频率可以使用公式 1-3 计算。

公式 1-3. PWM 频率

$$f_{PWM} = \frac{1}{PWM\ 周期}$$

左对齐、右对齐和中心对齐模式的 PWM 占空比可以使用公式 1-4 计算。

公式 1-4. PWM 占空比

$$PWM\ 占空比 = \frac{PWMxS1Px}{\quad} \times 100$$

Filename: PWM Left Aligned.vsdx
Title:
Last Edit: 10/10/2019
First Used:
Notes:

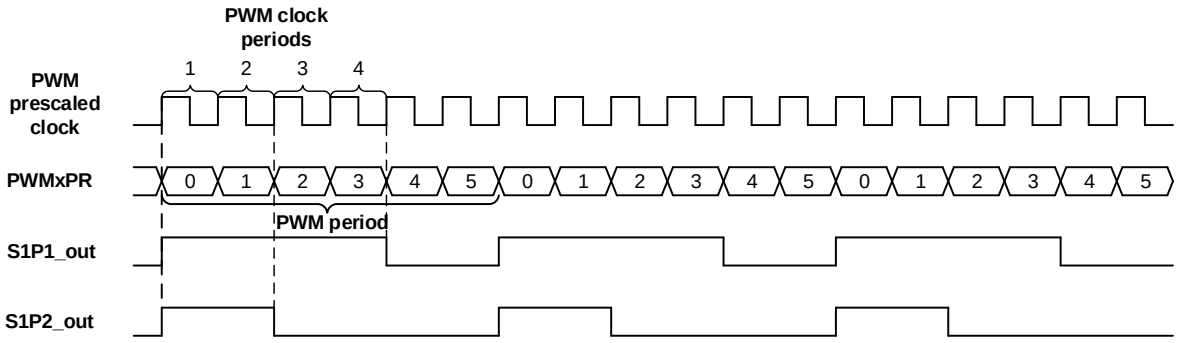
注

1.1.2.2.1 左

在 有效状态，并在指定的预分频 PWM 时钟周期数（由每个输出各自的参数寄存器 PWMxS1P1 或 PWMxS1P2 指定）内保持有效状态，然后在 PWM 周期的剩余时间内转换为无效状态。 图 1-1 显示了在左对齐模式下工作的 PWM 输出。

图 1-1. PWM 左对齐模式

Rev. PWM left A
10/10/2019



注：

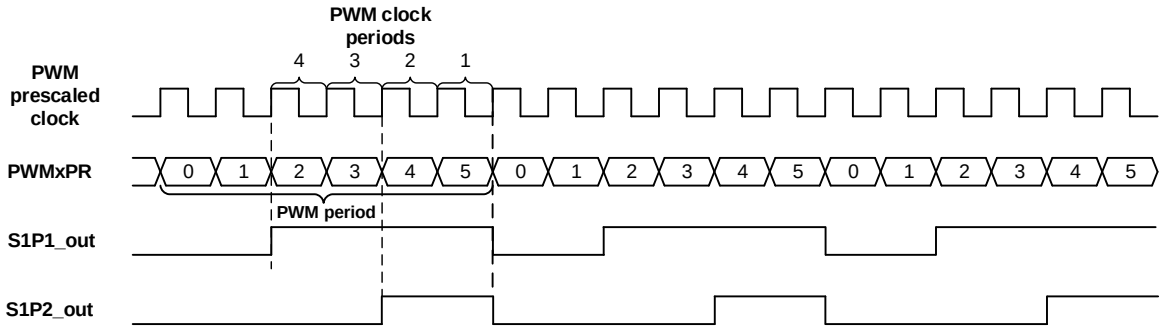
Filename: PWM Right Aligned.vsdx
- Title:
Last Edit: 10/10/2019
- First Used:
Notes:
-

1.1.2.2.2 右对齐模式

在右对齐模式下，占空比的有效部分位于 PWM 周期结束时。输出在每个周期开始时处于无效状态。然后，在 PWM 周期结束前的预分频 PWM 时钟周期数内进入有效状态，该周期数由每个输出的参数寄存器指定。 图 1-2 显示了在右对齐模式下工作的 PWM 输出。

Rev. PWM Right
10/10/2019

图 1-2. 右对齐模式



注：

- PWMxPR = 5
- PWMxS1P1 = 4
- PWMxS1P2 = 2

Filename: PWM Center Aligned.vsdx
Title:
Last Edit: 10/10/2019
First Used:
Notes:

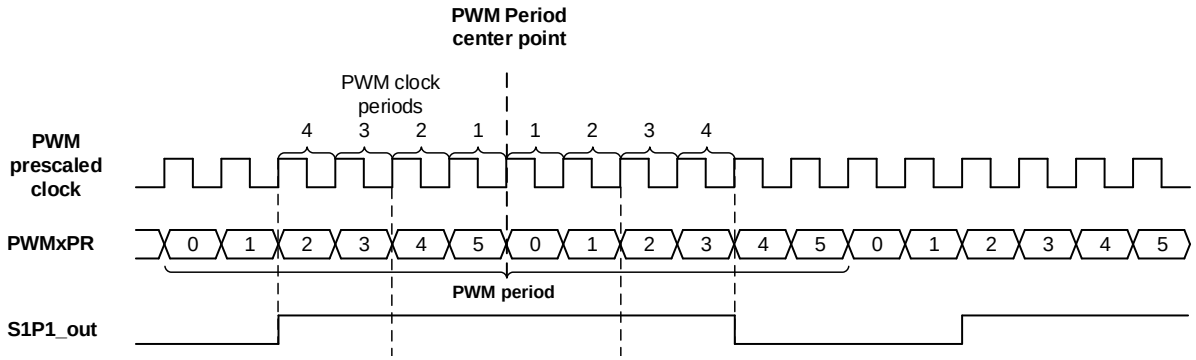
1.1.2.2.3 中心

在中 期是其他模式的两倍，如公式

每个 M 周期中心并进入有效状态，之后再经过相同长度的预分频 PWM 时钟周期后，进入无效状态。 图 1-3 显示了在中心对齐模式下工作的 PWM 输出。

图 1-3. PWM 中心对齐模式

Rev. PWM Center 10/10/2019



Filename: PWM Variable Aligned.vsdx
Title:
Last Edit: 10/10/2019
First Used:
Notes:

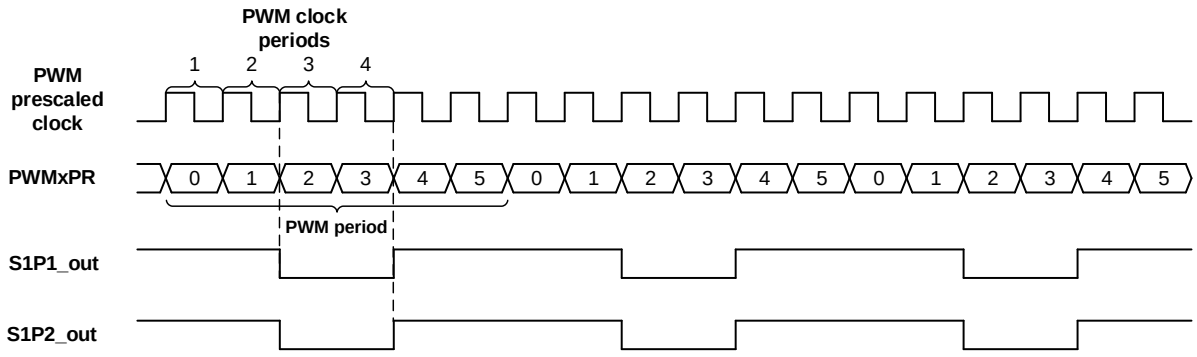
- PWMxPR = 5
- PWMxS1P1 = 4
- PWMxS1P2 = 2

Rev. PWM Variab 10/10/2019

1.1.2.2.4 可变对齐模式

在可变对齐模式下，占空比的有效部分在参数 1 的值与预分频 PWM 时钟周期数匹配时开始，并在参数 2 的值与预分频的 PWM 时钟周期数匹配时结束。在此模式下，由于两个参数值都用于相同的占空比，因此两个输出完全相同。 图 1-4 显示了在可变对齐方式下工作的输出。

