User's Guide

MSPM0 FOC 电机控制解决方案

Sal Ye; Vishnu, Ajith

摘要

本用户指南介绍了 MSPM0 FOC 电机控制解决方案的概况。其中还提供了在 CCS IDE 环境中使用支持的 DRV 硬件板搭建 MSPM0 LaunchPad 的分步说明，用于电机旋转与调优的参考调优流程，以及将工程应用于自定义电路板的迁移指南。

内容

商标

Code Composer Studio™ is a trademark of Texas Instruments.
所有商标均为其各自所有者的财产。

1 简介

无刷直流 (BLDC) 电机因其在处理可变负载、恒定负载和精密定位要求方面的多功能性，已在各种应用中得到广泛采用。这些电机广泛用于：

- 工业控制系统
- 车用
- 航空系统
- 自动化设备
- 医疗保健器件
- 各种其他专业应用

场定向控制 (FOC) 是一种常用的高性能 BLDC 电机控制方法。图 1-1 展示了利用两个电流传感器实现 FOC 的方框图，其中展示了一种全面的控制架构，包括：

- **双电流传感器反馈**
- **Park 变换**（将三相转换为旋转坐标系）
- **Clarke 变换**（将三相坐标系转换为静止坐标系）
- **Park 逆变换**（将静止坐标系转换回三相坐标系）
- **空间矢量 PWM 生成** (SVGEN)，用于优化开关

该设计采用增量开发方法来展示完整的 FOC 实现，并提供一个系统框架，其中集成了用于实现电机控制性能和效率的所有基本控制元件。

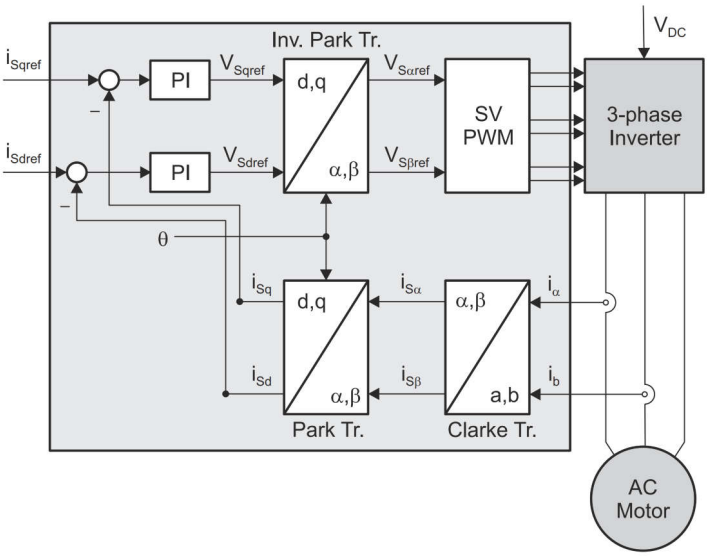


图 1-1. 使用两个电流传感器的无传感器磁场定向控制

了解转子磁通位置是 FOC 算法的基本基石，因此精确的位置反馈对于有效的电机控制至关重要。编码器提供了获取精确转子位置信息的最直接方法，其高精度可直接转化为出色的 FOC 性能。但是，由于存在额外的硬件要求，这种精度伴随着更高的系统成本。

无传感器 FOC 是一种很有吸引力的替代方案，无需物理位置传感器，从而显著降低了系统成本。没有传感器也使得无传感器系统本身更能抵抗电磁干扰，并能够在传统传感器发生故障的恶劣环境条件下可靠运行。这些优势的代价是算法复杂性增加，因为必须实施复杂的观测器技术来准确估算转子位置并保持高性能电机控制。

基于霍尔传感器的 FOC 代表了成本效益和性能之间的平衡。与无传感器实现方案相比，这种方法可增强低速控制的稳健性，但在更高的工作速度下会在性能上作出一定的妥协。

MSPM0 FOC 电机控制提供了一个中间件包来实现无传感器和带霍尔传感器 FOC，支持用户利用常见的电机驱动器设计，通过小型、简化的 MSPM0 固件示例在 30 分钟或更短时间内让 BLDC 电机旋转起来。

MSP FOC 电机控制工程需要：

- Code Composer Studio™ v12.8.0 或更高版本
- TI ARM CLANG Compilerv3.2.2 LTS 或更高版本
- MSPM0 SDK v2.05.00.05 或更高版本

2 电机控制理论

2.1 BLDC 电机基础知识

无刷直流 (BLDC) 电机由三个基本元件组成：一个绕线定子，一个永磁转子组件，以及可以在内部集成或外部安装的转子位置感应器件。这些位置传感器提供实时反馈，支持精确调整定子电压基准的频率和振幅，从而实现平滑的扭矩生成和转子连续旋转。

电机架构具有永磁转子磁芯，周围环绕外部定子绕组，可提供多种性能优势：

- **低转子惯性**，可改善动态响应
- 通过高效的散热路径实现**出色的散热**
- **紧凑的设计**，可减小电机整体尺寸

BLDC 电机可能具有梯形或正弦反电动势 (BEMF) 特性。本文档专门介绍了具有正弦 BEMF 波形的 BLDC 电机的电机控制实现方案 (1)。请遵循以下电机控制原则：

- 同步电机构造：永磁体被牢牢固定在旋转轴上，生成了一个恒定的转子磁通。这个转子磁通通常具有一个恒定的磁通量。定子绕组通电后可产生旋转电磁场。为了控制旋转的磁场，有必要控制定子电流。
- 根据机器的功率范围和额定速度，转子的实际结构会有所不同。永磁体适合于范围高达几千瓦的同步机器。为了获得更高的额定功率，转子通常由接通直流电的绕组组成。转子的机械结构是针对所需磁极的数量和所需的磁通梯度进行设计的。
- 定子和转子磁通的交感产生了一个转矩。由于定子被牢固地安装在电机架上，而转子可自由旋转，因此转子的旋转将产生一个有用的机械输出，如图 2-1 所示。
- 必须仔细控制转子磁场和定子磁场间的角度，以产生最大扭矩和实现较高的机电转换效率。为了实现这一目的，在同一转速和扭矩条件下，为了尽可能少地消耗电流，在关闭速度环路后需要使用无传感器算法进行微调。
- 旋转中的定子磁场的频率必须与转子永磁磁场的频率相同，否则转子就会经历快速的正负扭矩交替。这会减少最优扭矩产出量，并且在机器器件上产生过多的机械抖动、噪声和机械应力。此外，如果转子因惯性而不能对这些摆动做出响应，那么转子的转动会偏离同步频率，并且对静止转子的平均扭矩 (零扭矩) 做出响应。这意味着机器会出现一种称为*牵出*的现象。这也是为什么同步机器不能自启动的原因。
- 转子磁场与定子磁场间的角度必须等于 90º 以获得最高的互扭矩产出量。为了产生正确的定子磁场，该同步需要知道转子位置。
- 通过将不同转子相位的输出组合在一起，可将定子磁场设定为任一方向和强度以产生相应的定子磁通。

1. 包括库概述、软件设置、硬件设置等。具有正弦 BEMF 波形的 BLDC 电机通常也称为永磁同步 (PMSM) 电机。

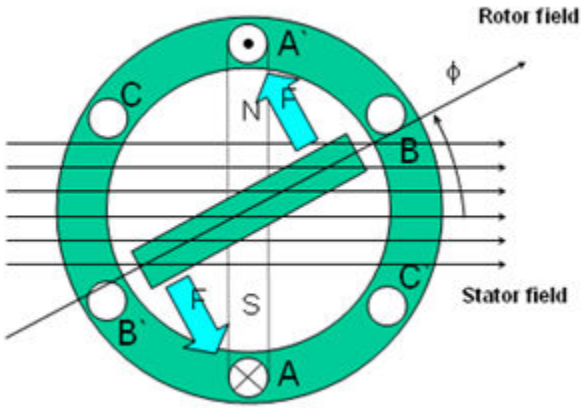


图 2-1. 旋转的定子磁通和转子磁通之间的相互作用产生扭矩

2.2 数学模型和 FOC 结构

PMSM 的无传感器 FOC 结构如图 2-2 所示。在该系统中，eSMO 用于实现 IPMSM 系统的无传感器控制，eSMO 模型是利用反电动势模型和 PLL 模型设计的，用于估算转子位置和转速。

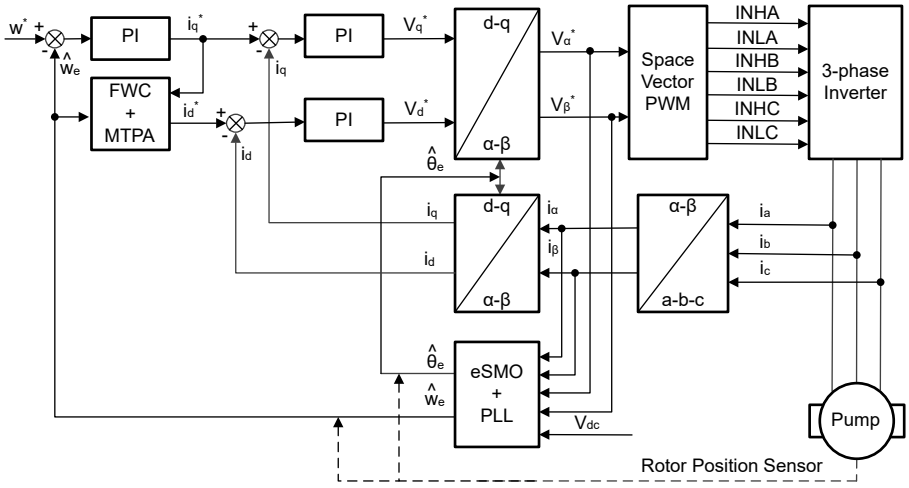


图 2-2. PMSM 系统的无传感器 FOC 结构

IPMSM 由一个三相定子绕组（a、b、c 轴）和用于励磁的永磁体 (PM) 转子组成。电机由标准的三相逆变器进行控制。可以使用相位 a-b-c 量对 IPMSM 进行建模。通过适当的坐标变换，可以得到 d-q 转子坐标系和 α - β 静止坐标系中的动态 PMSM 模型。这些坐标系之间的关系如下面的公式所示。通用 PMSM 的动态模型可以在 d-q 转子坐标系中写为：

$$\begin{bmatrix} v_d \\ v_q \end{bmatrix} = \begin{bmatrix} R_s + pL_d & -\omega_e L_q \\ \omega_e L_d & R_s + pL_q \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_e \lambda_{pm} \end{bmatrix} \quad (1)$$

其中 v_d 和 v_q 分别是 q 轴和 d 轴定子端电压；i_d 和 i_q 分别是 d 轴和 q 轴定子电流；L_d 和 L_q 分别是 q 轴和 d 轴电感；p 是导数算子，用于简写 $\frac{d}{dt}$；λ_{pm} 是永磁体产生的磁链；R_s 是定子绕组的电阻；ω_e 是转子的电角速度。

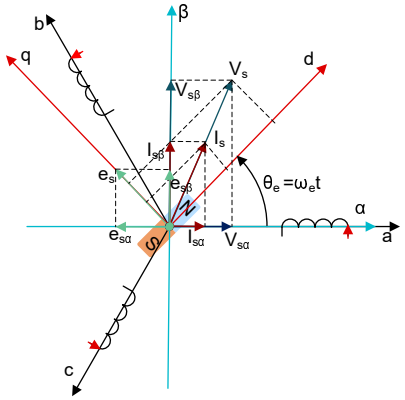


图 2-3. PMSM 建模坐标系的定义

通过使用上述 Park 逆变换，PMSM 的动力学可以在 α - β 静止坐标系中建模为：

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \begin{bmatrix} R_s + pL_d & \omega_e(L_d - L_q) \\ -\omega_e(L_d - L_q) & R_s + pL_q \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} \quad (2)$$

其中，e_a 和 e_β 是 α - β 轴上扩展电动势 (EEMF) 的分量，可以定义为：

$$\begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} = \left(\lambda_{pm} + \left(L_d - L_q\right) i_d\right) \omega_e \begin{bmatrix} -\sin(\theta_e) \\ \cos(\theta_e) \end{bmatrix} \tag{3}$$

根据这些公式，通过等效变换和引入 EEMF 概念，可以将转子位置信息从电感矩阵中去耦合，从而使 EEMF 成为唯一包含转子磁极位置信息的项。然后可以直接利用 EEMF 相位信息实现转子位置观测。使用定子电流作为状态变量，将 IPMSM 电压公式改写为状态公式：

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \frac{1}{L_d} \begin{bmatrix} -R_s & -\omega_e\left(L_d - L_q\right) \\ \omega_e\left(L_d - L_q\right) & -R_s \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \frac{1}{L_d} \begin{bmatrix} V_\alpha - e_\alpha \\ V_\beta - e_\beta \end{bmatrix} \tag{4}$$

由于定子电流是唯一可以直接测量的物理量，因此在定子电流路径上选择滑动面：

$$S(x) = \begin{bmatrix} \hat{i}_\alpha - i_\alpha \\ \hat{i}_\beta - i_\beta \end{bmatrix} = \begin{bmatrix} \tilde{i}_\alpha \\ \tilde{i}_\beta \end{bmatrix} \tag{5}$$

其中 $\hat{i}_\alpha$ 和 $\hat{i}_\beta$ 是估算的电流，上标 **^** 表示估算值，上标 **~** 表示变量误差，即观测值与实际测量值之间的差异。

2.3 无传感器场定向控制

在与家用电器类似的应用中，机械传感器会增加成本、影响可靠性并增加维护工作量。通常，在不要求以超低速度运行的应用中，会采用基于无传感器的转子位置估算方法来高效驱动电机。

为了使用无传感器方法检测转子位置，我们使用多种方法来估算电机的 BEMF，并且根据转子速度和角度进行了近似计算。滑模观测器 (SMO) 因其各种吸引人的特性（包括可靠性、所需的性能和针对系统参数变化的稳健性）而被广泛使用。

有限差分 BEMF 估算 (FD-BEMF) 方法也用于观测转子位置（仅限通用 FOC）。FD-BEMF 是基于简单公式的 BEMF 估算，没有用于 BEMF 的滑动模式控制器和滤波器，这消除了 Kslide 调优和滤波器调优工作，但 BEMF 容易产生噪声，并可能产生稳定性问题。

这两种方法得出的估算 BEMF 均用于使用 PLL 方法进行转子位置跟踪，如图 2-4 下。

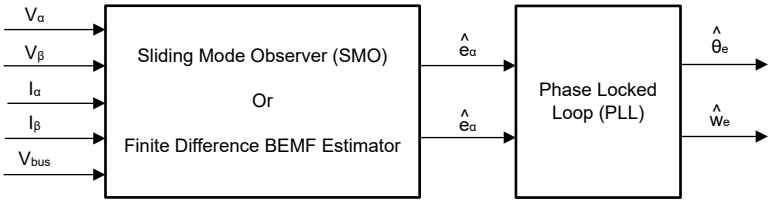


图 2-4. 转子角度观测器方框图

2.3.1 FOC 基础知识

为了实现更好的动态性能，需要采用更加复杂的控制方案来控制 PM 电机。借助微控制器提供的数学处理能力，我们可以实施先进的控制策略，这些策略使用数学变换将永磁电机中的扭矩生成和磁化功能解耦。这种解耦的扭矩和磁化控制通常称为转子磁通定向控制，或简称为磁场定向控制 (FOC)。

在直流 (DC) 电机中，定子和转子的励磁是独立控制的，产生的扭矩和磁通可以独立调整，如图 2-5 所示。磁场激励强度（例如，磁场激励电流的振幅）决定了磁通的大小。通过转子绕组的电流确定了扭矩是如何生成。转子上的换向器在扭矩产生过程中发挥着有趣的作用。换向器与电刷接触，这个机械构造旨在将电路切换至机械对齐的绕组以产生最大的扭矩。这样的安排意味着，电机的扭矩产生在任何时候都非常接近于最佳情况。这里的关键点是，通过管理绕组以保持转子绕组产生的磁通与定子磁场垂直。

为了实现更好的动态性能，需要采用更加复杂的控制方案来控制 PM 电机。借助微控制器提供的数学处理能力，我们可以实施先进的控制策略，这些策略使用数学变换将永磁电机中的扭矩生成和磁化功能解耦。这种解耦的扭矩和磁化控制通常称为转子磁通定向控制，或简称为磁场定向控制 (FOC)。

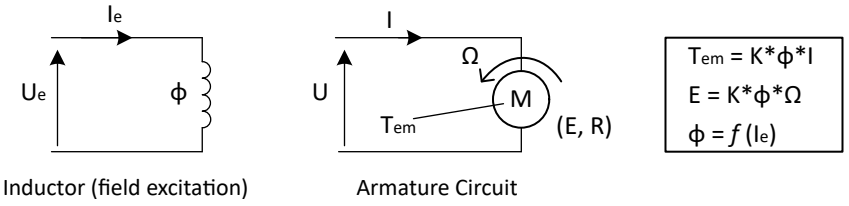


图 2-5. 在直流电机模型中磁通和扭矩是独立控制的

同步和异步电机上的 FOC（也称为矢量控制）旨在分别控制扭矩产生分量和磁化通量分量。利用 FOC 控制，我们能够解耦定子电流的扭矩分量和磁化通量分量。借助于磁化的去耦合控制，定子磁通的扭矩生成分量现在可以被看成是独立扭矩控制。为了去耦合扭矩和磁通，有必要采用几个数学变换，而这是最能体现微控制器价值的地方。微控制器提供的处理能力可非常快速地执行使这些数学变换。反过来，这意味着控制电机的整个算法可以高速率执行，从而实现了更高的动态性能。除了去耦合，现在一个电机的动态模型被用于很多数量的计算，例如转子磁通角和转子速度。这意味着，它们的影响被计算在内，并且总体控制质量更佳。

根据电磁定律，同步电机中产生的扭矩等于两个现有磁场的矢量叉积，如方程式 6 所示。

$$\tau_{em} = \vec{B}_{stator} \times \vec{B}_{rotor} \tag{6}$$

该表达式表明，如果定子和转子磁场正交，则扭矩最大，这意味着我们需要将负载保持在 90 度。如果我们能够始终确保满足这一条件，并且能够正确地对磁通进行定向，将减少扭矩纹波并确保实现更好的动态响应。然而，您需要了解转子的位置：这可以通过位置传感器（诸如递增编码器）实现。对于无法接近转子的低成本应用，采用不同的转子位置观察器策略可无需使用位置传感器。

简而言之，目标是使转子和定子磁通保持正交：例如，目标是将定子磁通与转子磁通的 q 轴对齐，从而与转子磁通正交。为了实现这个目的，控制与转子磁通正交的定子电流分量以产生命令规定的扭矩，并且直接分量被设定为零。定子电流的直接分量可用在某些磁场减弱的情况下，这有抗拒转子磁通的作用，并且减少反电动势，从而实现更高速的运行。

磁场定向控制包括控制由矢量表示的定子电流。该控制基于将三相时间和速度相关系统变换为两坐标（d 和 q 坐标）时不变系统的投影。这些设计导致一个与 DC 机器控制结构相似的结构。磁场定向控制 (FOC) 电机需要两个常数作为输入基准：扭矩分量（与 q 坐标对齐）和磁通分量（与 d 坐标对齐）。由于磁场定向控制只是基于这些投影，因此控制结构将处理瞬时电量。这使得在每次的工作运转过程中（稳定状态和瞬态）均可实现准确控制，并且与受限带宽数学模型无关。因此，FOC 通过以下方式解决了传统方案存在的问题：

- 轻松达到恒定基准（定子电流的扭矩分量和磁通分量）
- 轻松应用直接扭矩控制，这是因为在 (d, q) 坐标系中，扭矩的表达式定义如方程式 7 所示。

$$\tau_{em} \propto \psi_R \times i_{sq} \tag{7}$$

通过将转子磁通（ψ_R）的振幅保持在一个固定值，扭矩和扭矩分量（i_{Sq}）之间存在线性关系。然后我们可以通过控制定子电流矢量的扭矩分量来控制扭矩。

空间矢量定义和投影

交流电机的三相电压、电流和磁通可根据复数空间矢量进行分析。对于电流，空间矢量可定义如下。假设 i_a、i_b、i_c 是定子相中的瞬时电流，则复定子电流矢量的定义如方程式 8 所示。

$$\bar{i}_s = i_a + \alpha i_b + \alpha^2 i_c \tag{8}$$

其中 $\alpha = e^{j\frac{2}{3}\pi}$ 和 $\alpha^2 = e^{j\frac{4}{3}\pi}$ 表示空间运算符。

图 2-6 展示了定子电流复空间矢量。

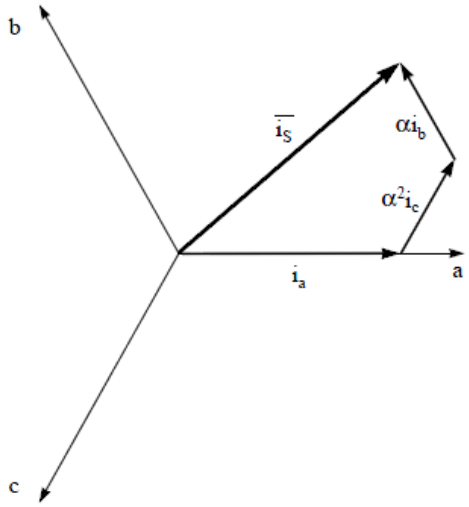


图 2-6. 定子电流空间矢量及其以 (a,b,c) 坐标系表示的分量

其中 (a,b,c) 是三相系统轴。这个电流空间矢量对三相正弦系统进行了描述，但仍需变换为一个两坐标非时变系统。这个变换可拆分为两个步骤：

- $(a,\ b) \Rightarrow (\alpha, \beta)$（Clarke 变换），输出一个两坐标时变系统
- $(\alpha, \beta) \Rightarrow (d,\ q)$（Park 变换），输出一个两坐标时不变系统

$(a,\ b) \Rightarrow (\alpha, \beta)$ Clarke 变换

可以使用另外一个仅包含两相 (α，β) 正交轴的坐标系来表示该空间矢量。假设 a 轴和 α 轴方向相同，我们可以得到下面图 2-7 所示的矢量图。

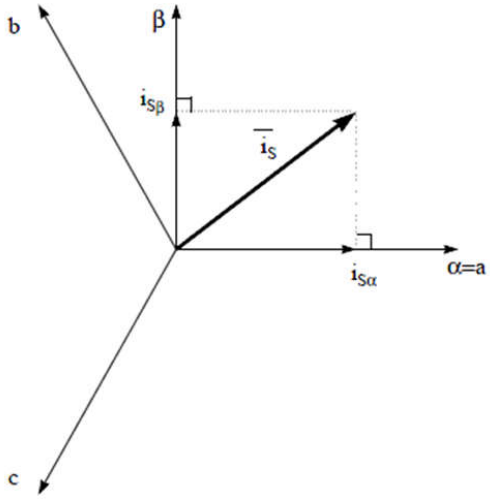


图 2-7. 静止坐标系中的定子电流空间矢量

将三相系统修改为 (α，β) 二维正交系统的投影如方程式 9 所示。

$$i_{s\alpha} = i_a$$
$$i_{s\beta} = \frac{1}{\sqrt{3}} i_a + \frac{2}{\sqrt{3}} i_b \quad (9)$$

两相 (α , β) 电流仍取决于时间和速度。

$(\alpha, \beta) \Rightarrow (d,\ q)$ **Park 变换**

这是 FOC 内最重要的变换。事实上，该投影在 (d, q) 旋转坐标系中修改了一个两相正交系统 (α , β)。如果我们考虑 d 轴与转子磁通对齐，那么图 2-8 显示了来自该二维坐标系的电流矢量的关系。