图 1-4. PWM 可变对齐模式



注：

- PWMxPR = 5
- PWMxS1P1 = 4
- PWMxS1P2 = 2

Filename: PWM Compare - Pulsed.vsdx
Title:
Last Edit: 10/10/2019
First Used:
Notes:

1.1.2.2.5

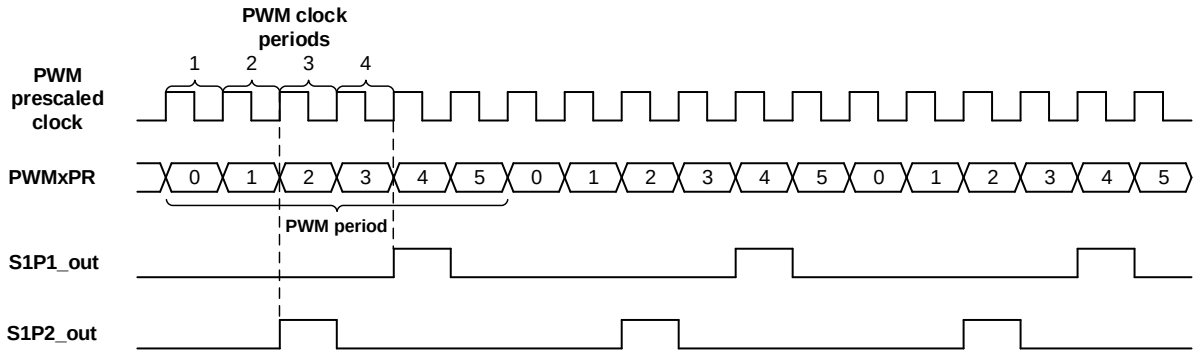
预分频 PWM 时钟周期数与

脉冲比较模式

在脉冲比较模式下，输出开始于无效状态。当预分频 PWM 时钟周期数与每个输出的参数值匹配时，将产生脉冲输并进入有效状态，持续一个预分频 PWM 时钟周期，然后返回至无效状态，在 PWM 周期的剩余时间内保持不变。 图 1-5 显示了在脉冲比较模式下工作的输出。

Rev. PWM Compar 10/10/2019

图 1-5. PWM 脉冲比较模式



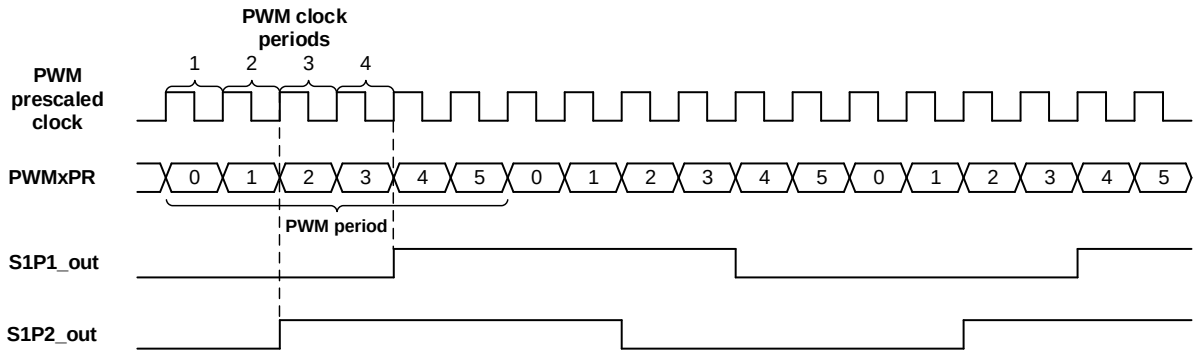
Filename: PWM Compare - Toggle.vsdx
Title:
Last Edit: 10/10/2019
First Used:
Notes:

翻转比较模式

在翻转比较模式下，占空比是交替的完整 PWM 周期。当预分频 PWM 时钟周期数与相应的参数值匹配时，每个输出都将进入有效状态，并在下一个 PWM 周期的相同匹配点进入无效状态。 图 1-6 显示了在翻转比较模式下工作的输出。

Rev. PWM Compar 10/10/2019

图 1-6. PWM 翻转比较模式



注：

- PWMxPR = 5
- PWMxS1P1 = 4
- PWMxS1P2 = 2

本项目采用左对齐模式来控制每台风扇的速度。随着热敏电阻温度的变化，每个输出的占空比都会根据需要进行调整，以适应冷却需求。 例 1-2 显示了 PWM 初始化序列。

注意： PWM 占空比根据温度和转速计输入来调整。实际占空比调整在 CCP 中断服务程序中执行，具体内容请参见捕捉/比较/ PWM（CCP）章节。

例 1-2. PWM 初始化

```
void PWM1_Initialize(void)
{
    PWM1ERS = 0x00;              // PWMERS 外部复位禁止
    PWM1CLK = 0x02;              // PWMCLK FOSC
    PWM1LDS = 0x00;              // PWMLDS 自动加载禁止
    PWM1PRL = 0x00;              // 0x0A00=25 kHz PWM freq @ 64 MHz FOSC
    PWM1PRH = 0x0A;
    PWM1CPRE = 0x00;             // PWMCPRE 无预分频
    PWM1PIPOS = 0x00;            // PWMPIPOS 无后分频
    PWM1GIR = 0x00;              // 清除 PWM 通用中断标志
    PWM1GIE = 0x00;              // PWMS1P2IE 禁止；PWMS1P1IE 禁止
    PWM1S1CFG = 0x00;            // PWM 左对齐模式
    PWM1S1P1L = 0x00;
    PWM1S1P1H = 0x05;
    PWM1S1P2L = 0x00;
    PWM1S1P2H = 0x05;
    PWM1CON = 0x80;              // PWMEN 使能
}
```

1.1.2.3 捕捉/比较/PWM（CCP）模块概述

捕捉/比较/PWM（CCP）模块是用于测量时间、控制某些事件以及产生 PWM 信号的外设。在捕捉模式下，可通过外设来测量事件的时间。比较模式则允许用户在超过预定时间后触发外部事件。PWM 模式用于产生不同频率和占空比的 PWM 信号。

1.1.2.3.1 捕捉模式

捕捉模式使用 16 位奇数定时器资源，例如 Timer1。当捕捉源上发生边沿事件时，16 位 CCPRx 寄存器将捕捉并存储 16 位定时器值。触发事件可能来自内部源（例如 CLC 输出），也可能来自通过 CCPxPPS 输入引脚选择的外部源。

捕捉事件可以由以下情形触发：

- CCP 输入的每个下降沿
- CCP 输入的每个上升沿
- 每 4 个 CCP 输入的上升沿
- 每 16 个 CCP 输入的上升沿
- CCP 输入的每个边沿（上升沿和下降沿）

当发生捕捉事件时，CCP 中断标志（CCPxIF）置 1，并且必须用软件将其清零。

重要： 如果在读取 CCPRx 寄存器中的值之前发生了另一个捕捉事件，则将使用新值覆盖旧值。如果该事件发生在 2 字节读取期间，则 CCPRxH 字节数据和 CCPRxL 字节数据将来自不同的事件。为了确保数据的有效性，建议在读取 CCPRx 寄存器对时禁止该模块，或者读取两次寄存器对。

1.1.2.3.2 比较模式

与捕捉模式类似，比较模式使用 16 位奇数定时器资源，例如 Timer1。载入 CCPRx 寄存器的 16 位值将不断与定时器寄存器的 16 位值进行比较。当发生匹配时，根据 CCP 模式选择（MODE）位的配置，将会发生以下事件之一：