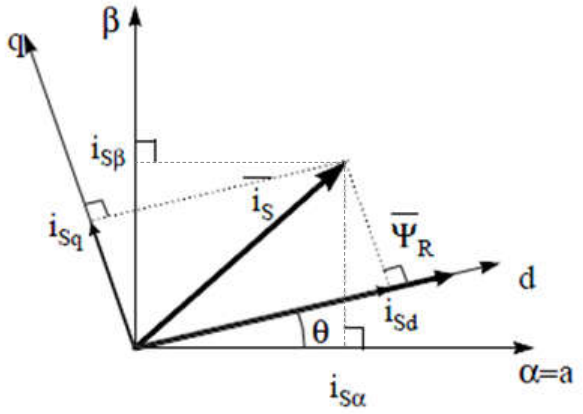


图 2-8. d,q 旋转坐标系中的定子电流空间矢量

电流矢量的磁通和扭矩分量由方程式 10 决定。

$$i_{sd} = i_{s\alpha}\cos(\theta) + i_{s\beta}\sin(\theta)$$
$$i_{sq} = -i_{s\alpha}\sin(\theta) + i_{s\beta}\cos(\theta) \quad (10)$$

其中 θ 是转子磁通位置

这些分量取决于电流矢量 (α , β) 分量和转子磁通位置；如果我们知道正确的转子磁通位置，那么，通过该投影，d,q 分量就变成一个常量。现在，两个相位电流变换为直流数量 (非时变)。此时扭矩控制变得更容易，其中恒定的 i_{sd} (磁通分量) 和 i_{sq} (扭矩分量) 电流分量单独受到控制。

交流电机 FOC 基本配置方案

图 2-9 总结了使用 FOC 进行扭矩控制的基本配置方案：

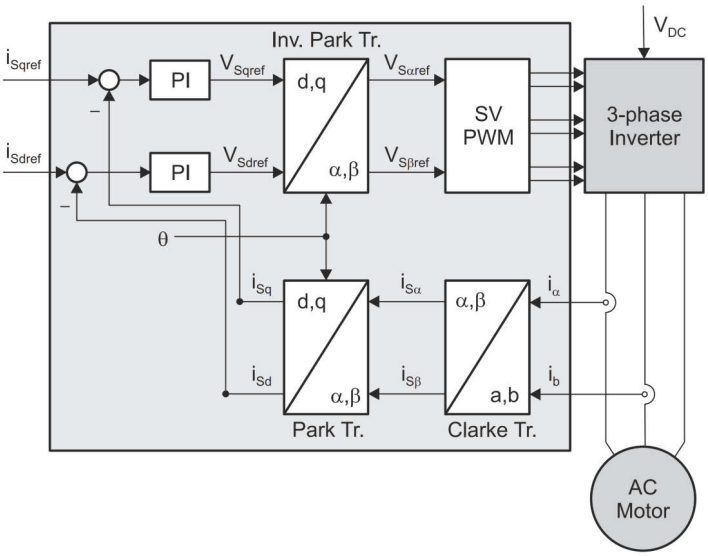


图 2-9. 交流电机 FOC 基本配置方案

测量了两个电机相电流。这些测量值馈入 Clarke 变换模块。这个模块的输出为 $i_{s\alpha}$ 和 $i_{s\beta}$。电流的这两个分量是 Park 变换的输入，该变换给出了 d,q 旋转坐标系中的电流。i_{sd} 和 i_{sq} 分量与基准 i_{sdref}（磁通基准分量）和 i_{sqref}（扭矩基准分量）进行比较。在这一点上，这个控制结构显示了一个有意思的优势：它可被用来控制同步或感应机器，采用的方法就是简单地改变磁通基准并获得转子磁通位置。与在同步永磁电机中一样，转子磁通是固定的，并由磁体确定；所以无需产生转子磁通。因此，当控制一个 PMSM 时，i_{sdref} 应被设定为 0。由于交流感应电机需要生成转子磁通才能运行，因此磁通基准一定不能为零。这很方便地解决了*经典*控制结构的一个主要缺陷：异步驱动至同步驱动的可移植性。当我们使用转速 FOC 时，扭矩命令 i_{sqref} 可以是转速调节器的输出。电流调节器的输出是 Vsdref 和 V_{sqref}；它们进行 Park 逆变换。

这个模块的输出是 $V_{s\alpha ref}$ 和 $V_{s\beta ref}$，它们是 (α，β) 静止正交坐标系中定子矢量电压的分量。这些是空间矢量脉宽调制 (PWM) 的输入。这个块的输出是驱动此反相器的信号。请注意，Park 和 Park 逆变换均需要转子磁通位置。这个转子磁通位置的获得由交流机器的类型（同步或异步机器）而定。

转子磁通位置

转子磁通位置的相关知识是 FOC 的核心。事实上，如果该变量存在误差，则转子磁通与 d 轴不对齐，并且定子电流的磁通和扭矩分量 i_{sd} 和 i_{sq} 不正确。图 2-10 展示了 (a, b, c)、(α，β) 和 (*d, q*) 坐标系，以及转子磁通的正确位置和以同步速度随 d,q 坐标旋转的定子电流和定子电压空间矢量。

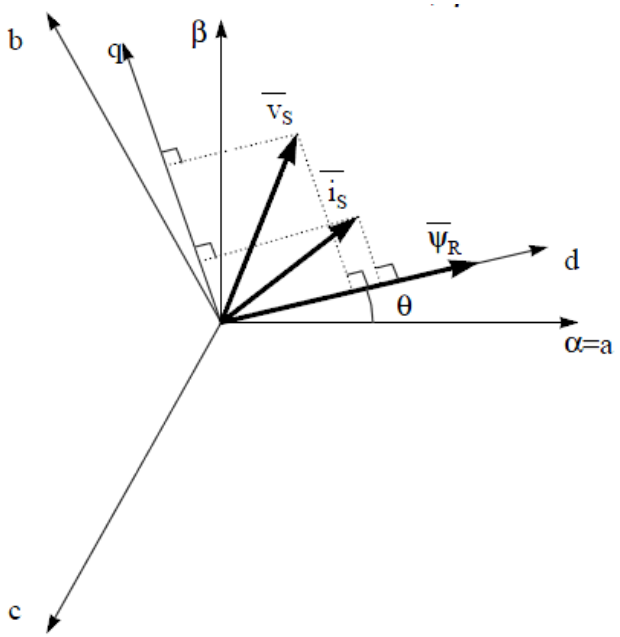


图 2-10. (d, q) 旋转坐标系中的电流、电压和转子磁通空间矢量

如果我们考虑同步或异步电机，转子磁通位置的测量是不同的：

- 在同步电机中，转子转速等于转子磁通转速。然后 θ（转子磁通位置）由位置传感器或转子速度的积分直接计算。
- 在异步电机中，转子转速不等于转子磁通转速（存在转差速度），因此需要使用特定的方法来计算 θ。基本方法是使用一个电流模型，该模型需要 *d, q* 坐标系中的电机模型的两个公式。

理论上，利用适用于 PMSM 驱动的磁场定向控制，可以使用磁通实现对电机扭矩的单独控制，这与直流电机的运行类似。换句话说，转矩和磁通互相之间去耦合。从静止坐标系到同步旋转坐标系间的变量变换需要知道转子位置信息。由于这种变换（所谓的 Park 变换），q 轴电流将控制扭矩，而 d 轴电流被强制设置为零。因此，该系统的关键模块是使用增强型滑模观测器 (eSMO) 或 FAST 估算器来估算转子位置。

图 2-11 展示了本文档中 PMSM 电机的无传感器 FOC（使用 eSMO 并具有弱磁控制 (FWC) 和每安培最大扭矩 (MTPA) 功能）的整体方框图。

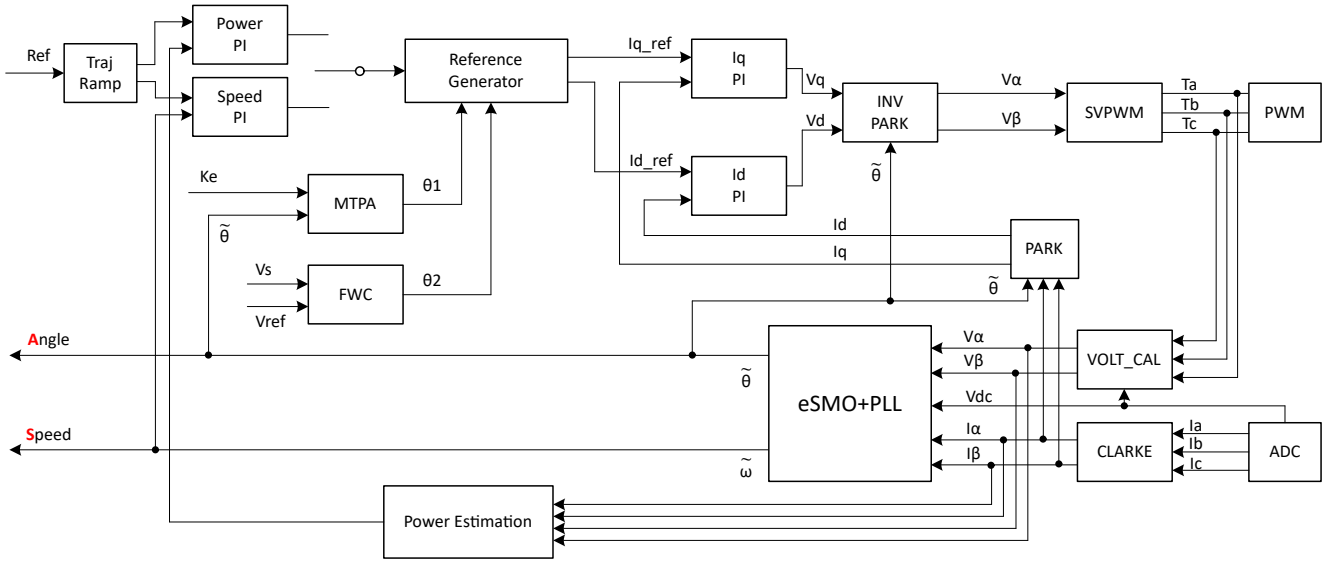


图 2-11. 使用 eSMO 并具有 FWC 和 MTPA 功能的 PMSM 的无传感器 FOC

2.3.2 增强型滑模观测器

基于模型的 BEMF 估算方法用于实现 IPMSM 驱动系统在电机以中高速运行时的无位置传感器控制。模型法通过反电动势或磁链模型估算转子位置。滑动模式观测器是基于滑模控制的观测器设计方法。系统的结构不是固定的，而是根据系统的当前状态有目的地改变，迫使系统按照预定的滑模轨迹运动。优点包括响应速度快、稳健性高以及对参数变化和干扰不敏感。

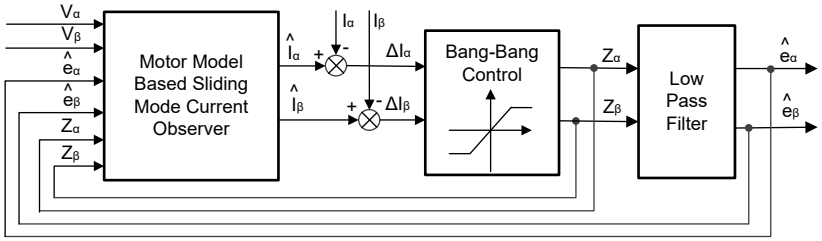


图 2-12. 传统滑模观测器的方框图

构建了传统的降阶滑模观测器，其数学模型如图 2-4 所示，方框图如图 2-12 所示。

$$\begin{bmatrix} \dot{\hat{i}}_\alpha \\ \dot{\hat{i}}_\beta \end{bmatrix} = \frac{1}{L_d}\begin{bmatrix} -R_s & -\hat{\omega}_e\left(L_d - L_q\right) \\ \hat{\omega}_e\left(L_d - L_q\right) & -R_s \end{bmatrix}\begin{bmatrix} \hat{i}_\alpha \\ \hat{i}_\beta \end{bmatrix} + \frac{1}{L_d}\begin{bmatrix} V_\alpha - \hat{e}_\alpha + z_\alpha \\ V_\beta - \hat{e}_\beta + z_\beta \end{bmatrix} \tag{11}$$

其中 z_α 和 z_β 是滑模反馈分量，其定义为：

$$\begin{bmatrix} z_\alpha \\ z_\beta \end{bmatrix} = \begin{bmatrix} k_\alpha sign\left(\hat{i}_\alpha - i_\alpha\right) \\ k_\beta sign\left(\hat{i}_\beta - i_\beta\right) \end{bmatrix} \tag{12}$$

其中 k_α 和 k_β 是通过李雅普诺夫稳定性分析设计的恒定滑模增益。如果 k_α 和 k_β 是足够大的正值，可保证 SMO 稳定运行， k_α 和 k_β 应足够大，以保持 $k_\alpha > \max(|e_\alpha|)$ 和 $k_\beta > \max(|e_\beta|)$ 。

α - β 轴上的 EEMF 估算值（ $\hat{e}_\alpha$ 、 $\hat{e}_\beta$ ）可通过低通滤波器从不连续开关信号中获得，这些信号为 z_α 和 z_α ：

$$\begin{bmatrix}\hat{e}_\alpha\\ \hat{e}_\beta\end{bmatrix} = \frac{\omega_c}{s+\omega_c}\begin{bmatrix}z_\alpha\\ z_\beta\end{bmatrix} \tag{13}$$

其中 $\omega_c = 2\pi f_c$ 是 LPF 的截止角频率，通常根据定子电流的基频来选择。

在数字控制应用中，需要使用 SMO 的时间离散方程。欧拉法是变换为时间离散观测器的合适方法。在 α - β 坐标中，方程式 11 的时间离散系统矩阵由方程式 14 给出：

$$\begin{bmatrix}\hat{i}_\alpha(n+1)\\ \hat{i}_\beta(n+1)\end{bmatrix} = \begin{bmatrix}F_\alpha\\ F_\beta\end{bmatrix}\begin{bmatrix}\hat{i}_\alpha(n)\\ \hat{i}_\beta(n)\end{bmatrix} + \begin{bmatrix}G_\alpha\\ G_\beta\end{bmatrix}\begin{bmatrix}V_\alpha^*(n) - \hat{e}_\alpha(n) + z_\alpha(n)\\ V_\beta^*(n) - \hat{e}_\beta(n) + z_\beta(n)\end{bmatrix} \tag{14}$$

其中矩阵 $[F]$ 和 $[G]$ 由方程式 15 和方程式 16 给出：

$$\begin{bmatrix}F_\alpha\\ F_\beta\end{bmatrix} = \begin{bmatrix}e^{-\frac{R_s}{L_d}}\\ e^{-\frac{R_s}{L_q}}\end{bmatrix} \tag{15}$$

$$\begin{bmatrix}G_\alpha\\ G_\beta\end{bmatrix} = \frac{1}{R_s}\begin{bmatrix}1-e^{-\frac{R_s}{L_d}}\\ 1-e^{-\frac{R_s}{L_q}}\end{bmatrix} \tag{16}$$

方程式 13 的时间离散形式由方程式 17 给出：

$$\begin{bmatrix}\hat{e}_\alpha(n+1)\\ \hat{e}_\beta(n+1)\end{bmatrix} = \begin{bmatrix}\hat{e}_\alpha(n)\\ \hat{e}_\beta(n)\end{bmatrix} + 2\pi f_c\begin{bmatrix}z_\alpha(n) - \hat{e}_\alpha(n)\\ z_\beta(n) - \hat{e}_\beta(n)\end{bmatrix} \tag{17}$$

2.3.3 有限差分 BEMF 估算器

在 Kslide (k_α 和 k_β) 不需要进行调优，或者系统中的噪声干扰极小的情况下，可以应用有限差分 BEMF 估算器。在 FD-BEMF 方法中，BEMF 根据方程式 18 所示的定子坐标系电压公式推导而来。

$$\begin{bmatrix}\mathrm{e}\,\hat{\alpha}\\ \mathrm{e}\,\hat{\beta}\end{bmatrix} = \begin{bmatrix}\mathrm{v}\,\hat{\alpha}\\ \mathrm{v}\,\hat{\beta}\end{bmatrix} - \begin{bmatrix}\mathrm{r_s} & 0\\ 0 & \mathrm{r_s}\end{bmatrix}\begin{bmatrix}\mathrm{i}\,\hat{\alpha}\\ \mathrm{i}\,\hat{\beta}\end{bmatrix} - \frac{\mathrm{d}}{\mathrm{dt}}\left(\begin{bmatrix}\mathrm{L_0} - \mathrm{L_1cos(2\theta)} & -\mathrm{L_1sin(2\theta)}\\ -\mathrm{L_1sin(2\theta)} & \mathrm{L_0} + \mathrm{L_1cos(2\theta)}\end{bmatrix}\begin{bmatrix}\mathrm{i}\,\hat{\alpha}\\ \mathrm{i}\,\hat{\beta}\end{bmatrix}\right) \tag{18}$$

其中， $\mathrm{L_0} = \left(\frac{\mathrm{L_d} + \mathrm{L_q}}{2}\right)$, $\mathrm{L_1} = \left(\frac{\mathrm{L_d} - \mathrm{L_q}}{2}\right)$ ；

如果电机没有任何凸极，例如在表面贴装 PMSM 电机中，则 L_1 值设置为零。

2.3.4 转子位置和转速估算

反正切估算器

图 2-4 中说明了与 SMO 配合使用的反正切估算器。

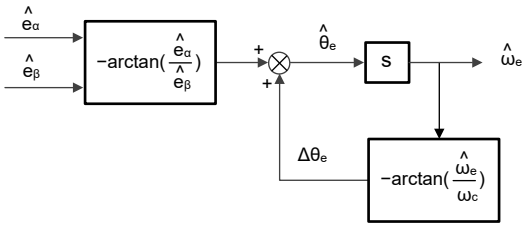


图 2-13. 反正切估算位置跟踪器的方框图

在传统的基于 SMO 的转子位置估算器中，转子磁通角根据估算的静止坐标 BEMF 值的反正切来确定，如方程式 19 所示：

$$\hat{\theta}_e = -\tan^{-1}\left(\frac{\hat{e}_\alpha}{\hat{e}_\beta}\right) \tag{19}$$

低通滤波器消除了滑模函数的高频项，这会导致出现相位延迟。可以通过截止频率 ω_c 和反电动势频率 ω_e 之间的关系对其进行补偿，定义为：

$$\Delta\theta_e = -\tan^{-1}\left(\frac{\omega_e}{\omega_c}\right) \tag{20}$$

这样使用 SMO 方法估算的转子位置就为：

$$\hat{\theta}_e = -\tan^{-1}\left(\frac{\hat{e}_\alpha}{\hat{e}_\beta}\right) + \Delta\theta_e \tag{21}$$

锁相环 (PLL) 跟踪器

在反正切估算法中，由于噪声和谐波分量的存在，位置和速度估算的精度会受到影响。为了消除该问题，可使用 PLL 模型对 PMSM 的无传感器控制结构中的转速和位置进行估算。图 2-4 中说明了与 SMO 配合使用的 PLL 结构。

反电动势 (BEMF) 估算 $\hat{e}_\alpha$ 和 $\hat{e}_\beta$ 可与 PLL 模型配合使用来估算电机角速度和位置，如图 2-14 所示。

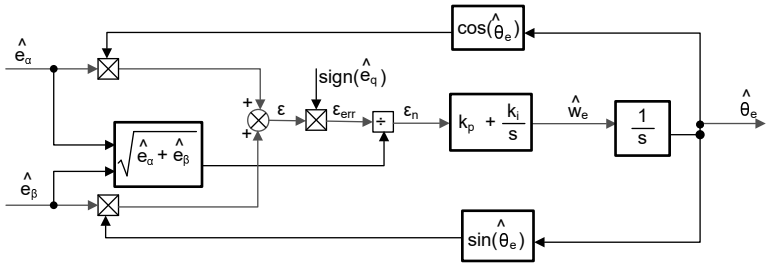


图 2-14. 锁相环位置跟踪器的方框图

由于 $e_\alpha = E\cos(\theta_e)$ 、 $e_\beta = E\sin(\theta_e)$ 和 $E = \omega_e\lambda_{pm}$ ，位置误差可定义为：

$$\varepsilon = \hat{e}_\beta\cos\left(\hat{\theta}_e\right) - \hat{e}_\alpha\sin\left(\hat{\theta}_e\right) = E\sin(\theta_e)\cos\left(\hat{\theta}_e\right) - E\cos(\theta_e)\sin\left(\hat{\theta}_e\right) = E\sin\left(\theta_e - \hat{\theta}_e\right) \tag{22}$$

其中 E 是 BEMF 的幅度，与电机转速 成正比ω_e。当 $\left(\theta_e - \hat{\theta}_e\right) < \frac{\pi}{2}$ ，方程式 22 可以简化为

$$\varepsilon = E\left(\theta_e - \hat{\theta}_e\right) \tag{23}$$

可以进一步得到 BEMF 归一化后的位置误差：

$$\varepsilon_n = \theta_e - \hat{\theta}_e \tag{24}$$

根据分析，可以得到正交锁相环位置跟踪器的简化方框图，如图 2-15 所示。

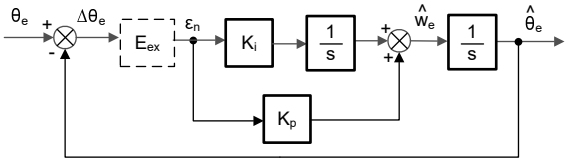


图 2-15. 锁相环位置跟踪器的简化方框图

PLL 的闭环传递函数可表示为方程式 25：

$$\frac{\hat{\theta}_e}{\theta_e} = \frac{k_p s + k_i}{s^2 + k_p s + k_i} = \frac{2\xi\omega_n s + \omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{25}$$

其中 k_p 和 k_i 是标准 PI 调节器的比例增益和积分增益，其固有频率 ω_n 和阻尼比 ξ 已给定：

$$k_p = 2\xi\omega_n, \qquad k_i = \omega_n^2 \tag{26}$$

3 MSP FOC 系统

下图展示了 FOC 电机控制系统的整体方框图。该系统主要包括三部分：电机驱动、电机控制和检测部分。电机驱动部分由栅极驱动器和 MOSFET 组成，直接连接到电机 (BLDC/PMSM)。电机通过电机驱动部分从电压总线获得电源。电机控制部分按照特定要求运行控制算法，并向栅极驱动器生成 6 个 PWM 信号。检测部分是电机驱动部分和电机控制部分的桥接器。检测部分通常包括电流检测、电压检测，可能还包括转子位置检测。检测部分用于为控制算法获取必要的电变量实时值。该拓扑的设计适用于低成本和小外形尺寸的 FOC 应用，例如泵、风扇、鼓风机和小型电器。

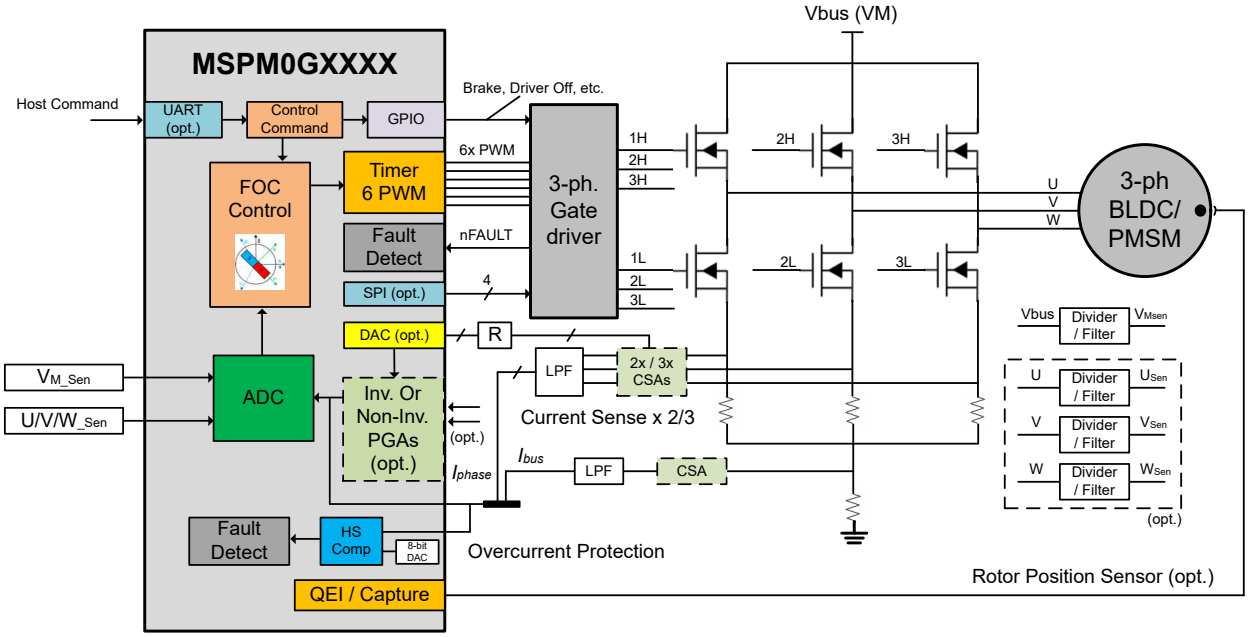


图 3-1. FOC 电机控制系统

MSPM0GX50X 系列器件提供了各种高性能模拟外设，例如两个 12 位 4Msps ADC（支持同步采样模式），可快速准确检测电机相电压、总线电压、相电流；一个 12 位 1Msps DAC，可自定义信号处理电路的基准电压；三个具有内置基准 DAC 的高速比较器，可执行硬件过流或过压保护；两个可编程的零漂移、零交叉放大器以及一个通用放大器，可放大小输入信号。

在低电压 FOC 应用中，许多栅极驱动器或电机驱动器器件集成了多达三个具有可编程增益的电流检测放大器，从而降低了 MSPM0GX50X 器件的模拟要求。无模拟集成的 MSPM0GX10X 器件采用小至 VSSOP-20 和 24-VQFN 的封装，可减小系统尺寸并降低成本。MSPM0GX51X 器件提供了更大的存储器和更大的封装选项。

3.1 设计资源

下表展示了总体 FOC 电机控制设计资源。可在 TI.COM 上选择可用的硬件解决方案，以搭配 LaunchPad 或板载 MSPM0（TIDA 参考设计）运行简单的代码示例。通过 TI Gallery 或 CCSUDIO IDE 工具中简单易用的 GUI 来旋转电机。

表 3-1. MSPM0 电机控制资源汇总

算法类型	MSPM0G LaunchPad™ 套件	电机驱动器硬件	GUI 链路	SDK 项目
无传感器/通用 FOC	LP-MSPM0G3507	DRV8316REVM	MSPM0 无传感器 FOC GUI 和 MSPM0 通用 FOC GUI	示例工程
	LP-MSPM0G3519			示例工程
	LP-MSPM0G3507	BOOSTXL-DRV8323RS		示例工程
	LP-MSPM0G3519			示例工程
	LP-MSPM0G3507	DRV8329AEVM		示例工程
	LP-MSPM0G3519			示例工程
	MSPM0G1507SPTR	TIDA-010250		示例工程
带传感器（霍尔）FOC	LP-MSPM0G3507	DRV8316REVM	MSPM0 带传感器 FOC GUI	示例工程
	LP-MSPM0G3519			示例工程
	MSPM0G1507SRHBR	TIDA-010251		示例工程

3.2 FOC 特性概述

下表展示了 MSP SDK FOC 控制算法支持的总体特性。

表 3-2. FOC 控制支持特性

支持特性		FOC 算法		
		无传感器 FOC	通用 FOC	带传感器 FOC
库类型	完全开源，便于用户自定义		√	√
	FOC 算法随静态库提供	√		
估算器	eSMO + PLL	√	√	
	FD-BEMF + PLL		√	
	霍尔（支持校准）			√
启动	对齐启动	√	√	
	慢速首循环启动	√	√	√
	IPD 启动	√	√	
	ISD 启动	√	√	
开环	速度和电流控制	√	√	√
	自动转换	√		
	自动扭矩电流斜降	√		
闭环	速度闭环	√	√	√
	电源闭环	√	√	√
	电流闭环	√	√	√
	PWM 开环	√	√	√
停止	滑行（高阻态）	√	√	√
	主动降速	√	√	√
	制动	√	√	√
故障处理	电机保护/电路板保护	√	√	√
死区时间补偿	补偿 PWM 死区时间	√		√
OVM	生成 SVPWM 时进行过调制	√	√	√
MTPA	每安培最大扭矩控制算法	√	√	√
FWC	弱磁控制算法	√	√	√
分流电阻器支持	单/双/三分流器采样模式	√	√	√

3.3 FOC 基准

下表展示了 MSP SDK FOC 控制算法的基准。

表 3-3. FOC 控制基准

FOC 算法	产品	CPU/时钟	PWM 频率	FOC 速率	FOC 时间	CPU 带宽
无传感器 FOC	MSPM0G3507(1)	M0+ MathACL / 80MHz	20kHz	10kHz	60.8us	60.8%
	MSPM0G3107(2)	不带 MathACL 的 M0+ / 80MHz	20kHz	10kHz	79.9us	79.9%
	MSPM0C1106(2)	不带 MathACL 的 M0+ / 32MHz	15kHz	5kHz	199us	99.5%
带传感器 FOC	MSPM0G3507(1)	M0+ MathACL / 80MHz	16kHz	16kHz	43.9us	70.3%
	MSPM0C1106(2)	不带 MathACL 的 M0+ / 32MHz	15kHz	5kHz	144us	72.0%

(1) 该基准是使用 SDK FOC 解决方案测试得出的。
(2) 该基准是使用 SDK FOC 解决方案估算得出的。

4 MSP FOC 硬件

表 4-1 展示了支持的 MSPM0 LaunchPad 套件和用于三相无传感器 FOC 电机控制的 EVM。在开发自定义硬件设计时，请参阅节 8。

表 4-1. 为 FOC 使用 MSPM0 时支持的硬件

电机驱动器硬件	硬件用户指南	电流检测放大器	SPI 驱动器支持	电机电压范围（建议值）	电机功率（建议值）
BOOSTXL-DRV8323RS	EVM 用户指南	3	是	6-60V	< 1000W
DRV8316REVM	EVM 用户指南	3	是	4.5-40V	< 75W
DRV8329AEVM	EVM 用户指南	1	否	4.5-60V	< 1000W
TIDA-010250	TIDA 设计指南	2（在 M0 中）	否	最大 265V AC	< 1000W
TIDA-010251	TIDA 设计指南	2（在 M0 中）	否	5-21V	< 600W

用于 LaunchPad 套件和 EVM 的跳线配置。按照表 4-2 中的指南将所选的硬件评估板与 MSPM0 LaunchPad 相连接。

表 4-2. LaunchPad 和 EVM 板的硬件连接指南

FOC 算法	SDK 用户指南(1)
无传感器 FOC	无传感器 FOC SDK 用户指南
通用 FOC	通用 FOC SDK 用户指南
带传感器 FOC	带传感器 FOC SDK 用户指南

(1) 包括库概述、软件设置、硬件设置等。

以下章节将介绍 LP-MSPM0G3507 的硬件设计。LP-MSPM0G3519 或其他器件的设计相同，只是由于 BoosterPack 引脚分配差异，PINMUX 略有不同。

4.1 PWM 引脚配置

下表给出了 LP-MSPM0G3507 的默认 PWM 输出引脚配置。所需的连接为六个 PWM 输出信号，这些信号将换向模式发送到栅极驱动器，以实现无传感器 FOC 电机控制。选择 MSPM0 的 TIMA 实例来进行 FOC 电机控制，该实例具有带死区的互补 PWM 输出、响应时间小于 40ns 的故障处理能力，以及用于配置 FOC 环路速率的重复计数器。

表 4-3. PWM 输出的引脚配置

MSPM0 引脚	引脚功能	DRV 引脚连接	DRV 功能
PB4	TIMA0_C2，TIMA0 通道 2 输出引脚	INHA	A 相高侧 PWM 输入
PB1	TIMA0_C2N，TIMA0 通道 2 互补输出引脚	INLA	A 相低侧 PWM 输入
PA28	TIMA0_C3，TIMA0 通道 3 输出引脚	INHB	B 相高侧 PWM 输入
PA31	TIMA0_C3N，TIMA0 通道 3 互补输出引脚	INLB	B 相低侧 PWM 输入
PB20	TIMA0_C1，TIMA0 通道 1 输出引脚	INHC	C 相高侧 PWM 输入
PB13	TIMA0_C1N，TIMA0 通道 1 互补输出引脚	INLC	C 相低侧 PWM 输入
PA0	TIMA0_C0，TIMA0 通道 0，无输出。它用于通过硬件触发 ADC 启动电流采样		

TIMA0 是电机控制的首选计时器，因为它可以直接从同一个计时器计数器（例如 TIMA0_C1 和 TIMA0_C1N）提供三对互补的 PWM 输出。此外，任何 TIMA0 或 TIMA1 输出对均可结合交叉触发功能使用，以提供同步的三对 PWM 输出。如果需要修改 PWM 输出通道或引脚，请参阅节 8.3.1。

4.2 ADC 引脚配置

ADC0 和 ADC1 是两个同时采样的 4Msps 模数转换器，用于测量相电流和电压。ADC0 和 ADC1 支持在正常电机运行条件下同时测量相电流，并根据转子角度按顺序测量总线电压。

如果需要修改 ADC 通道，请参阅节 8.3.2。

4.2.1 直流总线电压

针对 ADC 电压的 LP-MSPM0G3507 默认引脚配置如下表所示。

表 4-4. 针对直流总线的 ADC 引脚配置

MSPM0 引脚	MSPM0 功能	DRV8323 引脚连接	DRV8323 功能
A0_2	ADC 0，通道 2 输入	VSENVM	直流总线电压输出

检测电压通过一个电阻分压器和一个可选的旁路滤波电容器来实现，如下图所示。调整电阻器的大小，使任何电机电压瞬变都不超过 ADC 输入的最大电压。

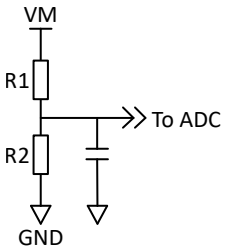


图 4-1. 直流总线电压分压器

4.2.2 电机相电压

仅当需要**初始速度检测** (ISD) 特性时，才需要相电压检测。针对 ADC 电压的 LP-MSPM0G3507 默认引脚配置如下表所示。

表 4-5. 针对电机相电压的 ADC 引脚配置

MSPM0 引脚	MSPM0 功能	DRV8323 引脚连接	DRV8323 功能
A1_6	ADC 1，通道 6 输入	VSENA	A 相检测电压输出
A0_7	ADC 0，通道 7 输入	VSENB	B 相检测电压输出
A1_5	ADC 1，通道 5 输入	VSENC	C 相检测电压输出

4.2.3 电机相电流

根据电流采样方法，需要将多个 (1~3 个) ADC 输入连接到电机驱动器的 CSA 输出、外部 CSA 或 MCU 内置 OPA。可以在 CSA 输出到 ADC 输入之间串联一个低通 RC 滤波器，以滤除开关输出信号中的任何高频噪声，从而进行正确的 ADC 采样，如下图所示。

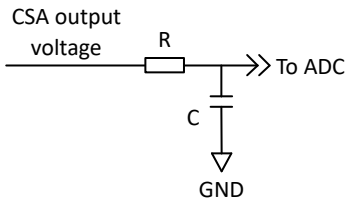


图 4-2. CSA 输出滤波器

如果用于检测电机电流，选择一个至少为 PWM 开关频率 10 倍的滤波频率。

4.2.3.1 单分流器电流检测

DRV8329 中针对单分流器电流检测的 LP-MSPM0G3507 默认 ADC 引脚配置如下表所示。在单分流器电流检测中，使用 ADC0/ADC1 中的一个在单个 PWM 周期内的两个不同实例上对相同的分流器电流进行采样，以便估算三相电流。

表 4-6. 针对单分流器检测的 ADC 引脚配置

MSPM0 引脚	MSPM0 功能	DRV8329 引脚连接	DRV8329 功能
A1_2	ADC 1，通道 2 输入	ISENA	直流总线电流检测

4.2.3.2 双或三分流器电流检测

针对三相电流检测的 LP-MSPM0G3507 默认 ADC 引脚配置如下表所示。需要将三个 ADC 输入连接到电机驱动器或外部 CSA 的三个 CSA 输出。

表 4-7. 针对三分流器电流检测的 ADC 引脚配置

MSPM0 引脚	MSPM0 功能	DRV8323 引脚连接	DRV8323 功能
A1_2	ADC 1，通道 2 输入	ISENA	A 相电流检测输出
A0_3	ADC 0，通道 3 输入	ISENB	B 相电流检测输出
A1_3	ADC 1，通道 3 输入	ISENC	C 相电流检测输出

为了减轻 ADC 转换延迟对 FOC 算法的影响，建议使用两个不同的 ADC 实例进行三相电流检测。

4.2.3.3 采用同步采样的三分流器电流检测

要精确获得电机相电流的实时值，用户可以启用同步采样模式，这需要将 B 相电流输入同时连接到两个不同的 ADC 实例，并将 A 相和 C 相分别连接到不同的 ADC 实例。下表给出了用于三分流器电机电流同步采样的 LP-MSPM0G3507 默认 ADC 引脚配置。

表 4-8. 采用同步采样的 ADC 引脚配置

MSPM0 引脚	MSPM0 功能	DRV8316 引脚连接	DRV8316 功能
A1_13	ADC 1，通道 13 输入	ISENA	A 相电流检测输出
A1_14	ADC 1，通道 14 输入	ISENB	B 相电流检测输出
A0_12	ADC 0，通道 12 输入	ISENB	B 相电流检测输出
A0_2	ADC 0，通道 2 输入	ISENC	C 相电流检测输出

在运行时，算法选择要同时采样的占空比较大的两个相位，然后根据采样结果计算最后一个相电流值。

4.3 故障引脚配置

通常，当检测到系统中存在故障时，电机驱动器会驱动低电平有效的开漏故障引脚 (nFAULT)，比如，在驱动器内与 MOSFET 过流、栅极驱动或电源相关故障的连接中。

MSPM0 MCU 可以通过专用硬件路径检测故障输入，以直接禁用 PWM 输出，并且与 GPIO 中断解决方案相比具有更低的延迟和响应时间。如果需要修改故障引脚，请参阅节 4.3。

下表给出了用于故障检测的默认引脚配置。

表 4-9. 故障引脚配置

MSPM0 引脚	MSPM0 功能	DRV8323 引脚连接	DRV8323 功能
TIMA_FAL2	TIMA 故障处理输入 2	nFAULT	开漏低电平有效故障引脚

4.4 霍尔 GPIO 引脚配置

带传感器 FOC 需要来自 BLDC/PMSM 电机的霍尔传感器信号，才能在闭环速度控制下运行电机并高效驱动电机。通常，霍尔信号需要外部上拉电阻来驱动高电平输入。三个霍尔传感器输入信号通过通用输入/输出 (GPIO) 输入馈送，并根据 GPIO 电平变化生成事件，发送到特定计时器。当三个 GPIO 输入中的任一电平发生变化时，将触发一个计时器捕获事件。32 位计时器 TIMG12 用于捕获该计时器捕获事件。如果需要修改霍尔输入引脚，请参阅节 8.3.4。

备注

所有三个 GPIO 引脚都必须连接到同一端口和段，以实现事件触发。

下表总结了 LP-MSPM0G3507 上具有 MSPM0 功能的 GPIO 引脚配置。

表 4-10. 霍尔 GPIO 引脚配置

MSPM0 引脚	MSPM0 功能	DRV8316 引脚连接	DRV8316 功能
PA10	GPIO 端口 A 输入引脚 10	HALLA	霍尔 A 相信号输出
PA11	GPIO 端口 A 输入引脚 11	HALLB	霍尔 B 相信号输出
PA12	GPIO 端口 A 输入引脚 12	HALLC	霍尔 C 相信号输出

4.5 GPIO 引脚配置

MSPM0 提供了多种 GPIO 输出功能，可用于由逻辑电平引脚控制的电机驱动器特定操作。电机驱动器功能的示例包括：

- 使能引脚 (ENABLE)/低电平有效睡眠模式控制 (nSLEEP)
- 高电平有效栅极驱动器关断 (DRVOFF)
- 高电平有效 CSA 校准 (CAL)
- 高电平有效制动 (BRAKE)/低电平有效制动 (nBRAKE)
- 方向引脚 (DIR)

有关 GPIO 可配置引脚，请参阅特定电机驱动器器件的数据表和 EVM 用户指南。如果需要修改 GPIO 引脚，请参阅节 4.5。

4.6 SPI 引脚配置

SPI 连接的默认引脚配置如下表所示。一些电机驱动器包括可选的 SPI，用于配置控制寄存器和读取状态寄存器以进行故障诊断。通过 SPI 通信，用户可以：

- 配置栅极驱动拉电流/灌电流强度
- 配置 CSA 输出行为
- 运行诊断
- 检测到故障引脚处于有效的低电平时读取故障位
- 故障条件消除后清除故障状态位

表 4-11. SPI 引脚配置

MSPM0 引脚	MSPM0 功能	DRV 引脚连接	DRV 功能
SPIx_CSy	SPI 芯片选择（y = 0、1、2、3）	nSCS	SPI 芯片选择
SPIx_SCK	SPI 总线时钟	SCLK	SPI 总线时钟
SPIx_POCI	SPI 外设输出控制器输入	SDO	SPI 数据输出
SPIx_PICO	SPI 外设输入控制器输出	SDI	SPI 数据输入

4.7 UART 引脚配置

UART 可用于接收命令以配置、旋转和控制电机。这些命令从主机 MCU 或 GUI 发送，并可选择性地用于 LIN 通信等高级协议。

备注

对 MSPM0GX50X 和 MSPM0GX10X 使用 UART 实例 3（UART3_RX、UART3_TX）以及 DMA。详细信息请参阅勘误表中的 DMA_ERR_01。

表 4-12 展示了 UART 与 PC 连接的默认引脚配置。表 4-13 展示了用于 UART 通信的 LP-MSPM0G3507 的 MSPM0 PINMUX。

表 4-12. UART 引脚配置

MSPM0 引脚	MSPM0 功能	J101 连接	XDS110 功能
UART3_TX	UART 发送	TXD>>	反向通道 UART 接收
UART3_RX	UART 接收	RXD<<	反向通道 UART 发送

表 4-13. 用于 UART 通信的 LP-MSPM0G3507 PINMUX

栅极驱动器 EVM 板	UART TX	UART RX
DRV8316REVM	PA26	PA13
BOOSTXL-DRV8323RS	PB12	
DRV8329AEVM	PB12	

4.8 评估板的外部连接

将 MSPM0 LaunchPad 连接到 DRV EVM 板时，请按照以下步骤操作：

1. 将三个电机相位端子连接到驱动器板（A、B 和 C 相）。如果电机有中心抽头连接，请将这些导线保持未连接状态。
2. 仅当部署带传感器 FOC 解决方案时，才将三个霍尔传感器输入连接到 DRV 霍尔接口接头。
   a. 如果 DRV EVM 板没有霍尔接口，用户可以将霍尔传感器输入直接连接到 LaunchPad，并启用 MCU 霍尔输入引脚的上拉电阻器。
3. 通过将 EVM 与 LaunchPad 套件匹配或使用跳线来实现从 MSPM0 LaunchPad 套件到 DRV83XX EVM 板的器件间连接。有关连接的硬件用户指南详细信息，请参阅《无传感器 FOC EVM 板连接指南》和《带传感器 FOC EVM 板连接指南》。
4. 用 Micro-USB 电缆将 MSPM0 LaunchPad 套件连接到 PC。
5. 提供符合电源电压 (VM) 范围的电压。有关建议的电压范围，请参阅特定于电路板的用户指南或 DRV 特定数据表。

5 MSP FOC 软件

MSPM0-SDK 中提供了 MSPM0 MCU 的 FOC 软件，并且提供了示例工程以供使用 Code Composer Studio™ (CCS) IDE 进行评估。

下表展示了 TI Resource Explorer 中的 FOC 控制所支持的软件和文档。

表 5-1. FOC 控制的软件支持

FOC 算法	SDK 代码示例
无传感器 FOC	无传感器 FOC 示例
通用 FOC	通用 FOC 示例
带传感器 FOC	带传感器 FOC 示例

下表展示了本文档使用的软件版本。对于更高的 FOC 算法版本，可能有一些升级功能，请参阅表 5-5 中链接的每个 FOC 算法的最新具体调优指南。

表 5-2. 使用的软件版本

FOC 算法	SDK 版本	算法版本
无传感器 FOC	2.08.00.03	2.04.01
通用 FOC	2.08.00.03	1.01.01
带传感器 FOC	2.08.00.03	1.02.01