- 翻转 CCP 输出，并清零相关的定时器寄存器
- 翻转 CCP 输出，不清零相关的定时器寄存器
- 置 1 CCP 输出
- 清零 CCP 输出

- 产生脉冲输出
- 产生脉冲输出，并清零相关的定时器寄存器

此外，当发生匹配时，CCP 中断标志（CCPxIF）置 1，该标志也可用于触发 ADC 转换。必须用软件将 CCPxIF 位清零。

1.1.2.3.3 PWM 模式

CCP 模块中的 PWM 充当 10 位脉宽调制器，它使用 8 位偶数定时器资源，例如 Timer2。通常，PWM 通过在完全 ON（打开）和完全 OFF（关闭）状态之间快速切换来控制负载的电源。PWM 周期的高电平部分（脉冲宽度）被视为 ON 或有效状态，低电平部分被视为 OFF 或无效状态。脉冲宽度可以随时间变化，并用步阶来定义。随着步阶数增加，脉冲宽度增加，提供给负载的功率就变大。随着步阶数减小，脉冲宽度减小，提供给负载的功率就变小。PWM 周期定义为一个完整周期的持续时间。

PWM 分辨率定义了单个 PWM 周期中可以出现的最大步阶数。更高的分辨率可以更精确地控制脉冲宽度时间。在 10 位 PWM 中，根据占空比的配置，单个 PWM 周期内最多可以有 1024 个步阶。

PWM 占空比是脉冲宽度与整个 PWM 周期的比率，以百分比表示，其中 0％视为完全关闭，而 100％视为完全打开。较低的占空比意味着提供给负载的功率较小，而较高的占空比意味着提供给负载的功率较大。

在此项目中，CCP 在捕获模式下使用 Timer1 作为捕捉时基来测量风扇转速计的输出。CCP 还被配置为捕捉转速计输出的每个上升沿。 例 1-3 显示了 CCP 初始化代码。

例 1-3. CCP 捕捉模式初始化

```
void CCP1_Initialize(void)
{
    CCP1CON = 0x85;                          // 模式为每个上升沿
    CCP1CAP = 0x00;                          // CCP1CTS CCP1 引脚
    CCPR1H = 0x00;
    CCPR1L = 0x00;
    CCPTMRS0bits.C1TSEL = 0x1;               // 选择 Timer 1
    PIR3bits.CCP1IF = 0;                     // 清除 CCP1IF
    PIE3bits.CCP1IE = 1;                     // 允许 CCP1 中断
}
```

1.1.2.4 通用异步收发器（UART）模块概述

UART 模块是一款串行通信外设。UART 包含执行输入或输出串行数据传输所需的所有时钟发生器、移位寄存器和数据缓冲区，与器件程序执行无关。UART 可以配置为全双工异步系统或几种自动协议中的一种（例如 LIN）。UART 也可以配置为半双工模式（只发送模式或只接收模式）。

UART 具有五种异步模式：

- 7 位
- 8 位
- 8 位，第 9 位为偶校验
- 8 位，第 9 位为奇校验
- 8 位，第 9 位为地址指示器

UART 使用标准的不归零（Non-Return-to-Zero，NRZ）格式发送和接收数据。NRZ 表示连续发送的具有相同值的数据位保持输出电平不变，不会在每个位发送之间返回中性状态。NRZ 的实现使用两个逻辑电平：

- 电压输出高电平（V_{OH}）“标记”状态，代表逻辑“1”数据位
- 电压输出低电平（V_{OH}）“空格”状态，代表逻辑“0”数据位

NRZ 发送端口在“标记”状态下空闲。每个字符传输都包括一个启动位，然后是七个或八个数据位，一个可选的奇偶校验位或地址位，并且始终以一个或多个停止位结束。启动位始终为“空格”，停止位始终为“标记”。最常用的格式是无奇偶校验的 8 位。每个发送位保持不变的时间为 1/波特率。

在所有异步模式下，UART 首先发送和接收最低有效位（Least Significant bit，LSb）。UART 的发送器和接收器完全独立，但是两者使用相同的数据格式和波特率。

在本项目中，UART 仅用作发送器。将温度和风扇速度发送到 PC 终端，这样用户就可以监视风扇的运行情况。 例 1-4 显示了 UART 初始化和发送程序。此外，还可以使用“printf”库函数以字符串格式来快速传输字符。

例 1-4. UART 初始化和发送程序

```
void UART1_Initialize(void)
{
    U1CON0 = 0xA0;                                  // BRGS HS; TXEN 使能
    U1CON1 = 0x80;                                  // ON 使能;
    U1CON2 = 0x00;                                  // TXPOL 不反相; 1 个停止位
    U1BRGL = 0xC7;                                  // 9600 BR @ 64 MHz FOSC
    U1BRGH = 0x06;
    U1FIFO = 0x00;
    U1UIR = 0x00;                                   // 自动波特率未使能
    U1ERRIR = 0x00;
    U1ERRIE = 0x00;
    UART1_SetFramingErrorHandler(UART1_DefaultFramingErrorHandler);
    UART1_SetOverrunErrorHandler(UART1_DefaultOverrunErrorHandler);
    UART1_SetErrorHandler(UART1_DefaultErrorHandler);
    uart1RxLastError.status = 0;
}

void putch(char txData)                             // 供“printf”函数库使用 library
{
    UART1_Write(txData);
}

void UART1_Write(uint8_t txData)
{
    while(0 == PIR4bits.U1TXIF)                     // 等待清除 TX 缓冲区
    {
    }
    U1TXB = txData;                                 // 将数据字节写入 UART
}
```

1.2 工作原理

本应用主要执行三项任务：

- 采集温度值
- 捕捉实际风扇速度并调整 PWM 输出
- 发送温度值和实际风扇速度

1.2.1 采集温度值

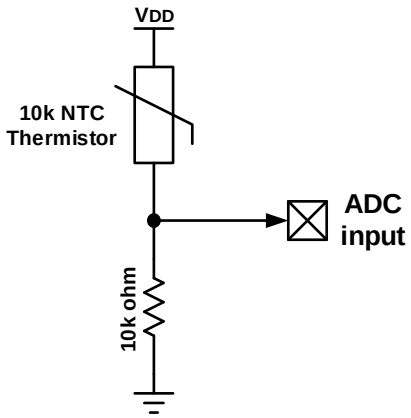
我们使用 10 kΩ 负温度系数（Negative Temperature Coefficient，NTC）热敏电阻进行温度采集。NTC 热敏电阻是一种非线性电阻，其电阻会随着温度的升高而降低。对于 ADC 电压而言，随着温度升高，热敏电阻的阻值减小，ADC 输入的电压就会变大。 表 1-1 显示了温度与 ADC 读数之间的关系。这些值是将热敏电阻置于温控环境中而获得的。将热敏电阻配置为分压器电路的一部分（见图 1-7），并且将分压器电路的输出馈入 ADC 输入。

表 1-1. 温度与 ADFLTR 值

温度（°C）	ADFLTR 值
0	918
20	1800
25	2028
30	2258
60	3280
65	3391
70	3488

Filename: Thermistor Circuit.vsdx
Title:
Last Edit: 10/14/2019
First Used:
Notes:

图 1-7. 热敏电阻分压器电路



项目软件配置为每秒钟读取一次热敏电阻输入。两次测量间延时为一秒，这样选择的目的仅仅是为了演示。大多数应用不会出现如此快速的温度大幅变化，因此可以选择更长的时间间隔进行温度测量。更长的测量时间间隔也有助于防止因风扇速度不断变化而产生明显的噪声。

在此应用中，将 ADC 配置为突发平均模式，该模式将对 ADC 输入进行 32 个快速突发采样，然后对这些样本取平均值。突发平均模式简单易用，软件只需设置一次 GO 位即可开始采样，并且当完成所有 32 个采样时，硬件会自动将 GO 位清零。这样可以节省指令时间，使应用更高效。

获取所有 32 个采样并取其平均值后，将平均值加载到 ADFLTR 寄存器中并执行阈值测试。在这种情况下，阈值测试将计算 ADFLTR 值与用户定义的设定点之间的误差或差异。然后，硬件将误差值与用户定义的两个阈值（上限阈值和下限阈值）进行比较。在此应用中，根据热敏电阻两种可能的故障条件来选择阈值。下限阈值设置为 ADC 值“50”，上限阈值设置为 ADC 值“4000”。下限阈值表示热敏电阻开路故障条件，此时热敏电阻的阻值理论上为无限大，可阻止任何电压通过热敏电阻。而 ADC 输入会通过分压器电路的下拉电阻拉至地。上限阈值表示热敏电阻短路故障条件，此时热敏电阻的阻值为零，因此所有电压都会施加到 ADC 的输入上。