5.1 工程结构

图 5-1 展示了通用 FOC 工程的 CCS 工程结构。

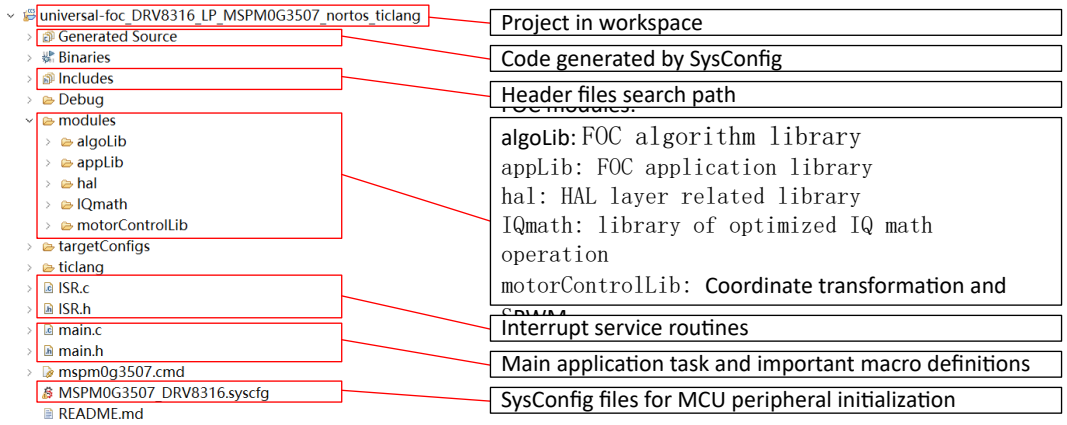


图 5-1. 工程结构概览

每个 FOC 模块的详细工程结构如下所示。

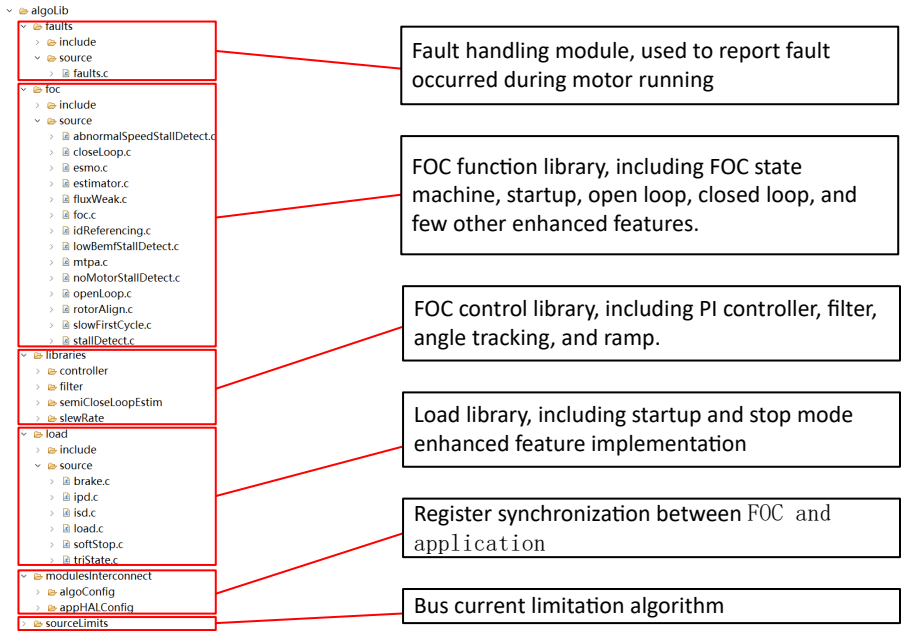


图 5-2. 工程结构概览：algoLib

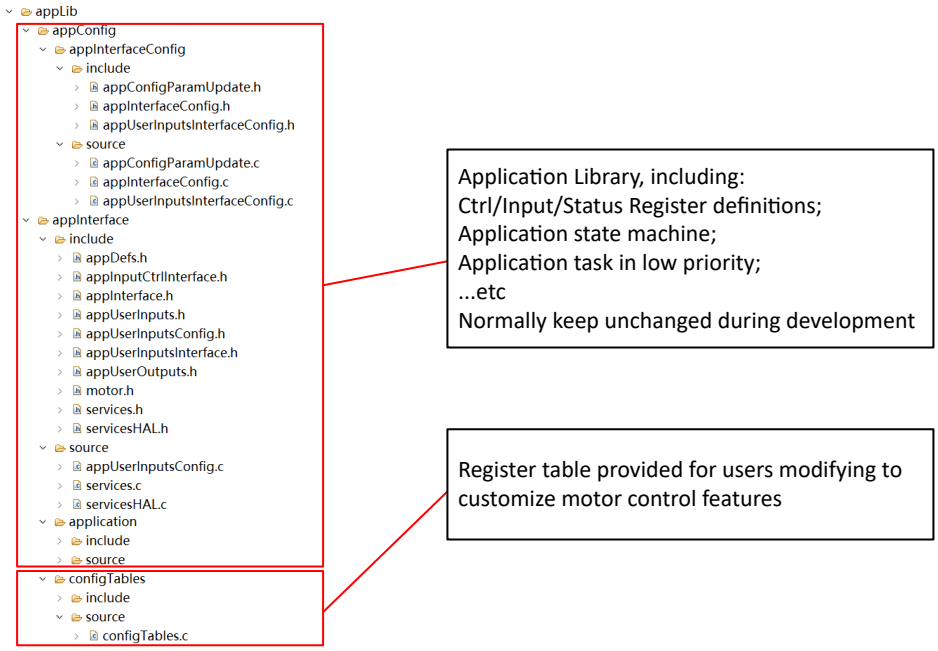


图 5-3. 工程结构概览：appLib

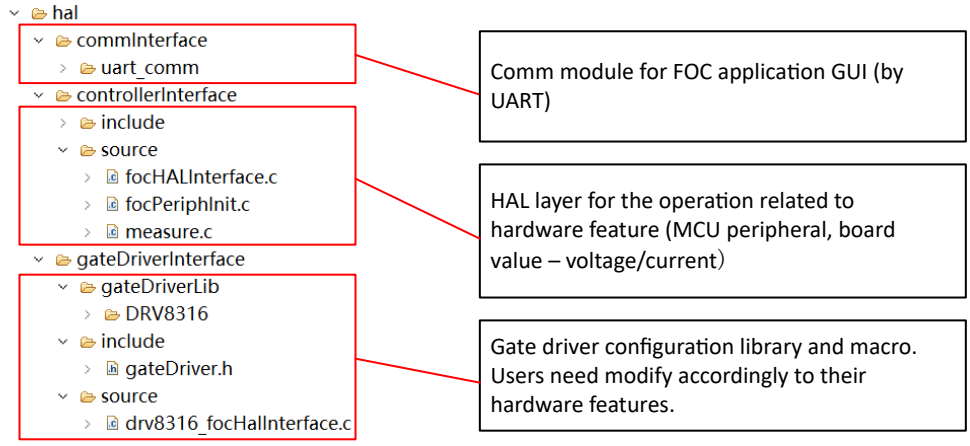


图 5-4. 工程结构概览：hal

无传感器 FOC 工程具有类似的工程结构，但它具有静态 FOC 库。带传感器 FOC 工程具有与霍尔估算器类似的工程结构。

5.2 软件概述

场定向控制 (FOC) 算法包含三个主要层：应用层、HAL 层和 MSPM0 Driverlib 层，如下图所示。

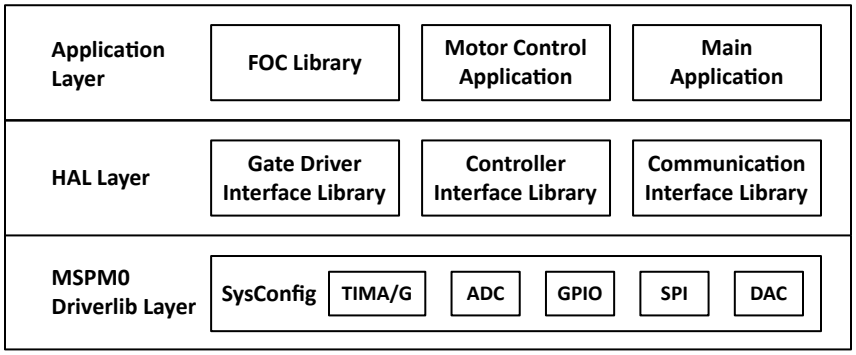


图 5-5. FOC 算法架构

5.2.1 应用层

应用层是执行电机控制等特定功能的软件层。如果任何应用执行任何硬件特定操作，将通过 HAL 层 API 执行。应用层包含三部分：FOC 库、电机控制应用和主应用。

5.2.1.1 FOC 库

FOC 库包含用于三相 FOC 电机控制的通用电机控制算法。表 3-2 总结了支持的 FOC 库特性。

备注

无传感器 FOC 库是一个静态库，不支持修改。如果需要自定义 FOC 电机控制源代码或状态机，请改用通用 FOC 库。

5.2.1.2 电机控制应用

电机控制应用负责对用户输入进行寄存器值转换并馈送到 FOC 库。该应用通过周期性 1ms 计时器触发，在较低优先级中断服务例程 (ISR) 中运行，主要任务如下所述：

- 切换电机控制应用状态
- 检查 HV_IDE 故障 (nFAULT 输入)
- 监测直流总线电压
- 设置 MTPA 和弱磁功能参考 d 轴电流
- 更新速度、功率、扭矩输入并运行输入斜坡
- 更新用户状态寄存器

5.2.1.3 主应用程序

主应用包含的用户代码用于初始化应用模块，并在 while(1) 循环中定期调用配置更新。下图展示了主应用代码流程方框图。

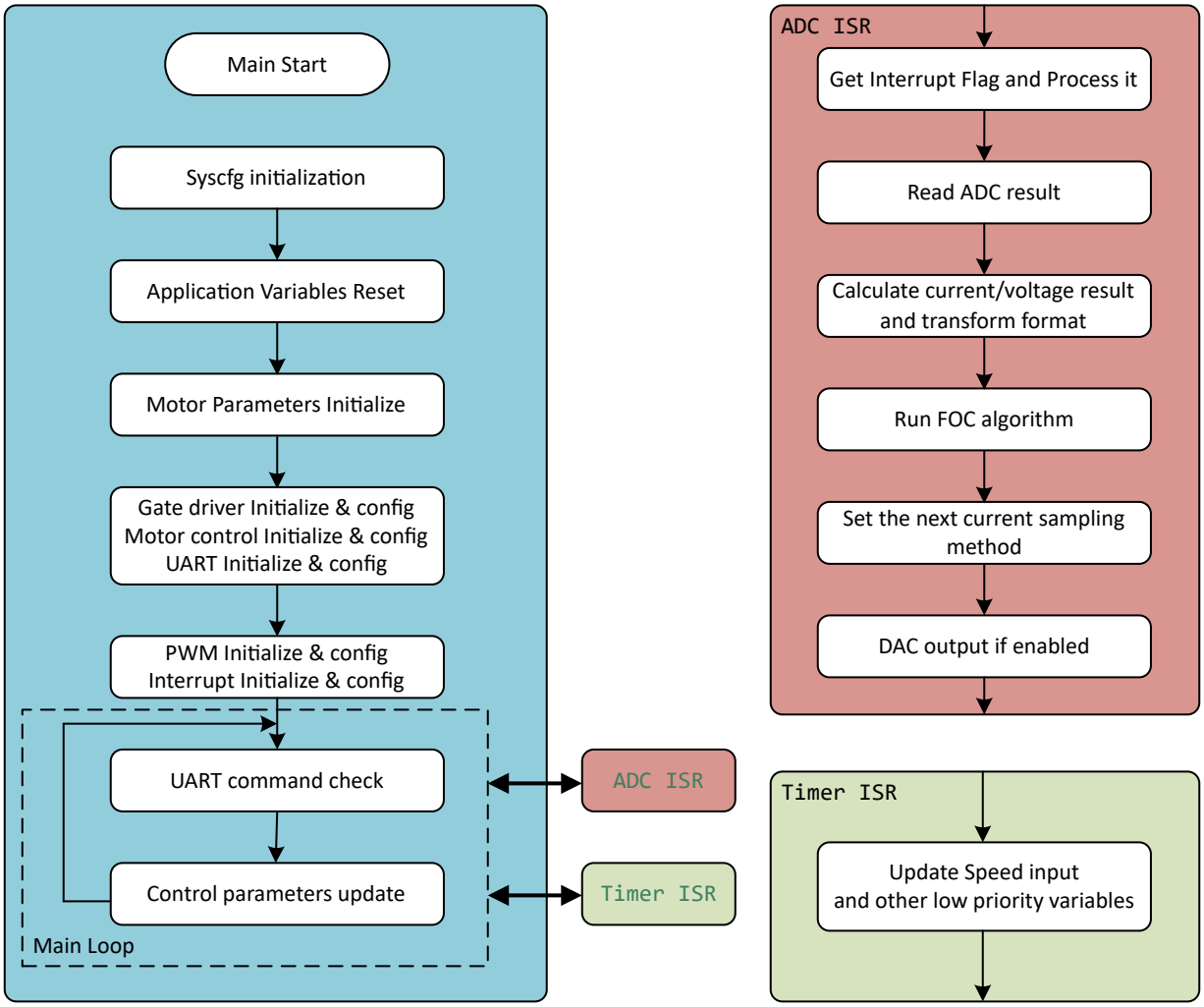


图 5-6. 主应用流程图

5.2.2 HAL 层

硬件抽象层 (HAL) 会创建一个抽象层，以提供用于配置不同引脚和外设的 API。使用 HAL 的目标是抽象处理所有器件特定配置，从而通过更大程度地减少其他元件所需的更新来简化向各种硬件的库移植。HAL 旨在仅抽象处理应用所需的引脚或外设，同时仍然具有移植到其他 MSPM0 MCU 或电机驱动器的灵活性和可扩展性。

下图展示了硬件配置关系。HAL 使用由 TI SysConfig 生成的宏定义。用户在 SysConfig 工具中修改外设实例或 PINMUX 后，HAL 会自动遵循更新后的定义，无需用户更改 HAL 层中的任何代码。

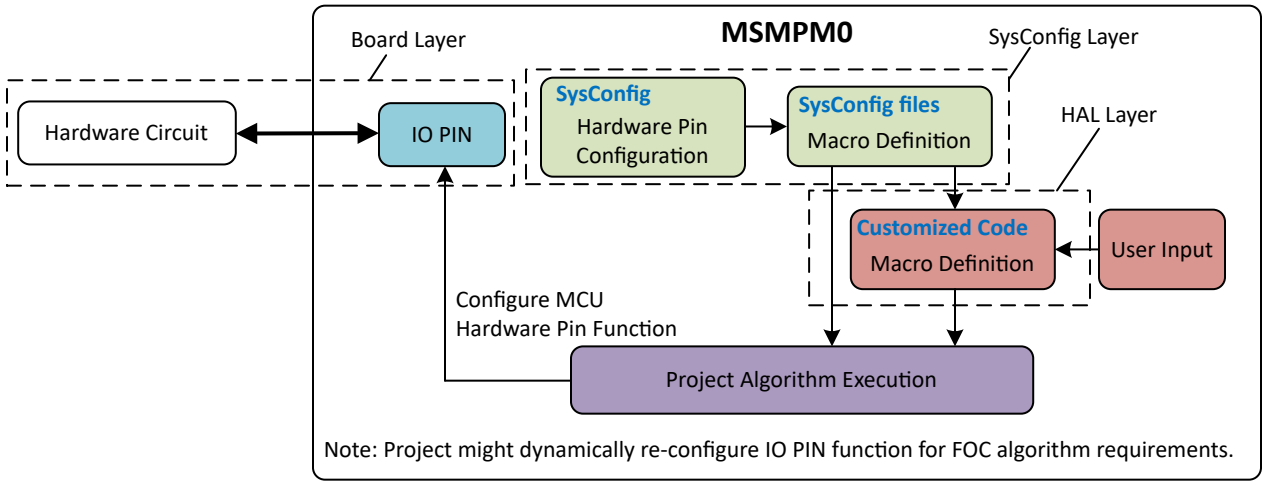


图 5-7. 硬件配置关系

5.2.2.1 栅极驱动器接口

栅极驱动器接口提供用于读取电流和电压通道的 API。此外，如果使用 DRV，该接口会通过 SPI 通信来配置 CSA 增益和其他栅极驱动器寄存器。栅极驱动器接口包括三部分：

- 栅极驱动器库：包含开源 API，用于通过 SPI 配置栅极驱动器寄存器（如果栅极驱动器基于 SPI）
- 栅极驱动器 HAL 接口：处理 ADC 通道的配置，以转换栅极驱动器的相电压输入和相电流输入
- 栅极驱动器宏定义：文件 (gateDriver.h) 中定义了用于读取电流和电压的 ADC 存储器索引

5.2.2.2 电流检测电路

FOC 算法支持使用基于单、双和三分流器的电流检测方法进行电机控制。根据栅极驱动器 EVM 板硬件特性，应用了不同的采样方法。下表提供了不同检测方法的详细信息。

表 5-3. 电流检测方法

栅极驱动器 EVM 板	电流检测方法	反相/非反相电流检测类型
DRV8316REVM	三分流器	反相电流检测
BOOSTXL-DRV8323RS	三分流器	反相电流检测
DRV8329AEVM	单分流器	非反相电流检测
TIDA-010250	双分流器	反相电流检测
TIDA-010251	单分流器	非反相电流检测

在基于三分流器的电流检测方法中，FOC 算法还支持同步采样方法。同步采样方法会动态修改给定空间扇区中的两个采样通道，以便这两个具有更大 PWM 占空比的通道可以同时使用不同的 ADC 实例（ADC0 和 ADC1）进行采样。

在基于双分流器的电流检测方法中，FOC 算法根据采样的两个相电流计算第三个相电流。在成本和电流检测精度方面，它是一种平衡的解决方案。TI 建议使用两个不同的 ADC 实例来对两个相电流进行采样，以便同时对电流进行采样。

在基于单分流器的电流检测方法中，三个相电流是根据来自两个实例的采样直流总线电流估算的。与在双相或三相电流检测中在 PWM 中心进行电流采样不同，在一个 PWM 周期内，在不同的相位开关点对采用单分流器方法的直流总线电流进行两次采样。为了实现此目的，PWM 在 FOC 算法中动态移动，以支持在给定空间扇区中的不同电压矢量上进行采样。为非重叠区域中的电流采样创造了足够的裕度。

5.2.2.3 硬件接口

在应用代码中，使用了多个 GPIO 作为硬件接口来控制电机：

- 制动输入：用于施加制动的引脚输入
- 方向引脚：用于选择方向的引脚输入

- NFAULT：输出引脚，如果检测到任何故障，则设置为低电平
- nSleep：输出引脚，设置栅极驱动器睡眠输入
- 霍尔传感器输入（仅适用于带传感器 FOC）：三相霍尔信号的引脚输入

5.2.2.4 通信接口

通信接口有助于使用 UART、I2C 等通信外设访问器件的存储器。用户可以通过通信外设对 FOC 应用寄存器进行读取或写入，以从外部控制电机。

FOC 工程实现了 UART 通信。有关通信协议的更多详细信息，请参阅 UART 通信用户指南。

5.2.3 MSPM0 Driverlib 层

MSPM0 Driverlib 是一组功能齐全的 API，用于配置、控制和操作 MSPM0 平台的硬件外设。有关更多信息，请参阅 SDK driverlib 文档。

5.3 寄存器映射（无传感器 FOC）

寄存器映射包含一组三个寄存器结构，分别使用用户输入寄存器、用户状态寄存器和用户控制寄存器来设置电机控制调优参数、监测电机状态变量和设置实时控制参数，如表 5-4 所示。

表 5-4. 寄存器概述

寄存器	类型	功能	地址
pUserCtrlRegs	USER_CTRL_INTERFACE_T	一组用户可配置参数，用于实时控制电机	0x20200400
pUserInputRegs	USER_INPUT_INTERFACE_T	一组可配置寄存器，针对各种电机控制特性实时调优电机性能	0x20200000
pUserStatusRegs	USER_STATUS_INTERFACE_T	一组合并变量，用户可用来读取电机状态和分析控制性能	0x20200430

可通过两种方式执行 FOC 寄存器的实时控制：

- 在代码调试期间，将结构导入 CCS 的表达式窗口
- 通过 UART 或 GUI 读取/写入参数

以下章节将介绍与这些无传感器 FOC 结构相关的寄存器和变量。FOC 算法设置寄存器参数的默认值，以对电机进行调优。通过根据应用需求在文件 (configTables.c) 中设置适当的值，可以在源代码中更新默认值。

对于带传感器 FOC 和通用 FOC，它们具有相似的寄存器结构，但支持的特性有所不同。可参阅每个 FOC 算法的具体调优指南来了解详细信息和寄存器差异，如表 5-5 所示。

表 5-5. FOC 调优指南

FOC 算法	FOC 调优指南
无传感器 FOC	MSPM0 无传感器 FOC 调优指南
通用 FOC	MSPM0 通用 FOC 调优用户指南
带传感器 FOC	MSPM0 带传感器 FOC 调优指南

备注

本文档整合了无传感器 FOC v2.04.01 的寄存器映射。请参阅 FOC 算法的具体调优指南中使用较新算法版本更新寄存器映射的内容。

5.3.1 用户控制寄存器（基址 = 0x20200400h）

可在应用代码中使用指针变量 ***pUserCtrlRegs*** 修改这些寄存器组。表 5-6 展示了无传感器 FOC 算法的用户控制寄存器的结构。

表 5-6. 用户控制寄存器

偏移	首字母缩写词	寄存器名称
0h	SPEED_CTRL	速度控制寄存器
4h	ALGO_DEBUG_CTRL1	算法调试控制 1 寄存器
8h	ALGO_DEBUG_CTRL2	算法调试控制 2 寄存器
Ch	ALGO_DEBUG_CTRL3	算法调试控制 3 寄存器
10h	DAC_CTRL	DAC 配置和控制寄存器

复杂的位访问类型经过编码可放入小表格单元格，如表 5-7 所示。

表 5-7. 寄存器配置访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
W	W	写入
复位或默认值		
-n		复位后的值或默认值

5.3.1.1 速度控制寄存器（偏移 = 0h）[复位 = 00000000h]

表 5-8 展示了用以控制电机速度的寄存器。

表 5-8. SPEED_CTRL 寄存器字段说明

位	字段	类型	复位	说明
31 - 15	保留	R	0h	保留
14-0	SPEED_CTRL	W	0000000000 00000b	目标电机转速/扭矩值 速度或扭矩命令的百分比 × 32768

5.3.1.2 算法调试控制 1 寄存器（偏移 = 4h）[复位 = 00000000h]

表 5-9 展示了用于控制算法调试函数的寄存器。

表 5-9. 算法调试控制 1 寄存器字段说明

位	字段	类型	复位	说明
31	CLEAR_FAULT	W	0b	用以清除设置控制器和栅极驱动器故障的位。位会自动复位。 1h = 清除故障命令。
30-0	保留	R	000000 0000b	保留

5.3.1.3 算法调试控制 2 寄存器（偏移 = 8h）[复位 = 00000000h]

表 5-10 展示了用于控制算法调试函数的寄存器。

表 5-10. 算法调试控制 2 寄存器字段说明

位	字段	类型	复位	说明
31	RESERVED	R	0h	保留
30	UPDATE_SYS_PARAMETERS	W	1h	每 200mS 动态更新一次系统参数（例如速度/扭矩/磁通的 PI 增益等），以调优至达到所需性能 0b = 动态系统更新已禁用 1b = 动态系统更新已启用
29	HALL_CALIB_ENABLE	W	0b	用于启用自动霍尔校准。校准完成后，此位会自动复位。 0h = 已禁用/完成霍尔校准 1h = 启用霍尔校准

表 5-10. 算法调试控制 2 寄存器字段说明 （续）

位	字段	类型	复位	说明
28	UPDATE_CONFIGS	R	0b	当算法更新配置时，该位会复位。用户可以在发出调优命令后设置此位，并等待此位复位，然后再启动速度命令。
27	STATUS_UPDATE_ENABLE	W	0b	此位会启用用户状态变量的实时持续更新。
26	CURRENT_LOOP_DIS	W	0b	用于控制 FORCE_VD_CURRENT_LOOP_DIS 和 FORCE_VQ_CURRENT_LOOP_DIS。如果 CURRENT_LOOP_DIS = 1b，则禁用电流环路和速度环路 0h = 启用电流环路 1h = 禁用电流环路
25-16	FORCE_VD_CURRENT_LOOP_DIS	W-IQ(9)	0h	在禁用电流环路和速度环路时以 IQ(9) PU 设置 Vd_ref。如果 CURRENT_LOOP_DIS = 1b，则使用 FORCE_VD_CURRENT_LOOP_DIS 控制 Vd，如果 FORCE_VD_CURRENT_LOOP_DIS < 500，则 Vd_ref = (FORCE_VD_CURRENT_LOOP_DIS / 500)，如果 FORCE_VD_CURRENT_LOOP_DIS > 512，则 Vd_ref = (FORCE_VD_CURRENT_LOOP_DIS - 512) / 500。有效值为：0 至 500 以及 512 至 1000
15-6	FORCE_VQ_CURRENT_LOOP_DIS	W-IQ(9)	0h	在禁用电流环路和速度环路时以 IQ(9) PU 设置 Vq_ref。如果 CURRENT_LOOP_DIS = 1b，则使用 FORCE_VQ_CURRENT_LOOP_DIS 控制 Vq，如果 FORCE_VQ_CURRENT_LOOP_DIS < 500，则 Vq_ref = (FORCE_VQ_CURRENT_LOOP_DIS/500)，如果 FORCE_VQ_CURRENT_LOOP_DIS > 512，则 Vq_ref = (FORCE_VQ_CURRENT_LOOP_DIS - 512) / 500。有效值为：0 至 500 以及 512 至 1000
5-0	RESERVED	R	0h	保留

5.3.1.4 算法调试控制 3 寄存器（偏移 = Ch）[复位 = 00000000h]

表 5-11 展示了用于控制算法调试 3 函数的寄存器。

表 5-11. 算法调试控制 3 寄存器字段说明

位	字段	类型	复位	说明
31-10	RESERVED	R	0h	
9-0	FLUX_MODE_REF	W-IQ(9)	0h	当需要控制电机沿 D 轴的磁通时，以 IQ(9) PU 格式设置 Id_ref 正 Id 控制： 如果 FLUX_MODE_REF < 512，则为 (FLUX_MODE_REF/511) 负 Id 控制： 如果 FLUX_MODE_REF > 512，则为 (FLUX_MODE_REF - 512)/511。有效值为 0 至 511 和 512 至 1000

5.3.1.5 DAC 配置寄存器（偏移 = 10h）[复位 = 00000000h]

DAC 控制寄存器定义了相应的配置，用于使用 MSPM0G 上提供的 12 位 DAC 在示波器上监控实时算法和硬件寄存器数据。有关如何使用 DAC 监测算法变量的详细示例，请参阅*实时变量跟踪*。

表 5-12. DAC 配置寄存器

变量	类型	复位	说明
DAC_EN	无符号短整型 (RW)	0h	0h = 禁用 DAC 1h = 启用 DAC
DAC_SHIFT	短整型 (RW)	0h	+ve 值指定将值加载到 12 位 DAC 寄存器之前的左位移位数。 -ve 值指定在将值加载到 12 位 DAC 寄存器之前的右移位数。 DAC 移位用于监控无符号整数值和寄存器
DAC_SCALING_FACTOR	整数 (RW)	0x00000000h	对于以 IQ 格式表示的数字，通过 DAC 进行监控时，需要使用非零比例因子。为了监控全局 IQ(27) 格式变量，使用 _IQ(1.0) 的 DAC 比例因子。 要表示其他 IQx 格式变量，请将 DAC 比例因子设置为 IQx/IQGlobal。
DACOUT_ADDRESS	无符号整型 (RW)	0x00000000h	定义要通过 DAC 进行监控的 32 位变量的地址。

5.3.2 用户输入寄存器（基址 = 0x20200000h）

可在应用代码中使用指针变量 ***pUserInputRegs*** 修改这些寄存器组。表 5-13 展示了无传感器 FOC 算法的用户输入寄存器的结构。

表 5-13. 用户控制寄存器

偏移	首字母缩写词	寄存器名称
0h	SYSTEM_PARAMETERS	系统参数
38h	ISD_CFG	初始速度检测配置
3Ch	RVS_DRV_CONFIG	反向驱动配置
40h	MOTOR_STARTUP1	电机启动 1 配置
44h	MOTOR_STARTUP2	电机启动 2 配置
48h	CLOSELOOP1	关闭 Loop1 配置
4Ch	CLOSELOOP2	闭环 2 配置
50h	FIELD_CTRL	磁通控制配置
54h	FAULT_CONFIG1	故障配置 1
58h	FAULT_CONFIG2	故障配置 2
5Ch	MISC_ALGO_CONFIG	其他算法配置
60h	PIN_CONFIGURATION	引脚配置
64h	PERI_CONFIG1	外设配置 1

复杂的位访问类型经过编码可放入小表格单元格，如下所示。

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
W	W	写入
复位或默认值		
-n		复位后的值或默认值

5.3.2.1 SYSTEM_PARAMETERS（偏移 = 0h）

这些表显示了电机控制系统功能所必需的一组基本系统配置参数。

表 5-14. 电机电阻配置寄存器（偏移 = 0h）

位	字段	类型	复位	说明
31-0	MTR_RESISTANCE	R/W	0000h	以毫欧为单位的电机电阻

表 5-15. 电机电感配置（偏移 = 4h）

位	字段	类型	复位	说明
31-0	MTR_INDUCTANCE	R/W	0000h	以微亨为单位的电机电感。对于凸极电机 (Lq + Ld)/2

表 5-16. 电机凸极配置（偏移 = 8h）

位	字段	类型	复位	说明
31-0	MTR_SALIENCY	R/W	0.0（浮点）	以浮点数表示电机的凸极 (Lq-Ld)/(Lq+Ld)。

表 5-17. 电机 BEMF 常数配置（偏移 = Ch）

位	字段	类型	复位	说明
31-0	MTR_BEMF_CONSTANT	R/W	0000h	电机 BEMF 常数，单位为 mV/Hz × 10。

表 5-18. 基极电压配置（偏移 = 10h）

位	字段	类型	复位	说明
31-0	VOLTAGE_BASE	R/W	0.0（浮点）	该电路板的基极电压基于分压器计算得出（3.3V × 分压比 / √3），单位为伏特。3.3V 是 ADC 的满量程值。

表 5-19. 基极电流配置（偏移 = 14h）

位	字段	类型	复位	说明
31-0	CURRENT_BASE	R/W	0.0（浮点）	该电路板的基极电流是根据 CSA 增益计算得出的（1.65V/CSA 增益，单位为伏/安培），单位为安培。 1.65V 是 ADC 用于双向电流检测的基准中点电压。 如果 CSA 增益以 V/V 为单位，则将乘以电流检测电阻值（以欧姆为单位）来计算 CSA 增益（以伏/安为单位）

表 5-20. 电机最大速度配置（偏移 = 18h）

位	字段	类型	复位	说明
31-0	MOTOR_MAX_SPEED	R/W	0.0（浮点）	数据表中以 Hz 为单位的额定电机转速

表 5-21. 电机最大功率配置（偏移 = 1Ch）

位	字段	类型	复位	说明
31-0	MOTOR_MAX_POWER	R/W	0.0（浮点）	电机的最大额定功率。 **备注** FOC 算法计算 PU 功率，该功率用于控制闭环中的输入功率。 **备注** PU 功率定义为 MOTOR_MAX_POWER/ √3 * VOLTAGE_BASE * CURRENT_BASE

表 5-22. 速度环路比例增益（偏移 = 20h）

位	字段	类型	复位	说明
31-0	SPEED_POWER_LOOP_KP	R/W	0.0（浮点）	以浮点方式进行闭环速度控制/闭环功率控制的比例增益

表 5-23. 速度环路积分增益（偏移 = 24h）

位	字段	类型	复位	说明
31-0	SPEED_POWER_LOOP_KI	R/W	0.0（浮点）	以浮点方式进行闭环速度控制/闭环功率控制的积分增益

表 5-24. 扭矩环路比例增益（偏移 = 28h）

位	字段	类型	复位	说明
31-0	CURR_LOOP_KP	R/W	0.0（浮点）	以浮点方式进行闭环扭矩控制的比例增益

表 5-25. 扭矩环路积分增益（偏移 = 2Ch）

位	字段	类型	复位	说明
31-0	CURR_LOOP_KI	R/W	0.0（浮点）	以浮点方式进行闭环扭矩控制的积分增益

表 5-26. 弱磁控制器比例增益（偏移 = 30h）

位	字段	类型	复位	说明
31-0	FLUX_WEAK_KP	R/W	0.0（浮点）	以浮点方式进行弱磁控制的比例增益

表 5-27. 弱磁控制器积分增益（偏移 = 34h）

位	字段	类型	复位	说明
31-0	FLUX_WEAK_KI	R/W	0.0（浮点）	以浮点方式进行弱磁控制的积分增益

表 5-28. 电机极对数（偏移 = 38h）

位	字段	类型	复位	说明
31-0	POLE_PAIRS	R/W	0(Int)	总电机极对数或极数/2。

5.3.2.2 MOTOR_STARTUP1 寄存器（偏移 = 3Ch）[复位 = 00000000h]

表 5-29 展示了用于配置电机启动设置 1 的寄存器。

表 5-29. MOTOR_STARTUP1 寄存器字段说明

位	字段	类型	复位	说明
31-30	RESERVED	R	00b	保留
29-26	CURR_RAMP_RATE	R/W	0h	霍尔校准期间的初始电流斜升速率，直至达到最大电流。 0h = 0.1A/s 1h = 1A/s 2h = 5A/s 3h = 10A/s 4h = 15A/s 5h = 25A/s 6h = 50A/s 7h = 100A/s 8h = 150A/s 9h = 200A/s Ah = 250A/s Bh = 500A/s Ch = 1000A/s Dh = 2000A/s Eh = 5000A/s Fh = 无限值 A/s

表 5-29. MOTOR_STARTUP1 寄存器字段说明（续）

位	字段	类型	复位	说明
25-22	CALIB_RUN_TIME	R/W	0h	针对每个 CALIBRATION_ANGLE_STEP（在 hallCalib.h 中定义）校准霍尔时花费的时间 0h = 10ms 1h = 50ms 2h = 100ms 3h = 200ms 4h = 300ms 5h = 400ms 6h = 500ms 7h = 750ms 8h = 1s 9h = 1.5s Ah = 2s Bh = 3s Ch = 4s Dh = 5s Eh = 7.5s Fh = 10s
21-17	CALIB_CURRENT_ILIMIT	R/W	00h	霍尔校准期间的电流限制，以 CURRENT_BASE 的百分比表示 0h = 7.5% 1h = 8.0% 2h = 8.5% 3h = 9.0% 4h = 9.5% 5h = 10% 6h = 11% 7h = 12% 8h = 13% 9h = 14% Ah = 15% Bh = 16% Ch = 17% Dh = 18% Eh = 20% Fh = 22.5% 10h = 25% 11h = 27.5% 12h = 30% 13h = 35% 14h = 40% 15h = 45% 16h = 50% 17h = 55% 18h = 60% 19h = 70% 1Ah = 75% 1Bh = 80% 1Ch = 85% 1Dh = 90% 1Eh = 95% 1Fh = 100%

表 5-29. MOTOR_STARTUP1 寄存器字段说明（续）

位	字段	类型	复位	说明
16-13	CALIB_ALIGN_TIME	R	000b	开始霍尔校准之前，转子初始对齐到零度所花的时间。 0h = 10ms 1h = 50ms 2h = 100ms 3h = 200ms 4h = 300ms 5h = 400ms 6h = 500ms 7h = 750ms 8h = 1s 9h = 1.5s Ah = 2s Bh = 3s Dh = 5s Eh = 7.5s Fh = 10s
12-2	RESERVED	R	0h	保留
1	OL_ILIMIT_CONFIG	R/W	0b	开环电流限值配置 0h = 由 OL_ILIMIT 定义的开环电流限值 1h = 由 ILIMIT 定义的开环电流限值
0	RESERVED	R	0b	保留

5.3.2.3 MOTOR_STARTUP2 寄存器（偏移 = 40h）[复位 = 00000000h]

表 5-30 展示了用于配置电机启动设置 2 的寄存器。

表 5-30. MOTOR_STARTUP2 寄存器字段说明

位	字段	类型	复位	说明
31-27	OL_ILIMIT	R/W	0h	开环电流限制（占 CURRENT_BASE 的百分比） 0h = 7.5% 1h = 8.0% 2h = 8.5% 3h = 9.0% 4h = 9.5% 5h = 10% 6h = 11% 7h = 12% 8h = 13% 9h = 14% Ah = 15% Bh = 16% Ch = 17% Dh = 18% Eh = 20% Fh = 22.5% 10h = 25% 11h = 27.5% 12h = 30% 13h = 35% 14h = 40% 15h = 45% 16h = 50% 17h = 55% 18h = 60% 19h = 70% 1Ah = 75% 1Bh = 80% 1Ch = 85% 1Dh = 90% 1Eh = 95% 1Fh = 100%
26-23	OL_ACC_A1	R/W	0h	开环加速系数 A1 0h = 0.01Hz/s 1h = 0.05Hz/s 2h = 1Hz/s 3h = 2.5Hz/s 4h = 5Hz/s 5h = 10Hz/s 6h = 25Hz/s 7h = 50Hz/s 8h = 75Hz/s 9h = 100Hz/s Ah = 250Hz/s Bh = 500Hz/s Ch = 750Hz/s Dh = 1000Hz/s Eh = 5000Hz/s Fh = 10000Hz/s

表 5-30. MOTOR_STARTUP2 寄存器字段说明（续）

位	字段	类型	复位	说明
22-19	OL_ACC_A2	R/W	0h	开环加速系数 A2 0h = 0.0Hz/s2 1h = 0.05Hz/s2 2h = 1Hz/s2 3h = 2.5Hz/s2 4h = 5Hz/s2 5h = 10Hz/s2 6h = 25Hz/s2 7h = 50Hz/s2 8h = 75Hz/s2 9h = 100Hz/s2 Ah = 250Hz/s2 Bh = 500Hz/s2 Ch = 750Hz/s2 Dh = 1000Hz/s2 Eh = 5000Hz/s2 Fh = 10000Hz/s2
18	RESERVED	R/W	0h	保留
17-13	OL_MAX_SPEED	R/W	0h	在强制换向期间，第一个电气周期的最大工作电气频率（占 MAX_SPEED 的百分比） 0h = 1% 1h = 2% 2h = 3% 3h = 4% 4h = 5% 5h = 6% 6h = 7% 7h = 8% 8h = 9% 9h = 10% Ah = 11% Bh = 12% Ch = 13% Dh = 14% Eh = 15% Fh = 16% 10h = 17% 11h = 18% 12h = 19% 13h = 20% 14h = 22.5% 15h = 25% 16h = 27.5% 17h = 30% 18h = 32.5% 19h = 35% 1Ah = 37.5% 1Bh = 40% 1Ch = 42.5% 1Dh = 45% 1Eh = 47.5% 1Fh = 50%
12-8	保留	R	0h	保留

表 5-30. MOTOR_STARTUP2 寄存器字段说明（续）

位	字段	类型	复位	说明
7-4	OL_FIRST_CYC_FREQ	R/W	0h	开环中第一个周期的频率（占 MAX_SPEED 的百分比） 0h = 1% 1h = 2% 2h = 3% 3h = 5% 4h = 7.5% 5h = 10% 6h = 12.5% 7h = 15% 8h = 17.5% 9h = 20% Ah = 25% Bh = 30% Ch = 35% Dh = 40% Eh = 45% Fh = 50%
3	FIRST_CYCLE_FREQ_SEL	R/W	0h	开环中第一个周期的频率 0h = 由 OL_FIRST_CYC_FREQ 定义 1h = 0Hz
2-0	RESERVED	R	0h	保留

5.3.2.4 CLOSED_LOOP1 寄存器（偏移 = 44h）[复位 = 00000000h]

表 5-31 展示了用于配置闭环设置 1 的寄存器。

表 5-31. CLOSED_LOOP1 寄存器字段说明

位	字段	类型	复位	说明
31-30	RESERVED	R/W	0h	保留
29-28	CONTROL_MODE	R/W	0h	FOC 闭环运行模式 0h = 闭环速度控制 1h = 闭环功率控制 2h = 闭环扭矩控制 3h = 电压控制模式
27	HIGH_FREQ_FOC_EN	R/W	0b	启用/禁用高 FOC 采样率。采样率越高，可用于其他任务的 CPU 带宽就越低。 0h = 高频 FOC 启用（最大 FOC 频率 16Khz） 1h = 高频 FOC 禁用（最大 FOC 频率 8Khz）

表 5-31. CLOSED_LOOP1 寄存器字段说明（续）

位	字段	类型	复位	说明
26-22	ILIMIT	R/W	0h	闭环扭矩模式和闭环速度控制下的电流限制（占 CURRENT_BASE 的百分比） 0h = 7.5% 1h = 8.0% 2h = 8.5% 3h = 9.0% 4h = 9.5% 5h = 10% 6h = 11% 7h = 12% 8h = 13% 9h = 14% Ah = 15% Bh = 16% Ch = 17% Dh = 18% Eh = 20% Fh = 22.5% 10h = 25% 11h = 27.5% 12h = 30% 13h = 35% 14h = 40% 15h = 45% 16h = 50% 17h = 55% 18h = 60% 19h = 70% 1Ah = 75% 1Bh = 80% 1Ch = 85% 1Dh = 90% 1Eh = 95% 1Fh = 100%
21-20	MTR_STOP	R/W	00b	电机停止方法 0h = 高阻态 1h = 主动降速 2h = 制动 3h = 保留
19	OVERMODULATION_ENABLE	R/W	0b	过调制启用 0h = 禁用过调制 1h = 启用过调制

表 5-31. CLOSED_LOOP1 寄存器字段说明（续）

位	字段	类型	复位	说明
18-14	CL_ACC	R/W	0h	闭环加速 0h = 0.5Hz/s 1h = 1Hz/s 2h = 2.5Hz/s 3h = 5Hz/s 4h = 7.5Hz/s 5h = 10Hz/s 6h = 20Hz/s 7h = 40Hz/s 8h = 60Hz/s 9h = 80Hz/s Ah = 100Hz/s Bh = 200Hz/s Ch = 300Hz/s Dh = 400Hz/s Eh = 500Hz/s Fh = 600Hz/s 10h = 700Hz/s 11h = 800Hz/s 12h = 900Hz/s 13h = 1000Hz/s 14h = 2000Hz/s 15h = 4000Hz/s 16h = 6000Hz/s 17h = 8000Hz/s 18h = 10000Hz/s 19h = 20000Hz/s 1Ah = 30000Hz/s 1Bh = 40000Hz/s 1Ch = 50000Hz/s 1Dh = 60000Hz/s 1Eh = 70000Hz/s 1Fh = 无限值
13	CL_DEC_CONFIG	R/W	0h	闭环减速配置 0h = 由 CL_DEC 定义的闭环减速 1h = 由 CL_ACC 定义的闭环减速

表 5-31. CLOSED_LOOP1 寄存器字段说明（续）

位	字段	类型	复位	说明
12-8	CL_DEC	R/W	0h	闭环减速。仅当 AVS 被禁用且 CL_DEC_CONFIG 被设置为“0”时，才使用该寄存器 0h = 0.5Hz/s 1h = 1Hz/s 2h = 2.5Hz/s 3h = 5Hz/s 4h = 7.5Hz/s 5h = 10Hz/s 6h = 20Hz/s 7h = 40Hz/s 8h = 60Hz/s 9h = 80Hz/s Ah = 100Hz/s Bh = 200Hz/s Ch = 300Hz/s Dh = 400Hz/s Eh = 500Hz/s Fh = 600Hz/s 10h = 700Hz/s 11h = 800Hz/s 12h = 900Hz/s 13h = 1000Hz/s 14h = 2000Hz/s 15h = 4000Hz/s 16h = 6000Hz/s 17h = 8000Hz/s 18h = 10000Hz/s 19h = 20000Hz/s 1Ah = 30000Hz/s 1Bh = 40000Hz/s 1Ch = 50000Hz/s 1Dh = 60000Hz/s 1Eh = 70000Hz/s 1Fh = 无限值
7-8	PWM_FREQ_OUT	R/W	0h	输出 PWM 开关频率 0h = 5kHz 1h = 10kHz 2h = 16kHz 3h = 20kHz 4h = 25kHz 5h = 32kHz 6h = 40kHz 7h = 48kHz 8h = 50kHz 9h = 64kHz Ah = 80kHz Bh = 不适用 Ch = 不适用 Dh = 不适用 Eh = 不适用 Fh = 不适用

表 5-31. CLOSED_LOOP1 寄存器字段说明（续）

位	字段	类型	复位	说明
14	PWM_MODE	R/W	0b	PWM 调制 0h = 连续空间矢量调制 1h = 不连续空间矢量调制
3	AVS_EN	R/W	0b	AVS 启用 0h = 禁用 1h = 启用
2	DEADTIME_COMP_EN	R/W	0b	死区时间补偿启用 0h = 禁用 1h = 启用
1	RESERVED	R/W	0b	保留

5.3.2.5 CLOSED_LOOP2 寄存器（偏移 = 48h）[复位 = 00000000h]

表 5-32 展示了用于配置闭环设置 2 的寄存器。

表 5-32. CLOSED_LOOP2 寄存器字段说明

位	字段	类型	复位	说明
31-28	ACT_SPIN_THR	R/W	0h	主动降速的速度阈值（占 MAX_SPEED 的百分比） 0h = 100% 1h = 90% 2h = 80% 3h = 70% 4h = 60% 5h = 50% 6h = 45% 7h = 40% 8h = 35% 9h = 30% Ah = 25% Bh = 20% Ch = 15% Dh = 10% Eh = 5% Fh = 2.5%
27-24。	BRAKE_SPEED_ THRESHOLD	R/W	0h	BRAKE 引脚和电机停止选项（低侧制动、高侧制动或对齐制动）的速度阈值（占 MAX_SPEED 的百分比） 0h = 100% 1h = 90% 2h = 80% 3h = 70% 4h = 60% 5h = 50% 6h = 45% 7h = 40% 8h = 35% 9h = 30% Ah = 25% Bh = 20% Ch = 15% Dh = 10% Eh = 5% Fh = 2.5%

表 5-32. CLOSED_LOOP2 寄存器字段说明（续）

位	字段	类型	复位	说明
23-19	BRK_CURR_THR	R/W	0h	制动电流限值（占 CURRENT_BASE 的百分比） 0h = 7.5% 1h = 8.0% 2h = 8.5% 3h = 9.0% 4h = 9.5% 5h = 10% 6h = 11% 7h = 12% 8h = 13% 9h = 14% Ah = 15% Bh = 16% Ch = 17% Dh = 18% Eh = 20% Fh = 22.5% 10h = 25% 11h = 27.5% 12h = 30% 13h = 35% 14h = 40% 15h = 45% 16h = 50% 17h = 55% 18h = 60% 19h = 70% 1Ah = 75% 1Bh = 80% 1Ch = 85% 1Dh = 90% 1Eh = 95% 1Fh = 100%
18-14	LEAD_ANGLE	R/W	0h	电压控制模式下应用的超前角（以度为单位） 0 - 15 = 1 * 位值 15 - 31 = 2 * (位值 - 15) + 15
13 - 0	RESERVED	R	0h	保留

5.3.2.6 FIELD_CTRL 寄存器（偏移 = 4Ch）[复位 = 00000000h]

用于配置磁通控制设置的寄存器

表 5-33. FIELD_CTRL 寄存器位说明

位	字段	类型	复位	说明
31-7	保留	R	0h	保留
6	MTPA_EN	R/W	0b	启用/禁用每安培最大扭矩控制 (MTPA) 0h = 禁用 MTPA 1h = 启用 MTPA
5-4	FLUX_WEAK_REF	R/W	00b	在弱磁模式下跟踪的调制指数基准 0h = 70% 1h = 80% 2h = 90% 3h = 95%

表 5-33. FIELD_CTRL 寄存器位说明（续）

位	字段	类型	复位	说明
3-1	FLUX_WEAK_CURR_RATIO	R/W	000b	弱磁电流基准的最大值（以 ILIMIT 百分比表示） 0h = 仅存在循环限制 1h = 80% 2h = 70% 3h = 60% 4h = 50% 5h = 40% 6h = 30% 7h = 20%
0	FLUX_WEAK_EN	R/W	0b	启用/禁用弱磁控制 (MTPA) 0h = 禁用弱磁 1h = 启用弱磁

5.3.2.7 FAULT_CONFIG1 寄存器（偏移 = 50h）[复位 = 00000000h]

用于配置故障设置 1 的寄存器

表 5-34. FAULT_CONFIG1 寄存器字段说明

位	字段	类型	复位	说明
31-6	保留	R/W	0h	保留
5-2	LCK_RETRY	R/W	0h	锁定检测重试时间 0h = 100ms 1h = 500ms 2h = 1s 3h = 2s 4h = 3s 5h = 4s 6h = 5s 7h = 6s 8h = 7s 9h = 8s Ah = 9s Bh = 10s Ch = 11s Dh = 12s Eh = 13s Fh = 14s
1-0	MTR_LCK_MODE	R/W	00b	电机锁定模式 0h = 电机锁定检测导致锁存故障；nFAULT 有效； 1h = 在 LCK_RETRY 时间后自动清除故障 2h = 电机锁定处于仅报告模式 3h = 禁用电机锁定检测

5.3.2.8 FAULT_CONFIG2 寄存器（偏移 = 54h）[复位 = 00000000h]

表 5-35 展示了用于配置故障设置 2 的寄存器。

表 5-35. FAULT_CONFIG2 寄存器字段说明

位	字段	类型	复位	说明
31-27	RESERVED	R/W	0h	保留
26	ABN_SPEED_LOCK_EN	R/W	0b	锁定 1：异常速度锁定 0h = 禁用 1h = 启用