阈值测试完成后，将 ADC 阈值中断标志（ADTIF）位置 1。可以将 ADC 阈值中断配置为在不同条件下触发，例如超出下限阈值或超出上限阈值。在此项目中，无论阈值测试的结果如何，都将其设置为始终中断。此方法允许软件执行以下操作：

- 手动检查 ADC 模块的大于上限阈值（UTHR）标志或小于下限阈值（LTHR）标志。
- 根据 ADC 输入计算所需的风扇速度。
- 计算热敏电阻温度。

如果 ADC 误差计算值超出相应的 ADC 上限阈值（ADUTH）寄存器和 ADC 下限阈值（ADLTH）寄存器中的上限阈值或下限阈值，则将 UTHR 和 LTHR 状态标志置 1。如果这两个状态标志中的任何一个标志置 1，则指示风扇全速运转，并且将向 PC 终端程序发送一条消息，说明发生了热敏电阻故障。

如果两个状态标志均未置 1，则软件会使用线性方程（可对风扇速度进行线性化）计算所需的风扇速度。 公式 1-5 显示了标准线性方程和以风扇速度和 ADC 值表示的等效线性方程。

公式 1-5. 线性方程

$$y = mx + b$$

其中：

y = 风扇速度

$$m = \frac{(\text{风扇最高速度} - \text{风扇最低速度})}{(ADFLTR\ @\ 70C - ADFLTR\ @\ 20C)}$$

$$x = \text{当前}\ ADFLTR$$

$$b = \text{风扇最高速度} - (m \times ADFLTR@70C)$$

计算出所需的风扇速度后，便可计算出热敏电阻温度值。此计算可使用简化的 β（贝塔）参数方程（公式 1-6）完成，该方程可由 Steinhart-Hart 方程导出。简化的 β 参数方程使用的参数项（例如 25°C 时的热敏电阻值）很容易获得，能够降低软件所需执行的计算的复杂性。为了进一步简化软件计算，已预先手工计算了一些数学运算量较大的变量，并将其列为软件常数。

公式 1-6. 简化的 β 参数方程

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B}\ln\left(\frac{R}{R_0}\right)$$

其中：

T = *开尔文温度（K）*

T_0 = *298.15 K（室温 25°C）*

B = *热敏电阻系数（在本示例中为 3950）*

R = *热敏电阻测量值*（ADC 电压值（V）必须转换为电阻值）

R_0 = *热敏电阻在 25°C 时的额定电阻值*（在本示例中为 10 kΩ）

例 1-5 显示了中断服务程序，该程序用来监视阈值状态标志、计算所需的风扇速度和执行热敏电阻温度计算，并列出了计算中使用的预计算常数。对于此项目，在温度为 20°C 或更低时，风扇设置为以 1000 RPM 的固定最低速度旋转，在温度为 70°C 或更高时（或在热敏电阻故障条件下），风扇以 8000 RPM 的固定最高速度旋转，在 20°C 和 70°C 之间速度呈线性变化。

例 1-5. ADC 阈值 ISR

```
#define MAXADC          4095.000      // 12 位 ADCC
#define K               273.1500      // 开尔文常数
#define LN              0.4343        // Log(e)等效值

#define MIN_THERM_ADC   1800          // 20C 时的 ADFLTR 值
#define MAX_THERM_ADC   3488          // 70C 时的 ADFLTR 值
#define MIN_SPD         1000          // 风扇最低速度
#define MAX_SPD         8000          // 风扇最高速度
#define SLOPE_M         4.15
```

```
            // = ((MAX_SPD - MIN_SPD) / (MAX_THERM_ADC - MIN_THERM_ADC))
#define B_OFFSET        -6475       //(MAX_SPD-(SLOPE_M * MAX_THERM_ADC))

void ADCC_DefaultInterruptHandler(void)
{
    float T0 = 0.00335;                         1 × 298.15
    float B = 0.00025;                          // 1 / 3950（β 值）

    adc_val = ADCC_GetFilterValue();            // 获取热敏电阻值
    if(adc_val <= MIN_THERM_ADC)                // 温度是否达到或低于 20℃？
    {
        if(ADSTATbits.LTHR == 0)                // 未超出下限阈值
        {
            y_therm = MIN_SPD;                  // 将风扇设为最低速度
            therm_short = 0;
            therm_open = 0;
        }
        else                                    // 超出下限阈值
        {
            y_therm = MAX_SPD;                  // 将风扇设为最高速度
            therm_open = 1;                     // 热敏电阻开路故障
        }
    }
    else if(adc_val >= MAX_THERM_ADC)           // 温度是否达到或超出 70℃？
    {
        y_therm = MAX_SPD;                      // 将风扇设为最高速度
        if(ADSTATbits.UTHR == 1)                // 检查热敏电阻是否短路
        {
            therm_short = 1;                    // 热敏电阻短路故障
            therm_open = 0;
        }
    }
    else                                         // 20C < 温度 < 70C
    {   // 按照 y = mx + b 线性方程计算速度
        y_therm = SLOPE_M * adc_val + B_OFFSET;
        therm_short = 0;
        therm_open = 0;
    }

    temperature = (MAXADC/(float)adc_val)-1.00000;  // 输入电压的 R 等效值
    temperature = log10f(temperature);              // 以 10 为底取温度的对数值
    temperature = temperature / LN;                 // 将 log(x)转换为 ln(x)
    temperature = B * temperature;                  // 乘以常数 B
    temperature += T0;                              // 加上常数 T0
    temperature = 1 / temperature;                  // 转换为开尔文温度（K）
    temperature -= K;                               // 减去 K = 摄氏温度 C

    PIR2bits.ADTIF = 0;
}
```

1.2.2 捕捉风扇速度并调整 PWM

在 PWM 初始化期间，PWM 周期设置为 25 kHz 频率和 50%占空比。选择此 PWM 频率是为了匹配风扇的预期输入频率，而选择此占空比只是为了使风扇开始旋转，并在执行首次温度计算时提供一个转速计输出信号。

获取温度值并计算出预期速度后，CCP 的捕捉模块将开始测量风扇转速计的输出信号。San Ace 40 无刷直流（BLDC）风扇每转一圈输出两个脉冲的转速计信号（有关 BLDC 风扇运行的简化说明，参见 BLDC 基础知识）。CCP 配置为捕捉转速计信号的每个上升沿，使用 Timer1 作为时基以有效测量每个转速计周期。一旦模块硬件检测到上升沿，CCP 硬件就会记录定时器值，并产生中断事件，在该事件中将计算实际风扇速度并进行风扇速度调整。

CCP 的捕捉中断服务程序开始时，首先要获取上次存储的 CCPR 寄存器值与新的 CCPR 寄存器值之间的差。由于本项目中使用的三个 CCP 模块共享 Timer1 资源，因此 Timer1 以自由运行模式工作，并且在两次捕捉事件之间不会被清除或复位。这意味着每个 CCP 模块必须保存先前捕捉到的值，并将其与新捕捉到的值进行比较。这两个值之间的差就是转速计周期（以定时器计数为单位），然后必须将其转换为转速（以 RPM 为单位）。 公式 1-7 显示了将定时器计数值转换为 RPM 所需的计算，并将其应用于风扇 1。

公式 1-7. 将捕捉到的值转换为 RPM

$$fan1_spd = \frac{TIMER_CONST}{fan1_tmr_val}$$

其中：

$1_spd =$ *风扇1 的测量速度（以RPM为单位）*

$$TIMER_CONST = \frac{Timer1\ 时钟}{Timer1\ 预分频比} \times \frac{60}{磁极对数}$$

$1_tmr_val =$ *当前捕捉到的值 − 先前捕捉到的值*

转换后的转速计信号代表实际风扇速度，它可能与所需风扇速度匹配，也可能不匹配。在计算出实际风扇速度之后，将其与所需风扇速度进行比较。如果实际风扇速度小于所需速度，则存储在 PWMxS1Py 参数寄存器中的当前 PWM 占空比值将递增 1；如果实际速度大于所需速度，则存储在参数寄存器中的值将递减 1。每次捕捉到中断都会继续执行此过程：计算实际风扇速度，并将其与所需速度进行比较，然后增大或减小占空比，直到两者匹配为止。