表 5-35. FAULT_CONFIG2 寄存器字段说明（续）

位	字段	类型	复位	说明
25	HALL_INVALID_LOCK_EN	R/W	0b	锁定 2：无效霍尔输入锁定 0h = 禁用 1h = 启用
24	NO_MOTOR_LOCK_EN	R/W	0b	锁定 3：无电机锁定启用 0h = 禁用 1h = 启用
23-21	LOCK_ABN_SPEED	R/W	000b	异常速度锁定阈值（占 MAX_SPEED 的百分比） 0h = 130% 1h = 140% 2h = 150% 3h = 160% 4h = 170% 5h = 180% 6h = 190% 7h = 200%
20-18	RESERVED	R	0b	保留
17-13	NO_MTR_THR	R/W	00000b	无电机电流限值（占 CURRENT_BASE 的百分比） 0h = 7.5% 1h = 8.0% 2h = 8.5% 3h = 9.0% 4h = 9.5% 5h = 10% 6h = 11% 7h = 12% 8h = 13% 9h = 14% Ah = 15% Bh = 16% Ch = 17% Dh = 18% Eh = 20% Fh = 22.5% 10h = 25% 11h = 27.5% 12h = 30% 13h = 35% 14h = 40% 15h = 45% 16h = 50% 17h = 55% 18h = 60% 19h = 70% 1Ah = 75% 1Bh = 80% 1Ch = 85% 1Dh = 90% 1Eh = 95% 1Fh = 100%
12-8	RESERVED	R/W	0h	保留。

表 5-35. FAULT_CONFIG2 寄存器字段说明（续）

位	字段	类型	复位	说明
7-5	MIN_VM_MOTOR	R/W	000b	运行电机的最小电压（占 BASE_VOLTAGE 的百分比） 0h = 无限值 1h = 5% 2h = 10% 3h = 12% 4h = 15% 5h = 18% 6h = 20% 7h = 25%
4	MIN_VM_MODE	R/W	0b	欠压故障模式 0h = 欠压锁存 1h = 如果电压处于界定范围之内，则自动清除
3-1	MAX_VM_MOTOR	R/W	000b	运行电机的最大电压（占 BASE_VOLTAGE 的百分比） 0h = 60% 1h = 65% 2h = 70% 3h = 75% 4h = 80% 5h = 85% 6h = 90% 7h = 最大电压
0	MAX_VM_MODE	R/W	0b	过压故障模式 0h = 过压锁存 1h = 如果电压处于界定范围之内，则自动清除

5.3.2.9 MISC_ALGO 寄存器（偏移 = 58h）[复位 = 00000000h]

表 5-36 展示了用于多种杂项算法配置的寄存器。

表 5-36. MISC_ALGO 寄存器字段说明

位	字段	类型	复位	说明
31-10	RESERVED	R/W	0h	保留
9-6	CL_SLOW_ACC	R/W	0h	估算器尚未完全对齐时的闭环加速 0h = 0.1Hz/s 1h = 1Hz/s 2h = 2Hz/s 3h = 3Hz/s 4h = 5Hz/s 5h = 10Hz/s 6h = 20Hz/s 7h = 30Hz/s 8h = 40Hz/s 9h = 50Hz/s Ah = 100Hz/s Bh = 200Hz/s Ch = 500Hz/s Dh = 750Hz/s Eh = 1000Hz/s Fh = 2000Hz/s
5-2	RESERVED	R	0b	保留

表 5-36. MISC_ALGO 寄存器字段说明（续）

位	字段	类型	复位	说明
1-0	BRAKE_CURRENT_PERSIST	R/W	00b	制动期间电流低于阈值的持续时间 0h = 50ms 1h = 100ms 2h = 250ms 3h = 500ms

5.3.2.10 PIN_CONFIG 寄存器（偏移 = 5Ch）[复位 = 00000000h]

表 5-37 展示了用于配置硬件引脚的寄存器。

表 5-37. PIN_CONFIG 寄存器字段说明

位	字段	类型	复位	说明
31-20	保留	R/W	0h	保留
19	VDC_FILT_DIS	R/W	0b	Vdc 滤波器禁用 0h = 启用 1h = 禁用
18-3	保留	R/W	0h	保留
2	BRAKE_PIN_MODE	R/W	0b	制动引脚模式 0h = 低侧制动 1h = 对齐制动
1-0	BRAKE_INPUT	R/W	00b	制动引脚覆盖 0h = 硬件引脚制动 1h = 根据 BRAKE_PIN_MODE 覆盖引脚和制动/对齐 2h = 覆盖引脚，不制动/对齐 3h = 硬件引脚制动

5.3.2.11 PERI_CONFIG 寄存器（偏移 = 60h）[复位 = 00000000h]

表 5-38 展示了外设的寄存器。

表 5-38. PERI_CONFIG1 寄存器字段说明

位	字段	类型	复位	说明
31-15	RESERVED	R	0h	保留
14-9	MCU_DEAD_TIME	R/W	0h	高侧和低侧开关之间应用的死区时间 = 50ns × MCU_DEAD_TIME

表 5-38. PERI_CONFIG1 寄存器字段说明（续）

位	字段	类型	复位	说明
8-4	BUS_CURRENT_LIMIT	R/W	00000b	总线电流限值（占 CURRENT_BASE 的百分比） 0h = 7.5% 1h = 8.0% 2h = 8.5% 3h = 9.0% 4h = 9.5% 5h = 10% 6h = 11% 7h = 12% 8h = 13% 9h = 14% Ah = 15% Bh = 16% Ch = 17% Dh = 18% Eh = 20% Fh = 22.5% 10h = 25% 11h = 27.5% 12h = 30% 13h = 35% 14h = 40% 15h = 45% 16h = 50% 17h = 55% 18h = 60% 19h = 70% 1Ah = 75% 1Bh = 80% 1Ch = 85% 1Dh = 90% 1Eh = 95% 1Fh = 100%
3	BUS_CURRENT_LIMIT_ENABLE	R/W	0b	总线电流限制启用 0h = 禁用 1h = 启用
2-1	DIR_INPUT	R/W	00b	DIR 引脚覆盖 0h = 硬件引脚 DIR 1h = 通过顺时针旋转 OUTA-OUTB-OUTC 覆盖 DIR 引脚 2h = 通过逆时针旋转 OUTA-OUTC-OUTB 覆盖 DIR 引脚 3h = 硬件引脚 DIR
0	DIR_CHANGE_MODE	R/W	0b	对 DIR 引脚状态变化的响应 0h = 在检测到 DIR 变化时遵循电机停止选项和 ISD 例程 1h = 在持续驱动电机的同时通过反向驱动改变方向

5.3.3 用户状态寄存器（基址 = 0x20200430h）

可在应用代码中使用指针变量 ***pUserStatusRegs*** 修改这些寄存器组。表 4-29 展示了无传感器 FOC 算法的用户状态寄存器的结构和定义。用户状态寄存器会在周期性计时器中断中每 1ms 更新一次。

表 5-39. 用户状态寄存器

变量	复位值	说明
SYSTEM_FAULT_STATUS	NO_FAULT	定义电机故障的状态。 **MOTOR_STALL**：指示电机锁定故障 — 异常 BEMF、无电机、异常速度 **VOLTAGE_OUT_OF_BOUNDS**：指示欠压或过压 **LOAD_STALL**：指示 IPD 故障。 HARDWARE_OVER_CURRENT：指示直流总线电流限制故障 **HARDWARE_OVER_CURRENT**：指示直流总线中存在过流 **HV_DIE**：如果适用，指示栅极驱动器故障（nFAULT 输入引脚）
MOTOR_STATE	MOTOR_IDLE	定义当前电机运行状态 **MOTOR_IDLE**：电机空闲状态 **MOTOR_ISD**：电机处于初始速度检测状态 **MOTOR_TRISTATE**：电机处于三态或高组态模式 **MOTOR_BRAKE_ON_START**：启动期间的电机制动 **MOTOR_IPD**：电机处于初始位置检测状态 **MOTOR_SLOW_FIRST_CYCLE**：电机处于慢速首循环启动状态 **MOTOR_ALIGN**：电机处于对齐启动状态 **MOTOR_OPEN_LOOP**：电机处于开环斜升状态。 **MOTOR_CLOSE_LOOP_UNALIGNED**：电机处于闭环运行状态且角度未对齐 **MOTOR_CLOSE_LOOP_ALIGNED**：电机处于闭环运行状态且角度对齐 **MOTOR_SOFT_STOP**：电机处于停止状态 **MOTOR_BRAKE_ON_STOP**：电机处于制动停止状态 **MOTOR_FAULT**：电机处于电机故障状态
V_DQ_FILT	IQ27(0)	指示施加于电机的滤波 Vd 和 Vq。电流 PI 控制器的输出。
I_DQ_PI	IQ27(0)	指示电流 PI 控制器的 Kp 和 Ki 值。
PI_SPEED	IQ27(0)	指示由 FOC 算法以 PU 设置的速度 PI 控制器的基准值和反馈值。
PI_POWER	IQ27(0)	指示由 FOC 算法以 PU 设置的功率 PI 控制器的基准值和反馈值。
PI_ID	IQ27(0)	指示由 FOC 算法以 PU 设置的直流 PI 控制器的基准值和反馈值。
PI_IQ	IQ27(0)	指示由 FOC 算法以 PU 设置的正交电流 PI 控制器的基准值和反馈值。
IPD_IDENTIFIED_SECTOR	0b	指示 IPD 识别的最近转子状态。
ESTIMATED_SPEED	IQ27(0)	指示由 FOC 观测器算法估算的电机转速，以 PU 为单位。
DC_BUS_VOLTAGE	IQ27(0)	指示以 PU 为单位的直流总线电压矢量值
TORQUE_LIMIT	IQ27(0)	指示由 FOC 设置的正交电流控制器饱和限制。此值基于 ClosedLoop1 配置中设置的限值。
GATE_DRIVER_FAULT_STATUS	0x00000000h	定义在 gateDriverLib 中定义的栅极驱动器特定故障的索引。
CONTROLLER_FAULT_STATUS	0x00000000h	定义在 main.h 中定义的 FOC 控制算法特定故障的索引
APP_VERSION	0x00000000h	定义应用固件的版本号

6 快速入门指南

TI 提供用于评估 MSPM0 Arm Cortex-M0+ 微控制器的 LaunchPad™ 开发套件，以及用于评估 DRV83xx 系列无刷直流电机驱动器的评估模块 (EVM)。这些评估板可从 TI.COM 上获取，并可用作无传感器、通用或带传感器 FOC 电机控制的系统评估平台。有关与所选 DRV83xx EVM 板的硬件连接，请参阅下表。

表 6-1. LaunchPad 和 EVM 板的硬件连接指南

FOC 算法	SDK 用户指南(1)
无传感器 FOC	无传感器 FOC SDK 用户指南
通用 FOC	通用 FOC SDK 用户指南
带传感器 FOC	带传感器 FOC SDK 用户指南

(1) 包括库概述、软件设置、硬件设置等。

6.1 CCS IDE

6.1.1 工程设置

按照以下步骤在 CCS 中设置调试环境。

1. 转到 Import CCS Projects。

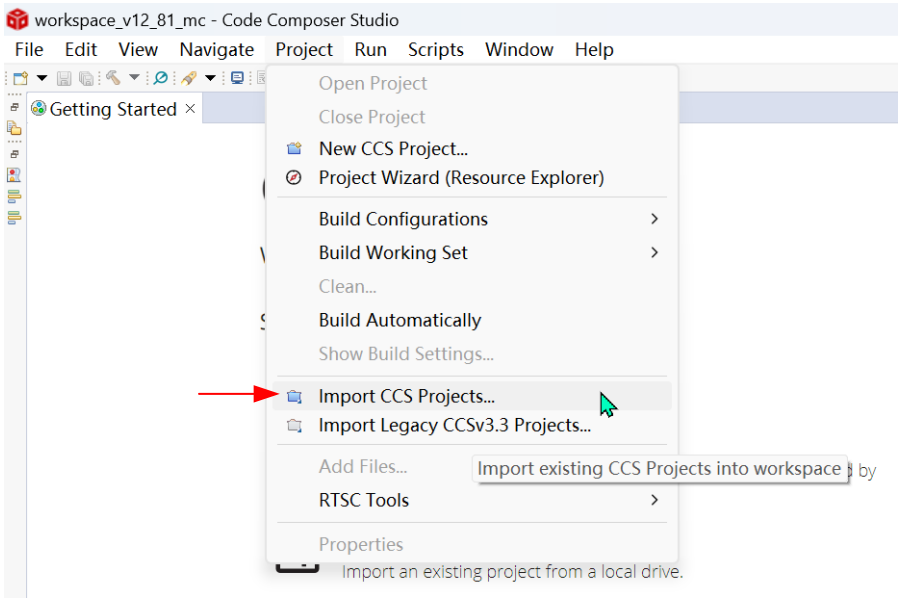


图 6-1. 导入 CCS 工程

2. 点击 Browse，然后导航至以下地址：

```
C:\ti\mspm0_sdk_<SDK_Version>\examples\nortos\[LAUNCHPAD]\motor_control_xxxx_xxxx_foc
```

图 6-2 显示了该流程：

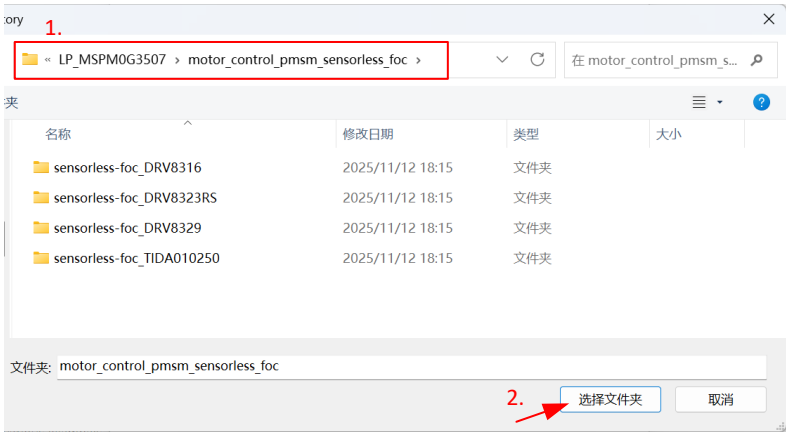


图 6-2. 导航至 SDK 工程

3. 点击 Select Folder。检查所需的硬件板，然后点击 Finish 将工程导入您的工作区。

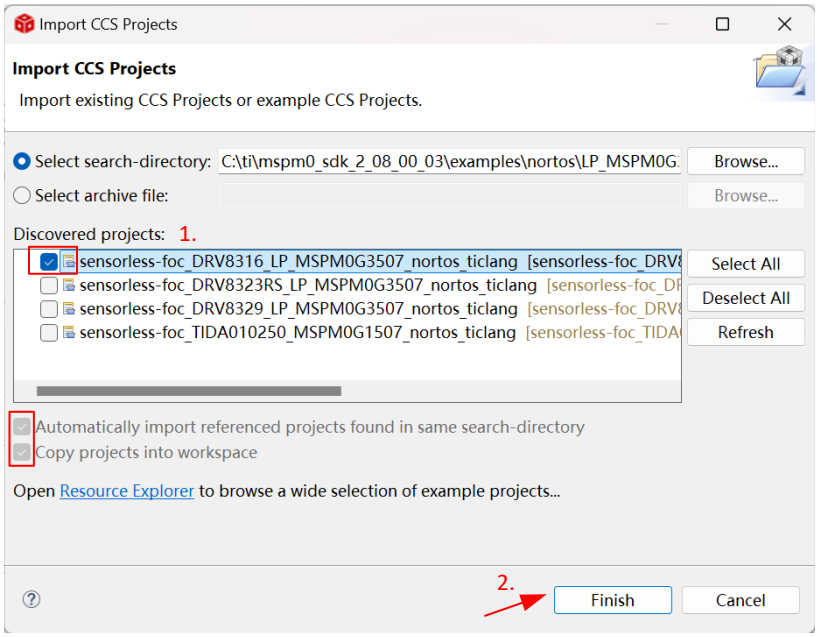


图 6-3. 选择并导入 SDK 工程

4. 该工程显示在 Project Explorer 中并已准备好进行调试。

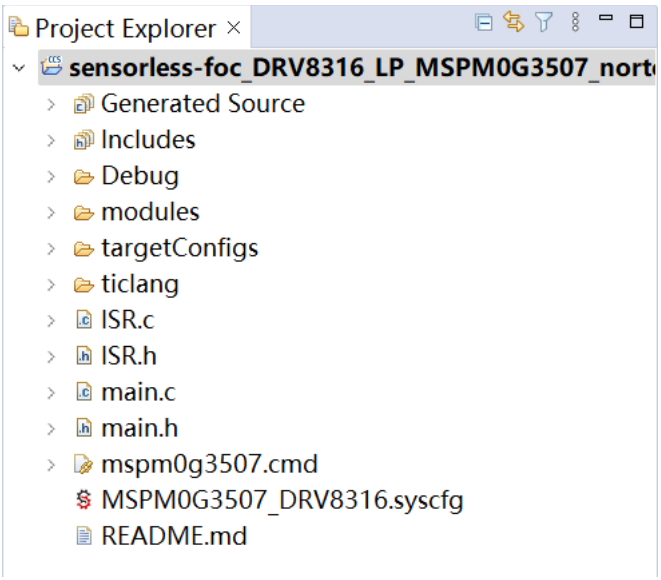


图 6-4. 示例工程概述

6.1.2 工程调试

按照以下步骤调试 FOC 工程。

1. 连接硬件并打开电源。电源电流不应高于 50mA。
2. 点击 Build 按钮： 。工程应在构建时没有错误。
3. 点击 Debug 按钮： 。
4. 打开 Expressions 窗口并添加以下全局结构指针： ***pUserCtrlRegs*** / ***pUserInputRegs*** / ***pUserStatusRegs*** 。

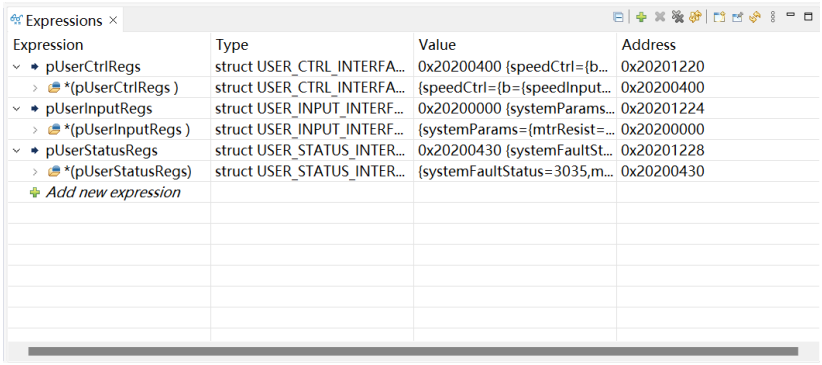


图 6-5. Expressions 窗口

5. 在 Expressions 窗口中启用“连续刷新”： 。
6. 按播放按钮 ：启动 FOC 应用。

6.2 GUI

有关 GUI 快速入门指南，请参阅下方表 6-2 中的 FOC 调优指南。

表 6-2. FOC 调优指南

FOC 算法	FOC 调优指南(1)
无传感器 FOC	无传感器 FOC SDK 用户指南
通用 FOC	通用 FOC SDK 用户指南

表 6-2. FOC 调优指南（续）

FOC 算法	FOC 调优指南(1)
带传感器 FOC	带传感器 FOC SDK 用户指南

(1) 包括 GUI 设置、基本和高级功能调优等。

7 电机调优指南

本节提供了全面的参考工作流程，旨在指导新手完成 MSPM0 FOC 工程的实施过程。本分步指南可确保用户能够高效执行初始设置、电机旋转过程和后续调优过程，从而实现出色的电机控制性能。

7.1 硬件板参数

硬件板参数包含电流和电压的 ADC 检测范围。FOC 算法使用标幺值对测量和计算进行标准化处理。因此，用户应根据 ADC 采样硬件电路设置正确的板参数。

下表展示了硬件板参数的变量结构。

表 7-1. 硬件板参数

变量	说明
pUserInputRegs-> systemParams.voltageBase	电路板的基极电压计算为以下公式中详述的最大可测量电压（以伏特为单位）：MAX_DC_VOLTAGE / SQRT(3)
pUserInputRegs-> systemParams.currentBase	电路板的基极电流根据 CSA 增益（以安培为单位）计算得出

备注

板参数不正确会导致功率、电压和电流控制环路不准确，并降低 FOC 控制效率。

7.1.1 基极电压 (V)

最大电压表示电机控制系统中可测量的最大总线电压和相位电压。有关电压分压器比例调节率的硬件配置，请参阅图 7-1。

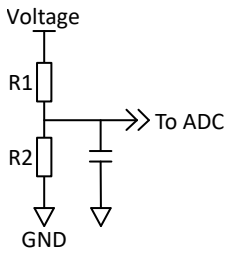


图 7-1. 用于电压检测的 ADC 电压分压器

用户可以根据电压调节电阻分压器电桥值 R1 和 R2 以及 3.3V 的满量程 ADC 电压 (FSV) 来计算系统 MaxVoltage，如方程式 27 所示：

$$\text{Max Voltage} = \text{FSV} / \text{Voltage Divider Scaling Ratio} = 3.3\text{V} \times (\text{R1}+\text{R2}) / \text{R2} \quad (27)$$

基极电压表示 FOC 控制系统中的最大电压，它是标幺值系统中的基极电压值：

$$\text{Base Voltage} = \text{Max Voltage} / \text{SQRT}(3) \quad (28)$$

例如，在从直流电源电压到 ADC 输入的电阻分压器比例调节率为 1/20 的系统中，ADC 可测量的最大系统电压为 3.3V / (1/20) = 66V，基极电压为 38.1V。

备注

使用相同的硬件电压分压器进行总线电压和电机相电压检测。

7.1.2 基极电流 (A)

基极电流表示电机控制系统中可测量的最大电机相电流。

用户可以根据以伏/安为单位的电流检测放大器增益 (CSAGAIN) 和满量程 ADC 电压 (FSV) 来计算系统基极电流，如方程式 29 中所示。在以 1.65V 作为零电流偏移的情况下，考虑使用因数 2 来支持双向电流检测，如图 7-2 所示。

$$\text{Base Current} = (\text{FSV} / 2) / \text{CSAGAIN [V/A]} \tag{29}$$

例如，在 CSAGAIN = 0.15V/A 的系统中，基极电流或 ADC 的最大可测量系统电流为 3.3V / (2 × 0.15V/A) = 11A。

如果系统使用电流检测电阻器 (R_{SENSE}) 和以伏/伏 (V/V) 为单位的 CSAGAIN，则可以使用方程式 30 计算 CSA 增益（以伏/安为单位）。

$$\text{CSAGAIN [V/A]} = \text{RSENSE} \times \text{CSAGAIN [V/V]} \tag{30}$$

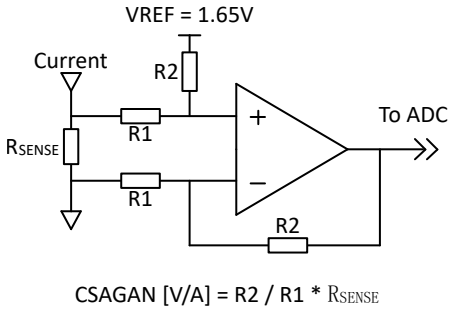


图 7-2. 双向电流检测电路

如果系统使用单向电流检测电路（通常用于单分流器电流检测方法），则会消除零电流偏移，如图 7-3 所示。引入了 OffsetVoltage，表示放大器的共模电压，用于根据方程式 31 获得电流测量的最大分辨率。

$$\text{Base Current} = (\text{FSV - OffsetVoltage}) / \text{CSAGAIN [V/A]} \tag{31}$$

例如，在 OffsetVoltage = V_{CSAREF}/8 和 CSAGAIN = 0.4V/A 的 DRV8329 系统中，基极电流或 ADC 的最大可测量系统电流为 (3.3V – 3.3V/8) / (0.4V/A) = 7.22A。