例 1-6 显示了 CCP1 捕捉到的 ISR，它监视并校正风扇 1 的速度。CCP2 和 CCP3 针对各自的风扇使用相同的程序。

例 1-6. CCP1 捕捉到的 ISR

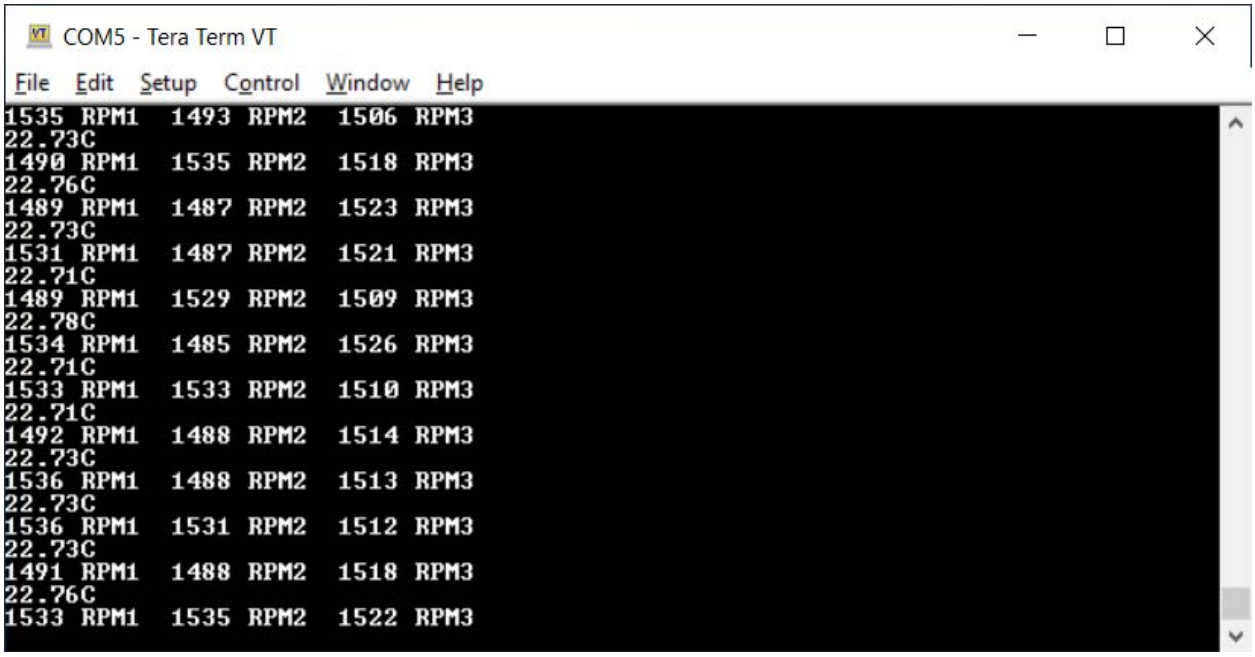
```
#define TIMER_CONST    3750000
// = (TMR1 时钟/TMR1 预分频比) * (60/磁极对数)
// = (1 MHz / 8) * (60 / 2)

void CCP1_CaptureISR(void)
{
    CCP1_PERIOD_REG_T module;
    module.ccpr1l = CCPR1L;                                 // 复制捕捉到的值
    module.ccpr1h = CCPR1H;
    prevCapt1 = currentCapt1;                               // 复制上次捕捉到的值
    currentCapt1 = module.ccpr1_16Bit;                      // 获取新捕捉到的值

    if(currentCapt1 > prevCapt1)
    {
        fan1_tmr_val = currentCapt1 - prevCapt1;            // 找到转速计周期
    }
    else                                                    // Timer1 计满返回
    {
        fan1_tmr_val=(0xFFFF-prevCapt1)+currentCapt1;
    }

    fan1_spd = TIMER_CONST / fan1_tmr_val;                   // 计算实际速度

    if(fan1_spd < y_therm)                                   // 风扇 1 太慢?
    {
        PWM1S1P1 += 1;                                       // 增大占空比
        PWM1CONbits.LD = 1;                                  // 锁存新的 PWM 值
    }
    if(fan1_spd > y_therm)                                   // 风扇 1 太快?
    {
        PWM1S1P1 -= 1;                                       // 减小占空比
        PWM1CONbits.LD = 1;                                  // 锁存新的 PWM 值
    }
    PIR3bits.CCP1IF = 0;
}
```

1.2.3 发送温度值和实际风扇速度

最后一项任务由软件执行，即发送热敏电阻的温度值和每个风扇的实际速度（以 RPM 为单位）。数据发送由连接到 PC 终端程序的 UART 处理。这样用户就可以监视温度和各个风扇的速度。

在本项目中，UART 配置为 9600 位/秒（bits per second，bps）波特率，以异步 8 位传输模式工作，没有奇偶校验位，有一个停止位。PC 终端程序必须采用相同的方式配置。 图 1-8 显示项目工作期间的 PC 终端窗口。 例 1-7 显示主 while 循环，其中 UART 的“printf”语句用于传送温度和风扇速度。

图 1-8. PC 终端输出窗口



例 1-7. 项目主 `while()`循环

```
while (1)
{
    ADCC_StartConversion(Thermistor);                       // 读取热敏电阻

    if(therm_short == 1)
    {
         printf("Thermistor shorted failure!\r\n");
    }
    if(therm_open == 1)
    {
         printf("Thermistor open failure!\r\n");
    }
    if((therm_open == 0) && (therm_short == 0))
    {
         printf("%1.2fC \r\n", temperature);                // 显示温度
    }
    // 显示风扇速度
    printf("%d RPM1  %d RPM2  %d RPM3\r\n",fan1_spd, fan2_spd, fan3_spd);

    __delay_ms(1000);                                       // 1 秒延迟
}
```

2. BLDC 基础知识

无刷直流（BLDC）风扇使用电子开关控制流经其电机的电流量，以使转子转动。BLDC 风扇依靠转子位置反馈来控制其开关的开或关，从而允许电机产生适当的转矩来使转子旋转。转子位置反馈可能来自位置传感器，也可能来自检测反电动势（Electromotive Force，EMF）。BLDC 电机分为三种：单相、两相和三相。

BLDC 风扇主要由以下元件组成：

- 转子
- 定子（电机）
- 框架（外壳）
- 电机驱动电路

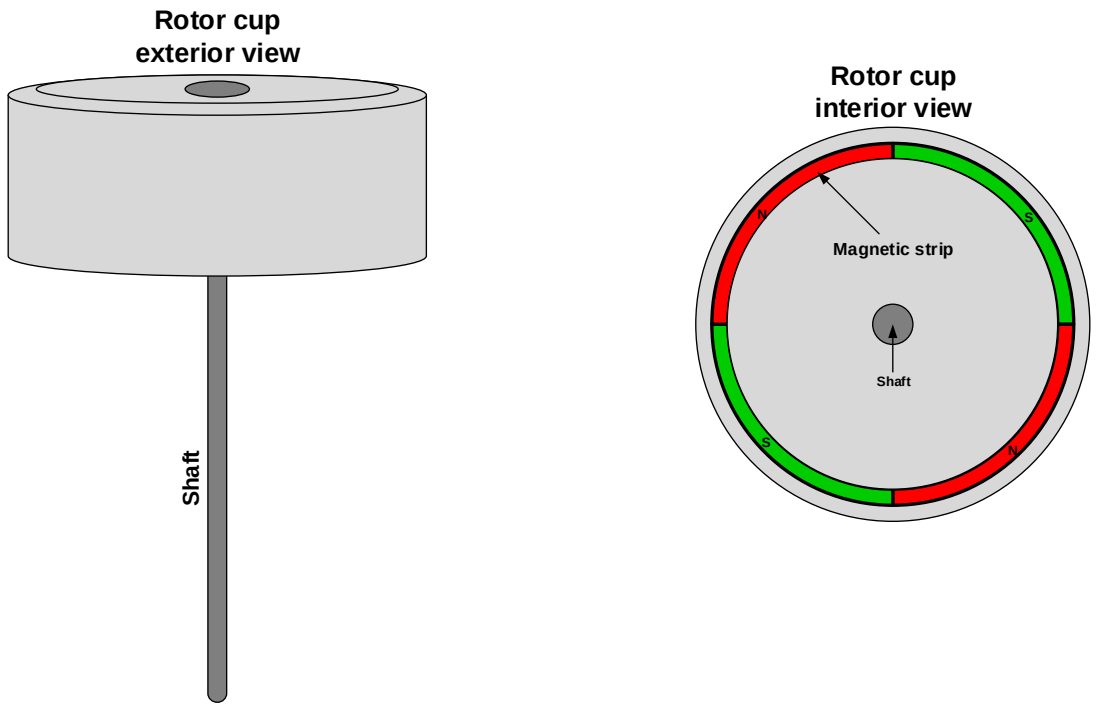
在本项目中，风扇使用单相电机，依靠霍尔效应传感器监视转子位置。风扇为轴流式风扇，这意味着气流与转子轴平行。