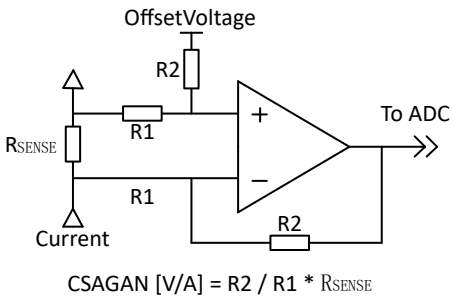


图 7-3. 单向电流检测电路

7.2 电机参数

电机参数包含电阻、电感、极点对、BEMF 和最大速度。下表展示了电机参数的变量结构。

表 7-2. 电机参数

变量	说明
pUserInputRegs ->systemParams.mtrResist	以毫欧为单位的电机相电阻

表 7-2. 电机参数 （续）

变量	说明
pUserInputRegs ->systemParams.mtrInductance	以微亨为单位的电机电感。对于凸极电机： (Lq + Ld)/2
pUserInputRegs ->systemParams.mtrSaliency	以浮点数表示电机凸极 (Lq-Ld)/(Lq+Ld)
pUserInputRegs ->systemParams.mtrBEMFConst	电机 BEMF 常数，单位为 mV/Hz × 10
pUserInputRegs ->systemParams.maxMotorSpeed	电机数据表中以 Hz 为单位的额定电机转速
pUserInputRegs ->systemParams.maxMotorPower	电机数据表中以 Hz 为单位的额定电机转速
pUserInputRegs ->systemParams.polePairs	仅适用于**带传感器 FOC**，表示电机的极点对总数或极点数/2

7.2.1 电机相电阻 (mΩ)

用户可以从电机数据表中获取电机相电阻。如果电机没有数据表，则使用数字万用表测量任意两相的相间电阻，并通过将相间电阻除以 2 来计算相电阻，如方程式 32 所示。

$$\text{Phase resistance} = \text{Measured Phase-to-Phase Resistance} \times 0.5 \tag{32}$$

此测量对星形绕组和三角形绕组电机均适用。在工程中，我们将其等同于星形绕线电机，并应用其与电机相关的参数来参与控制算法。

7.2.2 电机相电感 (μH)

用户可以根据电机数据表计算电机相电感和凸极，如方程式 33 和方程式 34 所示。

$$\text{Motor Inductance} = (Lq + Ld) / 2 \tag{33}$$

$$\text{Motor Saliency} = (Lq - Ld) / (Lq + Ld) \tag{34}$$

如果电机没有数据表，要了解电机电感，可使用 LCR 表在 1kHz 频率下测量任意两相的相间电感。通过将相间电感除以 2 来计算相电感，如方程式 35 所示。凸极可以设置为 0。

$$\text{Phase Inductance} = \text{Measured Phase-to-Phase Inductance} \times 0.5 \tag{35}$$

7.2.3 IPMSM 电机的凸极

IPMSM 电机的凸极是对正交轴和直接转子轴之间电感变化的度量。对于 FOC 算法，根据方程式 34 以浮点变量形式给出此值。

推断 Ld 和 Lq 值的简单方法是测量任意两相的电感，并在一次完整的旋转中缓慢改变转子位置。最大测量电感值可标记为 Lq，最小测量电感值可标记为 Ld。

7.2.4 电机极点对

用户可以从电机数据表中获取电机极点对。极点对参数广泛用于计算电气参数，包括 BEMF 和最大电机速度。对于带传感器 FOC 解决方案，极点对参数是执行霍尔传感器校准的关键特性。

如果电机没有数据表，请执行以下步骤：

1. 使用实验室电源，并确保电流限制设置为低于电机额定电流。请勿打开电源。
2. 将电源的 V+ 连接到电机的 A 相，将电源的 V- 连接到 B 相。如果所有相位都没有标记，可以随机选择 3 相中的任意 2 相。
3. 打开电源，转子应在注入电流后稳定在一个位置。
4. 手动旋转转子，直到转子对齐到另一个稳定位置。转子在一个机械周期周围的不同位置趋于稳定。
5. 对一次完整机械循环的稳定位置数进行计数，该数字即为极对数。乘以 2 后便可计算出极数。

注意电机内的传动系统。传动比将确定多少转子转数与轴的机械旋转相关联。

7.2.5 电机 BEMF 常数 (mV/Hz)

使用电机的数据表，用户能够以 mV/Hz 为单位输入电机的 BEMF 常数 ***Ke*** ，并将 mtrBEMFConst 编程为 ***Ke*** × 10。例如，如果电机的 BEMF 常数为 40mV/Hz，则用户应设置 mtrBEMFConst = 400。

有时，电机数据表使用不同的单元表示电机的 BEMF 常数。可使用方程式 36 和方程式 37 将其转换为 mV/Hz。

$$\text{BEMF Constant [mV/Hz]} = \text{BEMF Constant [mV/rpm]} \times 60 / \text{Pole Pairs} \tag{36}$$

$$\text{BEMF Constant [mV/Hz]} = \text{BEMF Constant [mN·m/A]} \times 2\pi / \text{Pole Pairs} \tag{37}$$

如果电机没有数据表，则通过手动旋转电机，使用示波器测量电机任意两相间的电压。示波器上应出现正弦或梯形电压。测量峰值电压 ***Ep*** （以毫伏为单位）和时间段 ***Tp*** （以秒为单位）。下图展示了示波器捕获的测试电机波形。然后，用户可以计算 BEMF 常数 ***Ke*** ，如方程式 38 所示。

$$\text{BEMF Constant Ke [mV/Hz]} = \text{Ep} \times \text{Tp} / \text{SQRT(3)} \tag{38}$$

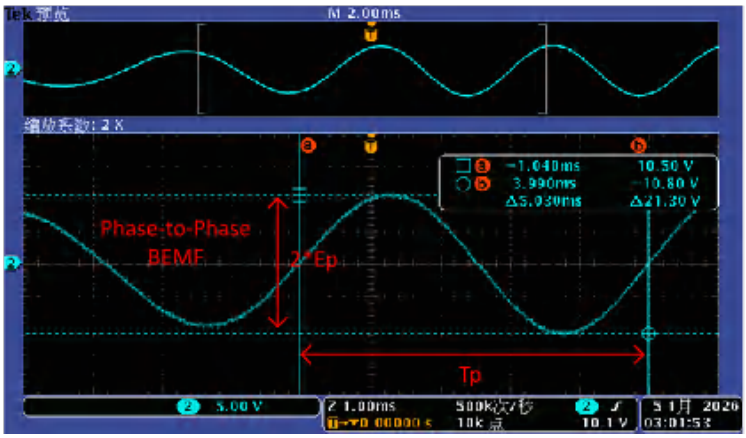


图 7-4. 电机相间 BEMF

7.2.6 最大电机频率 (Hz)

使用电机数据表，用户能以 Hz 为单位输入最大电机电气速度。用户可以使用方程式 39 将以 rpm 为单位的电机机械速度转换为以 Hz 为单位的电机电气速度。

$$\text{Max Motor Speed [Hz]} = \text{Pole Pairs} \times \text{Mechanical Speed [rpm]} / 60 \tag{39}$$

7.2.7 最大电机功率 (W)

当需要闭环功率控制时，用户需要输入电机的最大额定功率。要确定电机的最大额定功率，请参阅电机数据表，计算电机额定电压（以伏特为单位）和电机额定电流（以安培为单位）的乘积，并将该值馈入系统参数中的 MOTOR_MAX_POWER。

7.3 控制环路参数

FOC 算法实现了比例积分 (PI) 控制稳压器来进行闭环控制，包括电流环路和速度环路。如果用户启用了电源环路，电源环路将取代速度环路。

PI 稳压器包含两个参数 Kp 和 Ki，供用户进行调优。可使用默认 PI 参数旋转电机并在运行时进行调优，以提高控制性能。

7.3.1 速度/功率环路

FOC 算法使用集成式速度/功率控制环路，该环路有助于在不同的运行条件下保持恒定的速度/功率，如表 7-3 所示。

表 7-3. 速度/功率环路 PI 参数

变量	说明
pUserInputRegs ->systemParams.speedLoopKp	闭环速度/功率控制的比例增益（浮点型）
pUserInputRegs ->systemParams.speedLoopKi	闭环速度/功率控制的积分增益（浮点型）

速度环路/功率环路的输出用于生成扭矩（电流环路）控制的电流基准。当速度环路/功率环路的输出饱和时，积分器被禁用以防止积分饱和。速度环路 Kp 和 Ki 的调优是试验性的。表 7-4 给出了更改 PI 参数的一般准则。

表 7-4. PI 增益系数增大时的稳定状态和瞬态响应趋势

参数	上升时间	过冲	趋稳时间	稳态错误	稳定性
Kp	减小	提高	微小变化	减小	降级
Ki	减小	提高	提高	消除	降级

7.3.2 电流环路

FOC 算法有两个 PI 控制器，分别用于 Id 和 Iq，以实现磁通和扭矩的独立控制。Kp 和 Ki 系数对于两个 PI 控制器而言是相同的，可在根据表 7-5 进行配置。电流控制环路的输出用于生成要施加到电机的电压信号 Vd 和 Vq。电流环路的输出被钳制到电源电压 VM。首先执行 Id 电流 PI 环路，然后检查 Id 电流 PI 环路 Vd 的输出是否饱和。当电流环路的输出饱和时，积分被禁用以防止积分饱和。

表 7-5. 电流环路 PI 参数

变量	说明
pUserInputRegs ->systemParams.currLoopKp	用户为闭环扭矩控制设置的比例增益（浮点型）
pUserInputRegs ->systemParams.currLoopKi	用户为闭环扭矩控制设置的积分增益（浮点型）

FOC 应用支持根据电机参数（电阻和电感）自动计算电流环路 PI 参数。默认情况下，在算法中，电流环路带宽设置为 FOC 环路频率的 0.03 倍。例如，如果用户将 FOC 环路设置为 10kHz，则默认电流环路带宽为 300Hz。

将表 7-5 中所述的变量设置为 0 即可启用 PI 参数自动计算。然后，用户可以根据表 7-6 获得计算出的 PI 参数。

表 7-6. 自动计算出的电流环路 PI 参数

变量	说明
pUserStatusRegs ->currentPI.Kp	只读。电流环路的比例/积分增益。 该值从以下变量传递： ***pUserInputRegs*** ->systemParams.currLoopKp / currLoopKp（如果其中一个具有非零值）。 如果 ***pUserInputRegs*** ->systemParams. currLoopKp / currLoopKp 均为零，则该值由 FOC 算法生成。
pUserStatusRegs ->currentPI.Ki	

获得表 7-6 中所示的自动计算出的 PI 参数后，用户可以将计算出的参数值设置为 PI 参数的初始值，如表 7-5 所示。然后，用户可以根据实际控制性能进行手动参数调优。

备注

由于电机转速和电流基于 PU 值，因此用户在 ***pUserInputRegs*** 中设置的电流环路 PI 参数与 PI 控制器计算中使用的实际 PI 参数之间存在比例因子关系。

7.4 霍尔角度表

在**带传感器 FOC** 中，使用霍尔信号来检测转子位置信息并高效驱动电机。FOC 应用需要三个数字霍尔输入，这些输入以电气方式相隔 60 度，并连接到 GPIO，作为馈送转子位置信息的输入。

对于给定的霍尔序列，用户需要参考电机相位连接以 IQ27 (PU) 格式适当地填充霍尔角度表，如下表所示。

表 7-7. 霍尔角度表

文件	常量	说明
ISR.c	***forwardHallIndexLUT[MAX_HALL_INDEX]***	用于正向旋转的霍尔角度表。
	reverseHallIndexLUT[MAX_HALL_INDEX]	用于反向旋转的霍尔角度表。

通常，电机数据表中会提供与电机相位连接相关的霍尔引脚序列。霍尔引脚序列很重要，给定相位的驱动角取决于连接顺序。通常，霍尔传感器位置可能存在误差，电气位移可能小于或大于 60 度。霍尔位置的任何误差都会导致产生扭矩纹波和非正弦电流。用户需要进行霍尔传感器校准，以校正霍尔位置误差，从而改善电机中的电流波形。

7.4.1 霍尔校准

表 7-7 中所示的霍尔角度表值可通过**霍尔校准例程**自动生成，如下所述：

1. Motor_Align：校准序列的第一步是将电机对齐到已知的转子角度。在此例程中，电机在持续时间 (pUserInputRegs->mtrStartUp1.b.calibAlignTime) 内与角度对齐（CALIBRATION_ALIGN_ANGLE 宏）。需要配置足够的 calibAlignTime，以便电机在启动校准之前对齐并停止移动。
2. Motor_Calib_Run：Motor_Align 成功完成后，转子以宏 (CALIBRATION_ANGLE_STEP) 中指定的角度进行微步进，持续时间为 (pUserInputRegs->mtrStartUp1.b.calibRunTime)。电机在正向和反向均旋转一个完整的机械周期，以便计算平均霍尔角度。
3. Motor_Calib_Complete：成功完成 Motor_Calib_Run 后，霍尔角度表中会生成正向和反向上每次霍尔状态切换的转子角度，分别为 g_pMC_App->hallAngleTableForward 和 g_pMC_App->hallAngleTableReverse。

下表展示了用于霍尔校准的相关变量。

表 7-8. 霍尔校准变量

变量/宏	说明
pUserInputRegs ->mtrStartUp1.b.calibCurrLimit	霍尔校准期间的电流限制，表示为基极电流的百分比。
pUserInputRegs ->mtrStartUp1.b.calibAlignTime	转子初始对齐到 CALIBRATION_ALIGN_ANGLE 所花的时间。
pUserInputRegs ->mtrStartUp1.b.calibRunTime	针对每个 CALIBRATION_ANGLE_STEP 校准霍尔所花的时间。
pUserInputRegs ->mtrStartUp1.b.currRampRate	Motor_Align 期间直至达到最大电流的初始电流斜变速率。
pUserCtrlRegs ->algoDebugCtrl2.b.hallCaliEnable	用于启用自动霍尔校准。
pUserCtrlRegs ->speedCtrl.b.speedInput	目标电机速度/扭矩值，表示为速度或扭矩命令值的百分比 × 32768。
g_pMC_App->hallAngleTableForward	用于正向旋转的霍尔角度表。如果不执行霍尔校准，则从 ***forwardHallIndexLUT*** 传递该值。
g_pMC_App->hallAngleTableReverse	用于反向旋转的霍尔角度表。如果不执行霍尔校准，则从 ***reverseHallIndexLUT*** 传递该值。
CALIBRATION_ALIGN_ANGLE	***hallCalib.h*** 文件中定义的初始转子对齐。
CALIBRATION_ANGLE_STEP	***hallCalib.h*** 文件中定义的校准角度步长。

7.4.2 寄存器表

本节介绍用于霍尔校准的寄存器表。

MOTOR_STARTUP1 寄存器（带传感器 FOC）

表 7-9 展示了用于配置电机启动设置 1 的寄存器。

表 7-9. MOTOR_STARTUP1 寄存器字段说明

位	字段	类型	复位	说明
31-30	RESERVED	R	00b	保留

表 7-9. MOTOR_STARTUP1 寄存器字段说明（续）

位	字段	类型	复位	说明
29-26	CURR_RAMP_RATE	R/W	0h	霍尔校准期间的初始电流斜升速率，直至达到最大电流。 0h = 0.1A/s 1h = 1A/s 2h = 5A/s 3h = 10A/s 4h = 15A/s 5h = 25A/s 6h = 50A/s 7h = 100A/s 8h = 150A/s 9h = 200A/s Ah = 250A/s Bh = 500A/s Ch = 1000A/s Dh = 2000A/s Eh = 5000A/s Fh = 无限值 A/s
25-22	CALIB_RUN_TIME	R/W	0h	针对每个 CALIBRATION_ANGLE_STEP（在 hallCalib.h 中定义）校准霍尔时花费的时间 0h = 10ms 1h = 50ms 2h = 100ms 3h = 200ms 4h = 300ms 5h = 400ms 6h = 500ms 7h = 750ms 8h = 1s 9h = 1.5s Ah = 2s Bh = 3s Ch = 4s Dh = 5s Eh = 7.5s Fh = 10s

表 7-9. MOTOR_STARTUP1 寄存器字段说明（续）

位	字段	类型	复位	说明
21-17	CALIB_CURRENT_ILIMIT	R/W	00h	霍尔校准期间的电流限制，以 CURRENT_BASE 的百分比表示 0h = 7.5% 1h = 8.0% 2h = 8.5% 3h = 9.0% 4h = 9.5% 5h = 10% 6h = 11% 7h = 12% 8h = 13% 9h = 14% Ah = 15% Bh = 16% Ch = 17% Dh = 18% Eh = 20% Fh = 22.5% 10h = 25% 11h = 27.5% 12h = 30% 13h = 35% 14h = 40% 15h = 45% 16h = 50% 17h = 55% 18h = 60% 19h = 70% 1Ah = 75% 1Bh = 80% 1Ch = 85% 1Dh = 90% 1Eh = 95% 1Fh = 100%
16-13	CALIB_ALIGN_TIME	R	000b	开始霍尔校准之前，转子初始对齐到零度所花的时间。 0h = 10ms 1h = 50ms 2h = 100ms 3h = 200ms 4h = 300ms 5h = 400ms 6h = 500ms 7h = 750ms 8h = 1s 9h = 1.5s Ah = 2s Bh = 3s Dh = 5s Eh = 7.5s Fh = 10s
12-2	RESERVED	R	0h	保留
1	OL_ILIMIT_CONFIG	R/W	0b	开环电流限值配置 0h = 由 OL_ILIMIT 定义的开环电流限值 1h = 由 ILIMIT 定义的开环电流限值

表 7-9. MOTOR_STARTUP1 寄存器字段说明（续）

位	字段	类型	复位	说明
0	RESERVED	R	0b	保留

7.5 旋转电机 (LVBLDC)

本节提供了有关如何通过 DRV8316REVM 旋转 LVBLDC 电机的示例。用户可以在产品页面中找到相关参数，也可参阅下表。

表 7-10. LVBLDC 参数

器件型号	额定				线对线		扭矩常数	BEMF	惯性	极点对
	电压	速度	扭矩	电源	电阻	电感				
DN42040S24-026	VDC	RPM	mNm	W	Ω	mH	mNm/A	V/Krpm	g·cm2	/
	24	4000	62.5	26	1.50	2.10	35.00	2.45	24	4

请参阅节 6.1 运行 **sensorless-foc_DRV8316** 工程。在调试模式下，用户可以在 Expression 窗口中动态覆盖 FOC 控制寄存器。图 7-5 展示了为匹配电机参数而覆盖的变量结果。

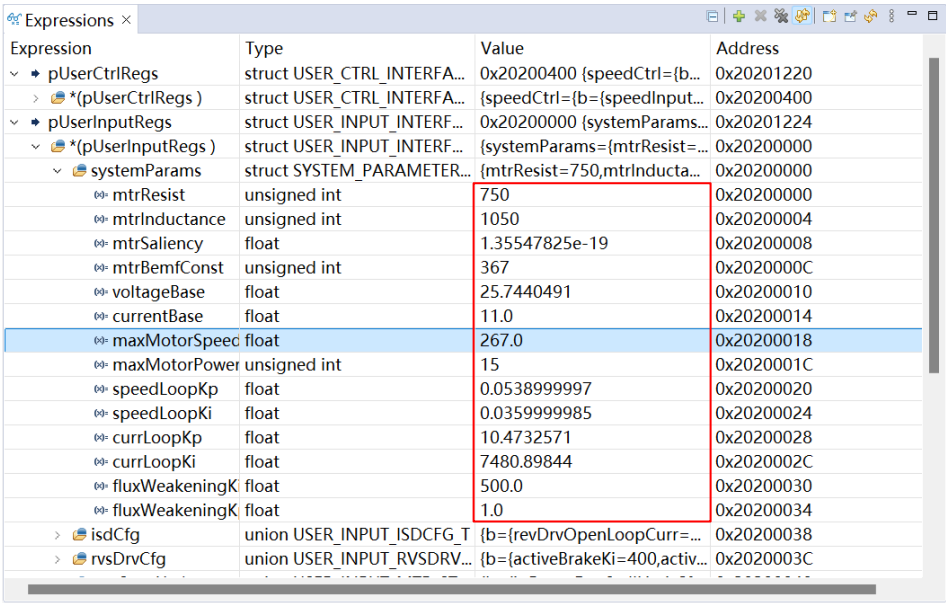


图 7-5. 在 Expression 窗口中动态覆盖 FOC 控制寄存器

电机参数也可以直接硬编码到源代码中，以避免重复修改相同参数的繁琐过程，如图 7-6 所示。

```
105   focPeriphInit();  /* Does foc application specific Peripheral configurations */
106
107   /* Configure the Motor Params */
108   pUserInputRegs->systemParams.mtrResist        = 750;
109   pUserInputRegs->systemParams.mtrInductance    = 1050;
110   pUserInputRegs->systemParams.mtrBemfConst     = 367;
111   pUserInputRegs->systemParams.maxMotorSpeed = 266.7;
112
113   /* Configure the Current Loop Params - copy from auto calculated */
114   pUserInputRegs->systemParams.currLoopKp     = 10.4732571;
115   pUserInputRegs->systemParams.currLoopKi     = 7480.89844;
116
117   while (1)
118   {
```

图 7-6. 将电机参数硬编码到源代码中

然后，用户可以设置 speedInput 的值以启动 FOC 控制并旋转电机，如图 7-7 所示。

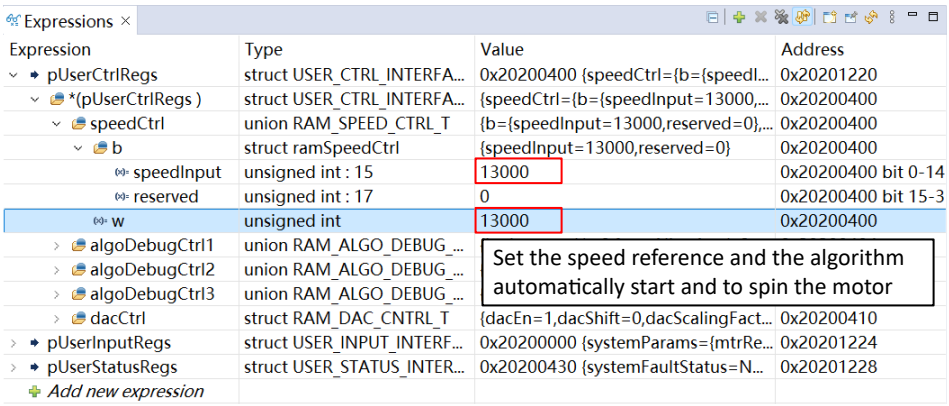


图 7-7. 设置 speedInput 的值以旋转电机

用户可以更改 ***pUserCtrlRegs*** ->speedCtrl.w 或 ***pUserCtrlRegs*** ->speedCtrl.b.speedInput 的值来实时更改速度。可以在 ***pUserStatusReg s*** 中观测电机实时状态，如图 7-8 所示。例如， ***pUserStatusRegs*** ->piSpeed.feedback 或 ***pUserStatusRegs*** ->estimatedSpeed 表示电机实时速度。