2.1

Filename: Rotor Cup.vsdx
Title:
Last Edit: 10/17/2019
First Used:
Notes:

转子杯的内部装有一

图 2-1. 转子杯



叶轮是一个塑料件，具有从中心延伸的法兰，通常将其胶合或压装至转子杯的外表面，从而形成转子总成（见图 2-2）。法兰的设计是为了满足特定的气流特性，当转子总成旋转时，气流会随着每个法兰的夹角流动。

将叶轮安装到转子杯上之后，必须平衡整个总成。可以使用单平面或双平面平衡方式，具体基于两个因素：

- 转子杯的长径比（不包括轴）
- 转子的转速

如果转子没有正确平衡，可能会发生不必要的振动，进而引起额外的音频噪声，并增加对其他元件（如风扇轴承）的压力。随着时间的推移，这可能会导致风扇元件过早出现故障。

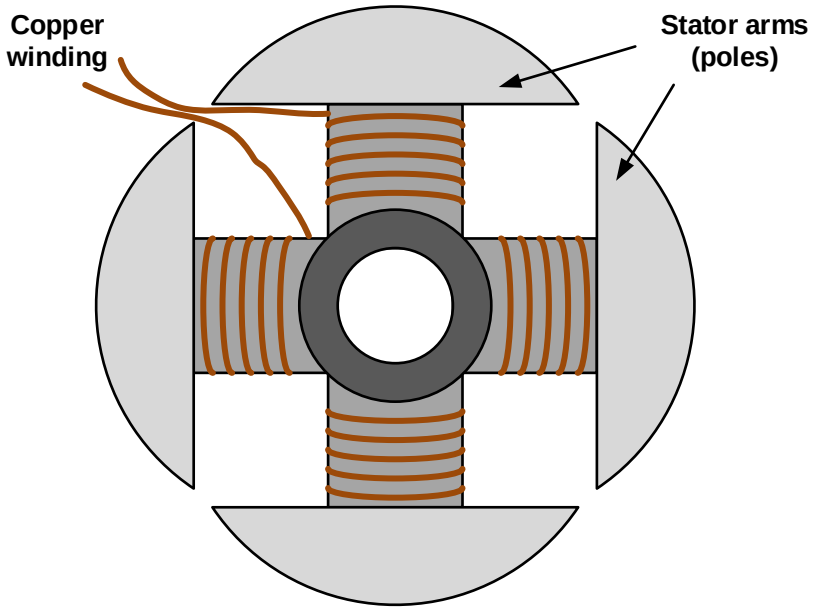
图 2-2 转子总成

2.2 定子

定子（即电机）用于通过磁力推动转子总成沿特定方向运动。定子的铁心由垂直堆叠的多层钢质层压板组成。带触针的塑料绝缘子围绕着层压钢板，从而使定子可以安装到印刷电路板（Printed Circuit Board，PCB）上，同时还可以防止定子绕组与钢板层接触。定子绕组是绕在每个定子臂（磁极）上的漆包铜线。在单相电机中，单股铜线绕在每个定子极上，每个极的绕线互相交替缠绕（极 1 顺时针绕线，极 2 逆时针绕线，以此类推）。绕线时，导线的两端都连接到电机驱动器电路。 图 2-3 显示了一个四极定子，这是一种常见的单相电机配置。

当电流流过定子绕组时，会在每个磁极产生磁场。这些磁场与转子总成的永磁体相互作用，定子磁极和永磁体磁极之间的磁引力将使转子旋转。电机驱动电路讨论了驱动电路的工作方式，并说明了磁场如何使转子运动。

图 2-3. 四极定子电机

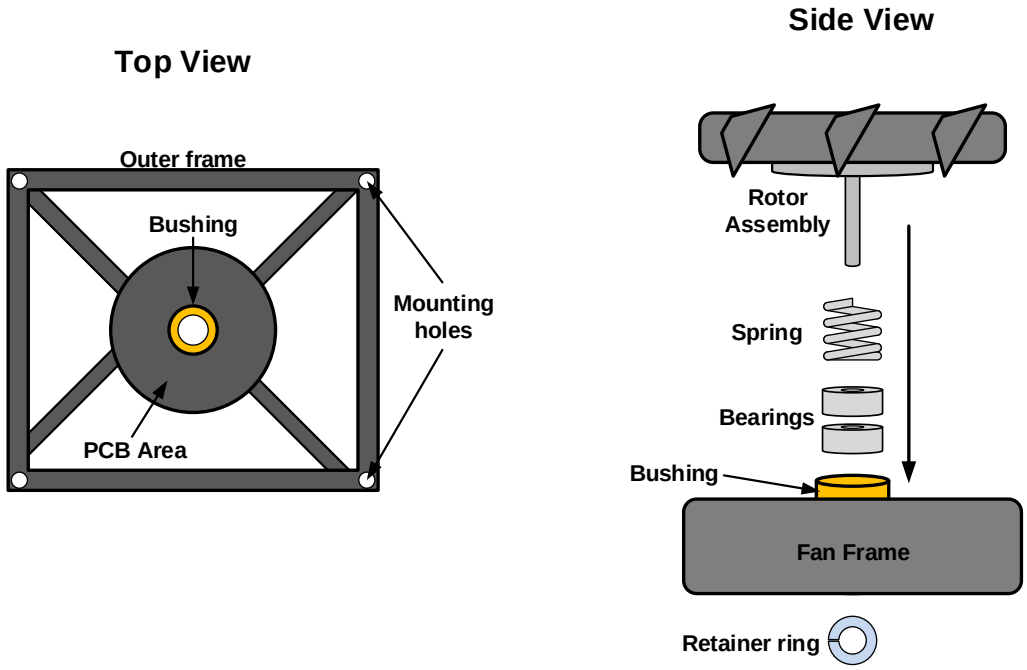


2.3 框架

风扇框架（外壳）通常由塑料制成，但也可能是铝制的。框架外侧有孔，用于将风扇安装到需要冷却的部件（例如计算机主板）上。结构叶片将外部框架与内部区域相连，内部区域中心有一个空心衬套，用于安装 PCB 和定子。中心衬套通常由黄铜制成，可用作定子的散热器。衬套内部是中空的，里面装有一组轴承和一个弹簧，转子轴可以穿过该装置滑动，确保平稳旋转。 图 2-4 显示了典型的轴流风扇框架。

图 2-4. 典型的轴流风扇框架

Rev. Fan Frame
10/18/2019



2.4 电机驱动电路

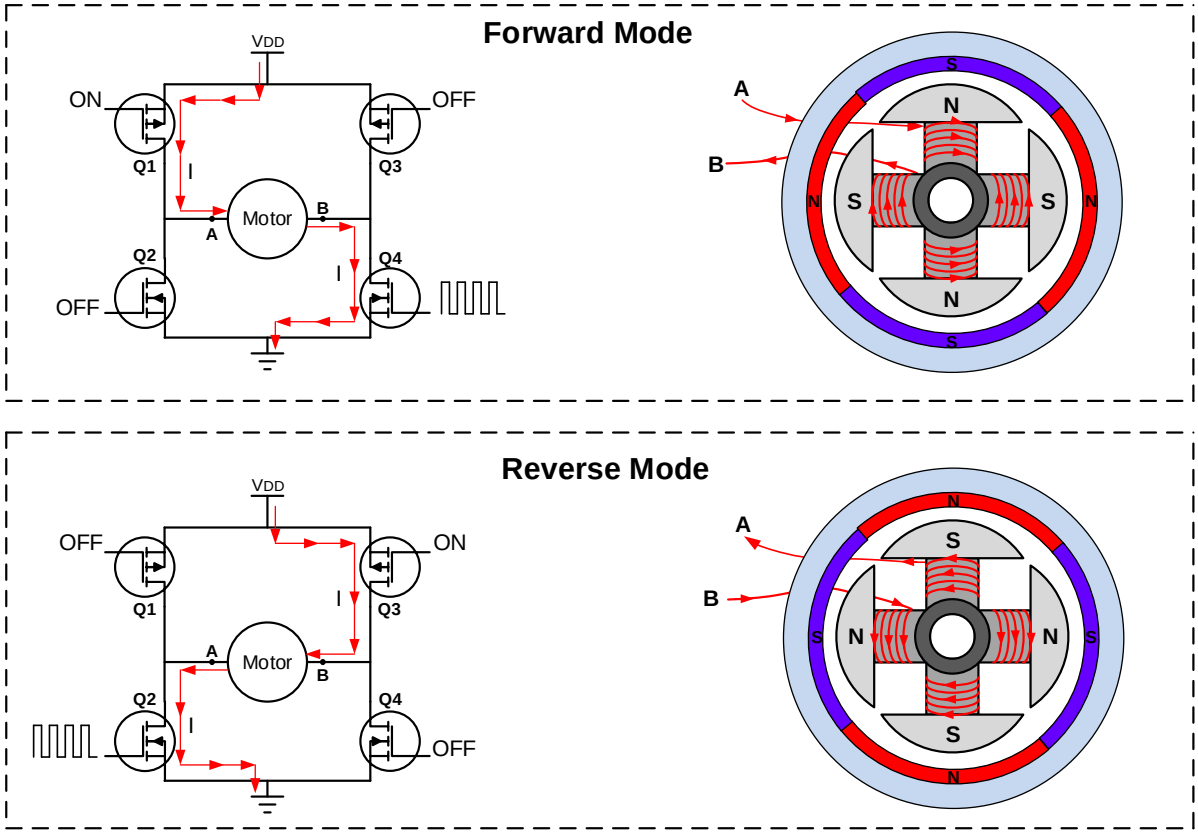
BLDC 的电机驱动电路包含控制转子总成旋转所需的元件。电机驱动电路的主要部分是 H 桥，由开关器件（通常为 MOSFET）和嵌入式控制器（如单片机）组成。对于本项目，我们将以一个简单的基于单片机的电机驱动电路作为参考，该电路具有一个 H 桥，由两个 P 沟道 FET 和两个 N 沟道 FET 以及一个霍尔传感器组成。

配置 P 沟道 FET，使其源极连接至 V_{DD}，漏极连接至电机绕组。N 沟道 FET 的漏极连接至电机绕组，源极接地。

单片机根据霍尔传感器的反馈确定何时打开/关闭或向 H 桥施加 PWM 信号。当霍尔传感器检测到转子总成永磁体的磁场发生变化时，它将根据检测到的磁极（北或南）向单片机发送二进制的“1”或“0”。然后，单片机决定使能或禁止哪些 FET，以及将 PWM 信号施加于哪些 FET。此外，霍尔传感器输出用于驱动转速计信号。

H 桥通过允许电流流过电机绕组来控制转子总成的旋转。 图 2-5 显示了在两种电流模式下电流如何流过 H 桥和电机：正向模式和反向模式。

图 2-5. 电机驱动器 H 桥电路



在正向模式下，Q1 完全打开，Q4 通过 PWM 信号打开和关闭。在正向模式下，Q2 和 Q3 关闭。电流流过 Q1，流入再流出电机绕组，然后经 Q4 至地。由于每个定子臂的绕组交替改变其绕组方向，因此流过每个臂的电流会产生交替的南北磁场。这些磁场吸引转子总成永磁体的相反磁场，因此定子的北磁场吸引永磁体的南磁场，反之亦然。这样就会导致转子总成旋转。

在反向模式下，Q3 完全打开，Q2 通过 PWM 信号打开和关闭，而 Q1 和 Q4 关闭。在反向模式下，电流以与正向模式相反的方向流过电机绕组。在正向模式下，电流通过 A 点进入电机并通过 B 点离开，而在反向模式下，电流通过 B 点进入电机并通过 A 点离开电机。这意味着在正向模式下极性为北磁场的定子臂，在反向模式下极性为南磁场。当霍尔传感器检测到磁场变化时，单片机会继续切换电流方向，使定子两极切换磁场，从而使永磁体跟随定子磁场变化。

转速通过 PWM 占空比控制。当 FET 接收到 0%占空比信号时，该 FET 不会导通，因此电流无法通过 FET 流至地，从而导致定子磁场的损失。随着占空比的增加，流经电机绕组的电流也会增加。这会在每个定子磁极处产生更强的磁场，进而将永磁体中的相反磁场更快拉到对齐状态。当占空比为 100%时，流经电机绕组的电流最大，从而产生最强的磁场密度，转子全速旋转。

3. 结论

本应用笔记讨论了如何使用 PIC18-Q43 单片机系列监视温度，控制一组风扇的速度，以及提供可视反馈以便查看。ADCC 模块用于确定热敏电阻的温度，16 位 PWM 模块用于控制每个风扇的转速，CCP 模块用于读取风扇的转速计输出以确定实际转速并进行速度调整，而 UART 模块则用于将温度和风扇速度发送至 PC 终端程序以便查看。此外，还简单讨论了 BLDC 风扇的工作原理。

Microchip 网站

Microchip 网站（www.microchip.com/）为客户提供在线支持。客户可通过该网站方便地获取文件和信息。我们的网站提供以下内容：

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持**——常见问题解答（FAQ）、技术支持请求、在线讨论组以及 Microchip 设计伙伴计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

产品变更通知服务

Microchip 的产品变更通知服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请访问 www.microchip.com/pcn，然后按照注册说明进行操作。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师（ESE）
- 技术支持

客户应联系其代理商、代表或 ESE 寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过 www.microchip.com/support 获得网上技术支持。

Microchip 器件代码保护功能

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规范。
- Microchip 确信：在正常使用的情况下，Microchip 系列产品非常安全。
- 目前，仍存在着用恶意、甚至是非法的方法来试图破坏代码保护功能的行为。我们确信，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。这种试图破坏代码保护功能的行为极可能侵犯 Microchip 的知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他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他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

法律声明

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc.及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc.的英文原版文档。

本出版物中提供的信息仅仅是为方便您使用 Microchip 产品或使用这些产品来进行设计。本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。

Microchip“按原样”提供这些信息。Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对非侵权性、适销性和特定用途的适用性的暗示担保，或针对其使用情况、质量或性能的担保。

在任何情况下，对于因这些信息或使用这些信息而产生的任何间接的、特殊的、惩罚性的、偶然的或间接的损失、损害或任何类型的开销，Microchip 概不承担任何责任，即使 Microchip 已被告知可能发生损害或损害可以预见。在法律允许的最大范围内，对于因这些信息或使用这些信息而产生的所有索赔，Microchip 在任何情况下所承担的全部责任均不超出您为获得这些信息向 Microchip 直接支付的金额（如有）。如果将 Microchip 器件用于生命维持和/或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切损害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任。除非另外声明，在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

商标

Microchip 的名称和徽标组合、Microchip 徽标、Adaptec、AnyRate、AVR、AVR 徽标、AVR Freaks、BesTime、BitCloud、chipKIT、chipKIT 徽标、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、HELDO、IGLOO、JukeBlox、KeeLoq、Kleer、LANCheck、LinkMD、maXStylus、maXTouch、MediaLB、megaAVR、Microsemi、Microsemi 徽标、MOST、MOST 徽标、MPLAB、OptoLyzer、PackeTime、PIC、picoPower、PICSTART、PIC32 徽标、PolarFire、Prochip Designer、QTouch、SAM-BA、SenGenuity、SpyNIC、SST、SST 徽标、SuperFlash、Symmetricom、SyncServer、Tachyon、TimeSource、tinyAVR、UNI/O、Vectron 及 XMEGA 均为 Microchip Technology Incorporated 在美国和其他国家或地区的注册商标。

AgileSwitch、APT、ClockWorks、The Embedded Control Solutions Company、EtherSynch、FlashTec、Hyper Speed Control、HyperLight Load、IntelliMOS、Libero、motorBench、mTouch、Powermite 3、Precision Edge、ProASIC、ProASIC Plus、ProASIC Plus 徽标、Quiet-Wire、SmartFusion、SyncWorld、Temux、TimeCesium、TimeHub、TimePictra、TimeProvider、WinPath 和 ZL 均为 Microchip Technology Incorporated 在美国的注册商标。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、Augmented Switching、BlueSky、BodyCom、CodeGuard、CryptoAuthentication、CryptoAutomotive、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、Espresso T1S、EtherGREEN、IdealBridge、In-Circuit Serial Programming、ICSP、INICnet、Intelligent Paralleling、Inter-Chip Connectivity、JitterBlocker、maxCrypto、maxView、memBrain、Mindi、MiWi、MPASM、MPF、MPLAB Certified 徽标、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REAL ICE、Ripple Blocker、RTAX、RTG4、SAM-ICE、Serial Quad I/O、simpleMAP、SimpliPHY、SmartBuffer、SMART-I.S.、storClad、SQI、SuperSwitcher、SuperSwitcher II、Switchtec、SynchroPHY、Total Endurance、TSHARC、USBCheck、VariSense、VectorBlox、VeriPHY、ViewSpan、WiperLock、XpressConnect 和 ZENA 均为 Microchip Technology Incorporated 在美国和其他国家或地区的商标。

SQTP 为 Microchip Technology Incorporated 在美国的服务标记。

Adaptec 徽标、Frequency on Demand、Silicon Storage Technology 和 Symmcom 均为 Microchip Technology Inc.在除美国外的国家或地区的注册商标。

GestIC 为 Microchip Technology Inc.的子公司 Microchip Technology Germany II GmbH & Co. KG 在除美国外的国家或地区的注册商标。

在此提及的所有其他商标均为各持有公司所有。

© 2021, Microchip Technology Incorporated 版权所有。

ISBN： 978-1-5224-6859-2

质量管理体系

有关 Microchip 的质量管理体系的信息，请访问 www.microchip.com/quality。

全球销售及服务网点

美洲	亚太地区	亚太地区	欧洲
公司总部 2355 West Chandler Blvd. Chandler, AZ 85224-6199 电话：480-792-7200 传真：480-792-7277 技术支持： www.microchip.com/support 网址： www.microchip.com **亚特兰大** 德卢斯，佐治亚州 电话：678-957-9614 传真：678-957-1455 **奥斯汀，德克萨斯州** 电话：512-257-3370 **波士顿** 韦斯特伯鲁，马萨诸塞州 电话：774-760-0087 传真：774-760-0088 **芝加哥** 艾塔斯卡，伊利诺伊州 电话：630-285-0071 传真：630-285-0075 **达拉斯** 阿迪森，德克萨斯州 电话：972-818-7423 传真：972-818-2924 **底特律** 诺维，密歇根州 电话：248-848-4000 **休斯顿，德克萨斯州** 电话：281-894-5983 **印第安纳波利斯** 诺布尔斯维尔，印第安纳州 电话：317-773-8323 传真：317-773-5453 电话：317-536-2380 **洛杉矶** 米慎维荷，加利福尼亚州 电话：949-462-9523 传真：949-462-9608 电话：951-273-7800 **罗利，北卡罗来纳州** 电话：919-844-7510 **纽约，纽约州** 电话：631-435-6000 **圣何塞，加利福尼亚州** 电话：408-735-9110 电话：408-436-4270 **加拿大 - 多伦多** 电话：905-695-1980 传真：905-695-2078	**澳大利亚 - 悉尼** 电话：61-2-9868-6733 **中国 - 北京** 电话：86-10-8569-7000 **中国 - 成都** 电话：86-28-8665-5511 **中国 - 重庆** 电话：86-23-8980-9588 **中国 - 东莞** 电话：86-769-8702-9880 **中国 - 广州** 电话：86-20-8755-8029 **中国 - 杭州** 电话：86-571-8792-8115 **中国 - 香港特别行政区** 电话：852-2943-5100 **中国 - 南京** 电话：86-25-8473-2460 **中国 - 青岛** 电话：86-532-8502-7355 **中国 - 上海** 电话：86-21-3326-8000 **中国 - 沈阳** 电话：86-24-2334-2829 **中国 - 深圳** 电话：86-755-8864-2200 **中国 - 苏州** 电话：86-186-6233-1526 **中国 - 武汉** 电话：86-27-5980-5300 **中国 - 西安** 电话：86-29-8833-7252 **中国 - 厦门** 电话：86-592-2388138 **中国 - 珠海** 电话：86-756-3210040	**印度 - 班加罗尔** 电话：91-80-3090-4444 **印度 - 新德里** 电话：91-11-4160-8631 **印度 - 浦那** 电话：91-20-4121-0141 **日本 - 大阪** 电话：81-6-6152-7160 **日本 - 东京** 电话：81-3-6880-3770 **韩国 - 大邱** 电话：82-53-744-4301 **韩国 - 首尔** 电话：82-2-554-7200 **马来西亚 - 吉隆坡** 电话：60-3-7651-7906 **马来西亚 - 槟榔屿** 电话：60-4-227-8870 **菲律宾 - 马尼拉** 电话：63-2-634-9065 **新加坡** 电话：65-6334-8870 **台湾地区 - 新竹** 电话：886-3-577-8366 **台湾地区 - 高雄** 电话：886-7-213-7830 **台湾地区 - 台北** 电话：886-2-2508-8600 **泰国 - 曼谷** 电话：66-2-694-1351 **越南 - 胡志明市** 电话：84-28-5448-2100	**奥地利 - 韦尔斯** 电话：43-7242-2244-39 传真：43-7242-2244-393 **丹麦 - 哥本哈根** 电话：45-4485-5910 传真：45-4485-2829 **芬兰 - 埃斯波** 电话：358-9-4520-820 **法国 - 巴黎** 电话：33-1-69-53-63-20 传真：33-1-69-30-90-79 **德国 - 加兴** 电话：49-8931-9700 **德国 - 哈恩** 电话：49-2129-3766400 **德国 - 海尔布隆** 电话：49-7131-72400 **德国 - 卡尔斯鲁厄** 电话：49-721-625370 **德国 - 慕尼黑** 电话：49-89-627-144-0 传真：49-89-627-144-44 **德国 - 罗森海姆** 电话：49-8031-354-560 **以色列 - 若那那市** 电话：972-9-744-7705 **意大利 - 米兰** 电话：39-0331-742611 传真：39-0331-466781 **意大利 - 帕多瓦** 电话：39-049-7625286 **荷兰 - 德卢内市** 电话：31-416-690399 传真：31-416-690340 **挪威 - 特隆赫姆** 电话：47-72884388 **波兰 - 华沙** 电话：48-22-3325737 **罗马尼亚 - 布加勒斯特** 电话：40-21-407-87-50 **西班牙 - 马德里** 电话：34-91-708-08-90 传真：34-91-708-08-91 **瑞典 - 哥德堡** 电话：46-31-704-60-40 **瑞典 - 斯德哥尔摩** 电话：46-8-5090-4654 **英国 - 沃金厄姆** 电话：44-118-921-5800 传真：44-118-921-5820