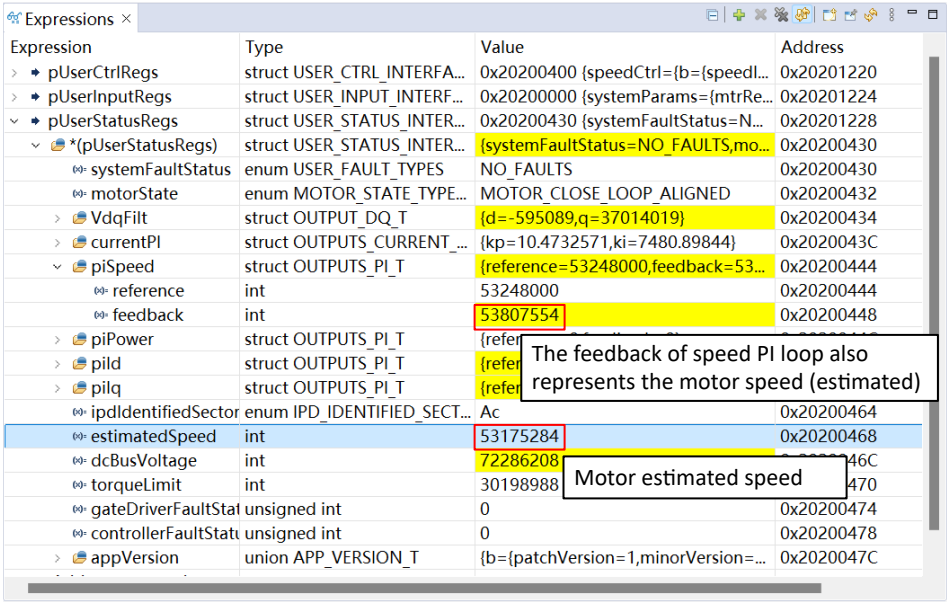


图 7-8. 在 Expressions 窗口中监测电机状态

FOC 算法使用定点数据来提高代码效率。MSPM0 driverlib 提供了 IQmath 库，让用户可以轻松集成定点数据来模拟浮点数据。在 CCS Expressions 窗口中，有一种显示 IQ 类型值的简单方法，如图 7-9 所示。

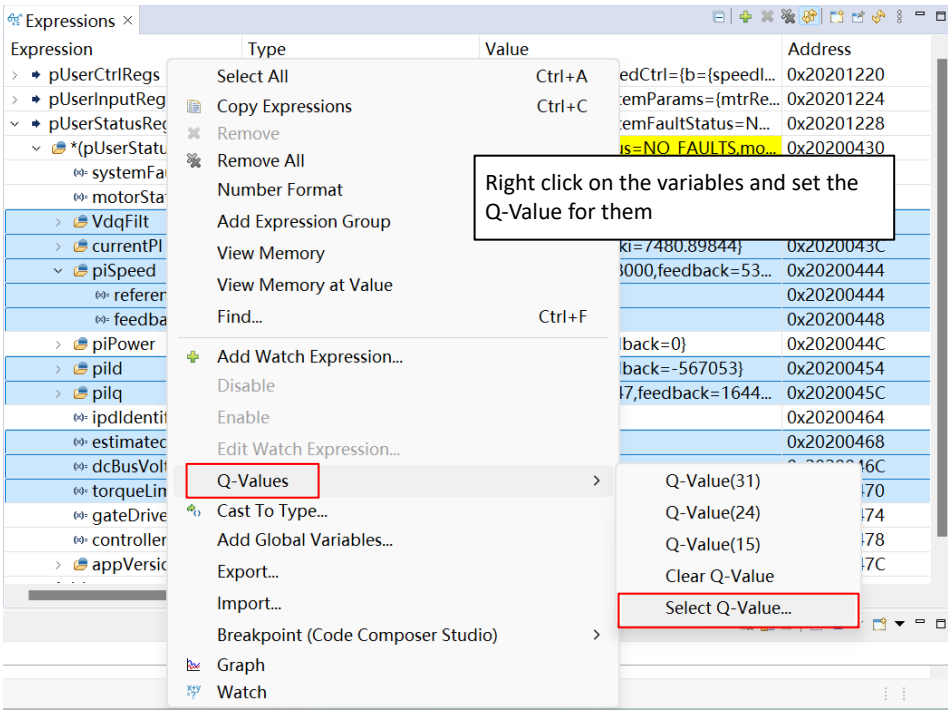


图 7-9. 将变量设置为 IQ 类型

电机速度输入变量设置为 IQ15 类型。如果适用于 PU 类型，寄存器中的其他变量将设置为 IQ27 类型。图 7-10 展示了以 IQ27 类型显示的变量。

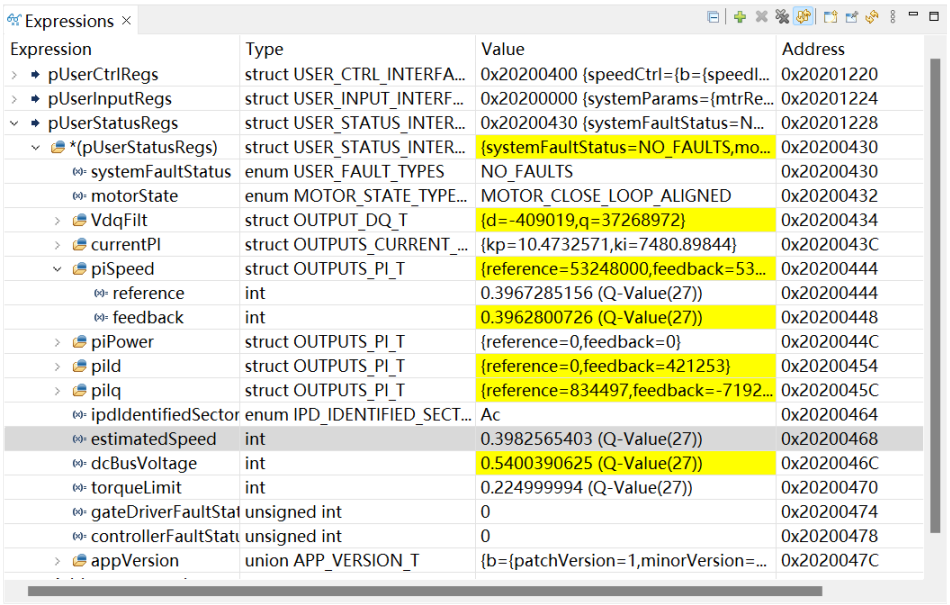


图 7-10. 以 IQ27 类型显示变量

7.6 旋转带霍尔传感器的电机

本节使用带霍尔传感器的标准 BLDC 电机 (DT4260) 来演示带传感器 FOC 解决方案。下表给出了该电机的参数。

表 7-11. DT4260 电机参数

器件型号	额定			线对线		扭矩常数	BEMF	惯性	极点对
	电压	速度	扭矩	电阻	电感				
DT4260-24-055-04H	VDC	rpm	mNm	Ω	mH	mNm/A	mV/Hz	g·cm2	/
	24	4000	125	0.8	1.2	35.50	35.7	48	4

按照以下步骤使用带传感器 FOC 解决方案旋转电机：

1. 有关 MSPM0 Launchpad 和 DRV8316AEVM 之间的硬件连接，请参阅带传感器 FOC SDK 用户指南。
2. 请参阅节 6.1.1 导入 CSS 工程：

   hall_sensored-foc_DRV8316_LP_MSPM0G3507_nortos_ticlang 或

   hall_sensored-foc_DRV8316_LP_MSPM0G3519_nortos_ticlang
3. 根据电机数据表（节 7.2）覆盖电机参数，或者进行手动测量（表 7-11）。
4. 根据节 7.4.1 设置霍尔校准参数。
5. 设置算法调试控制 2 寄存器中的霍尔传感器校准使能位。
6. 添加下列表达式：
   - g_pMC_App->foc.hallCalibObj.calibState
   - g_pMC_App->hallAngleTableForward
   - g_pMC_App->hallAngleTableReverse
7. 将 **speedInput** 设置为非零值即可启动霍尔校准。
8. 霍尔校准完成后，将 **speedInput** 设置为零即可复位 FOC 状态机。
   - 用户应使用新校准的霍尔角度值更新 ISR.c 中硬编码的霍尔角度表（请参阅 表 7-7）。
9. 将 **speedInput** 设置为非零值即可旋转电机。

备注

需要进行霍尔校准，才能使用霍尔接口旋转新的 BLDC 电机。

图 7-11 展示了用于设置电机参数 (DT4260) 和霍尔校准参数的硬编码。

```
105     focPeriphInit();  /* Does foc application specific Peripheral configurations */
106
107     /* Configure the Motor Params */
108     pUserInputRegs->systemParams.mtrResist        = 400;    //mOhm
109     pUserInputRegs->systemParams.mtrInductance    = 600;    //uH
110     pUserInputRegs->systemParams.mtrBemfConst     = 357;    //0.1 mV/Hz
111     pUserInputRegs->systemParams.polePairs     = 4;         // Sensored foc only
112     pUserInputRegs->systemParams.maxMotorSpeed = 266.7;
113
114     /* When set as 0, the algorithm will automatically calculate current loop parameters */
115     pUserInputRegs->systemParams.currLoopKp     = 0;
116     pUserInputRegs->systemParams.currLoopKi     = 0;
117     /* Configure the Current Loop Params - copy from auto calculated */
118 //    pUserInputRegs->systemParams.currLoopKp     = 5.9847188;
119 //    pUserInputRegs->systemParams.currLoopKi     = 3989.8125;
120
121     /* COnfigure HAL Table Calibration */
122     pUserInputRegs->mtrStartUp1.b.calibCurrLimit = 0x0; // 7.5%*base_current, Id
123     pUserInputRegs->mtrStartUp1.b.currRampRate = 0x1;   // 1 A/s8
124     pUserInputRegs->mtrStartUp1.b.calibAlignTime = 0x6; // 500ms for 0 degree
125     pUserInputRegs->mtrStartUp1.b.calibRunTime = 0x2;   // 100ms for each step
126     /*
127      * Note:
128      * Set macro definition CALIBRATION_ALIGN_ANGLE in hallCalib.h for initial position alignment
129      * Set macro definition CALIBRATION_ANGLE_STEP in hallCalib.h to use different step
130      */
131     while (1)
132     {
133         if(gdReadTestEn)
134         {
135             regData =  gateDriverRegRead(regAddr);
136         }
137         UART_checkForCommand(pUART);
138
139         updateConfigs();
140     }
141 }
```

图 7-11. 霍尔参数配置的硬编码

图 7-12 和图 7-13 展示了 CCS *Expressions* 窗口中霍尔校准的详细执行流程。

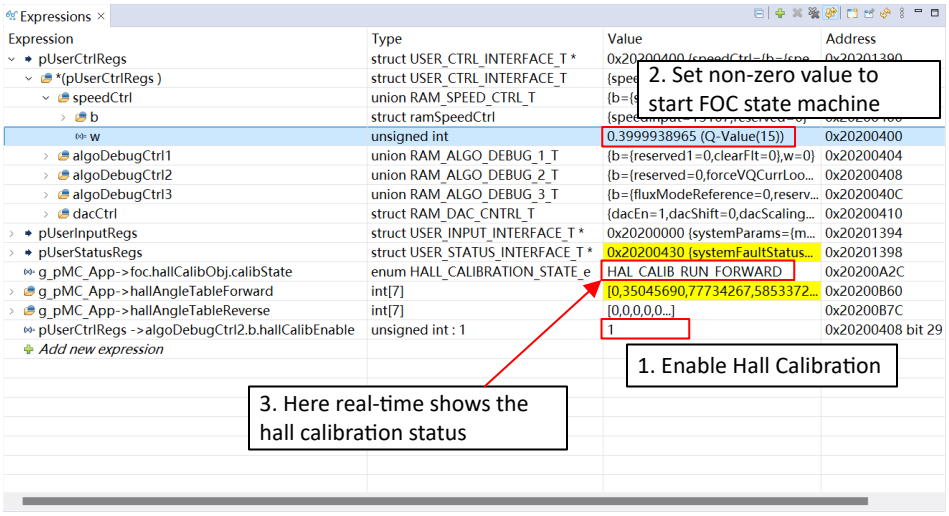


图 7-12. 霍尔校准运行

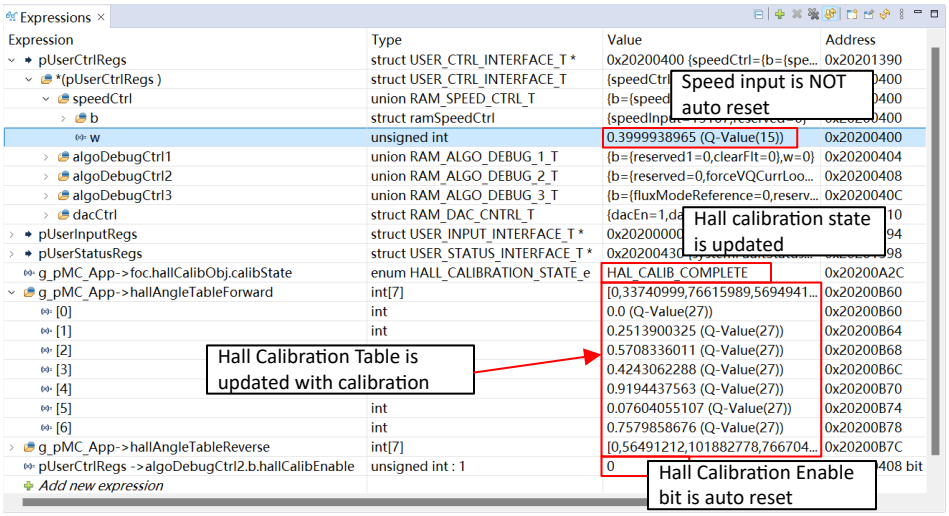


图 7-13. 霍尔校准完成

用户需要手动将 **speedInput** 设置为零来复位 FOC 状态机。然后，用户将 IQ15 格式的目标速度设置到 **speedInput** 寄存器中，以旋转电机，如图 7-14 所示。

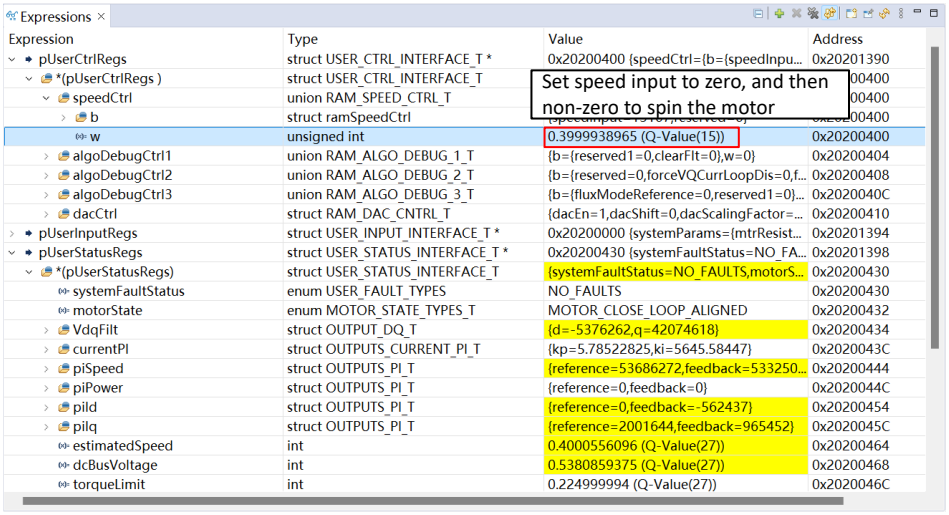


图 7-14. 霍尔校准完成后旋转电机

霍尔校准完成后，用户应覆盖 ISR.c 文件中的霍尔角度表，如图 7-14 所示。对霍尔信号接线的任何改动都需要重新执行完整的校准流程。

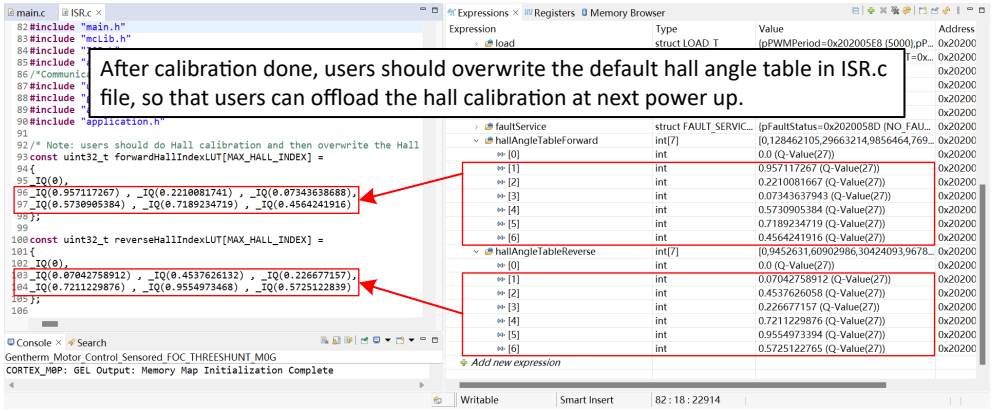


图 7-15. 覆盖霍尔角度表

霍尔校准需要对寄存器进行额外的修改，完成该校准需要一定的时间。有关自动实现霍尔校准的硬编码，请参阅以下代码：

```
    /* Start Calibration when first connected to the motor */
    __BKPT(0);    /* For debug */

    pUserCtrlRegs->algoDebugCtrl2.b.hallCalibEnable = 0x1;

    pUserCtrlRegs->algoDebugCtrl2.b.updateConfigs = 0x1;
    while(pUserCtrlRegs->algoDebugCtrl2.b.updateConfigs){
        updateConfigs();    /* Polling until all register updated */
    }

    /* Start Calibration */
    pUserCtrlRegs->speedCtrl.b.speedInput = 10000;
    while (g_pMC_App->foc.hallCalibObj.calibState != HAL_CALIB_COMPLETE) {
        updateConfigs();
        /* Polling until calibration done */
    }
    /* Reset motor control state machine */
    pUserCtrlRegs->speedCtrl.b.speedInput = 0;
    pUserCtrlRegs->algoDebugCtrl2.b.hallCalibEnable = 0x0;

    pUserCtrlRegs->algoDebugCtrl2.b.updateConfigs = 0x1;
    while(pUserCtrlRegs->algoDebugCtrl2.b.updateConfigs){
        updateConfigs();
    }

    __BKPT(0);    /* For debug */
    /* Calibration done */
```

7.7 调优电机 (LVBLDC)

下图展示了在默认调优参数下旋转电机时的单相电流波形。用户可以单独设置每种模式的配置，以满足特定的应用要求。

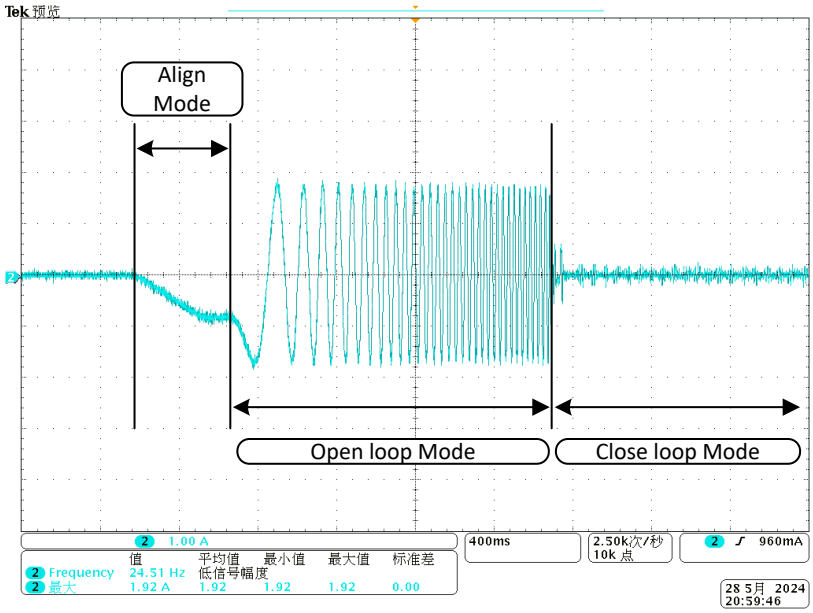


图 7-16. 电机启动相电流波形

备注

本节介绍了无传感器 FOC 的调优指南。同样的过程也适用于通用 FOC 和带霍尔传感器 FOC（如果支持这些特性）。

7.7.1 基本调优

固件具有默认设置以便于启动。根据应用要求，用户可以覆盖寄存器来调优电机。本节介绍基本的调优功能。

7.7.1.1 启动模式

FOC 算法支持不同的启动模式，包括对齐模式、双对齐模式、IPD 模式和 SFC 模式。固件将对齐模式设置为默认启动模式。用户可以将 ***pUserInputRegs*** ->mtrStartUp1.b.b.mtrStartUpOption 设置为不同的值来选择其他启动模式。

启动模式不适用于带传感器 FOC 算法，因为带传感器 FOC 算法直接从霍尔传感器获取电机转子初始位置（分辨率为 60 度）。

7.7.1.1.1 对齐模式

对齐方法是无传感器 FOC 控制算法的基本启动方法。它旨在获得电机转子的精确初始位置。将 mtrStartUpOption 设置为 0h 即可选择对齐模式。

按照以下步骤调优对齐模式参数：

1. 使用 pUserInputRegs->mtrStartUp2.b.alignAngle 设置电机转子角度对齐
2. 使用 pUserInputRegs->mtrStartUp1.b.alignTime 设置对齐模式持续时间
3. 使用 pUserInputRegs->mtrStartUp1.b.b.alignOrSlowCurrLimit 设置最大对齐电流
4. 使用 pUserInputRegs->mtrStartUp1.b.b.alignSlowRampRate 设置达到最大对齐电流前的不同电流斜升速率

7.7.1.1.1.1 电流环路中的强制对齐模式

为了进行调试，FOC 算法提供了一个选项，可强制电机控制状态机保持对齐模式，以调整对齐模式参数并高效验证性能。

将 ***pUserCtrlRegs*** ->algoDebugCrtl1.b.forceAlignEn 设置为 1b 即可启用强制对齐模式。将 speedInput 设置为非零值即可让电机旋转。下图展示了强制对齐模式下的电流波形。

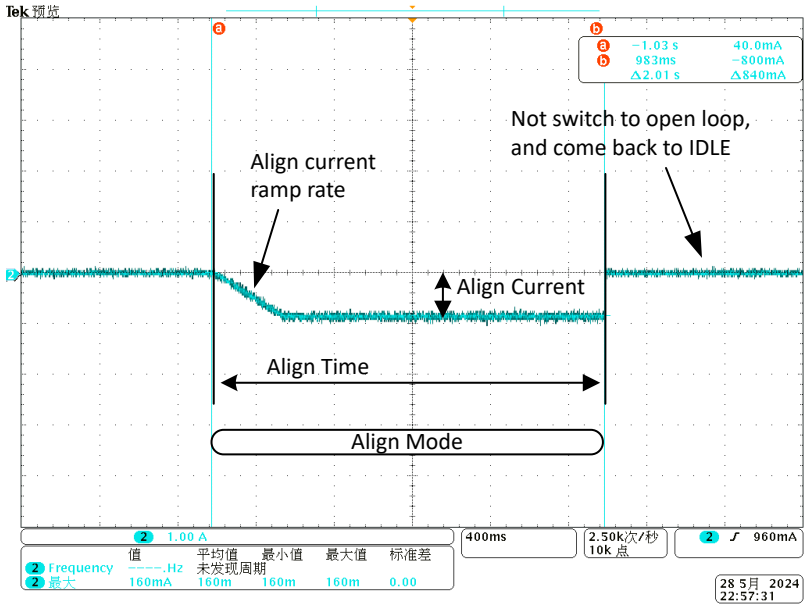


图 7-17. 电流环路中的强制对齐模式

在对齐模式下，默认应用 q 轴电流对齐方法。执行电流闭环，三相电流将包括一个正电流和两个负电流。

7.7.1.1.1.2 PWM 环路中的强制对齐模式

为了进行调试，FOC 算法提供了一个选项，可强制电机控制状态机保持在对齐模式并禁用电流闭环。

当禁用电流闭环时，FOC 算法直接将 PWM 占空比控制为三相。由于此时未启用控制环路，用户可以轻松检查 PWM 功能并验证硬件电路连接。

可设置 ***pUserCtrlRegs*** ->algoDebugCrtl2.b.currLoopDis 来禁用电流闭环。然后，用户可以覆盖 ***pUserCtrlRegs*** ->algoDebugCrtl2.b.forceVDCurrLoopDis 和 forceVQCurrLoopDis，以设置不同的 SVPWM 输出，如图 7-18 所示。

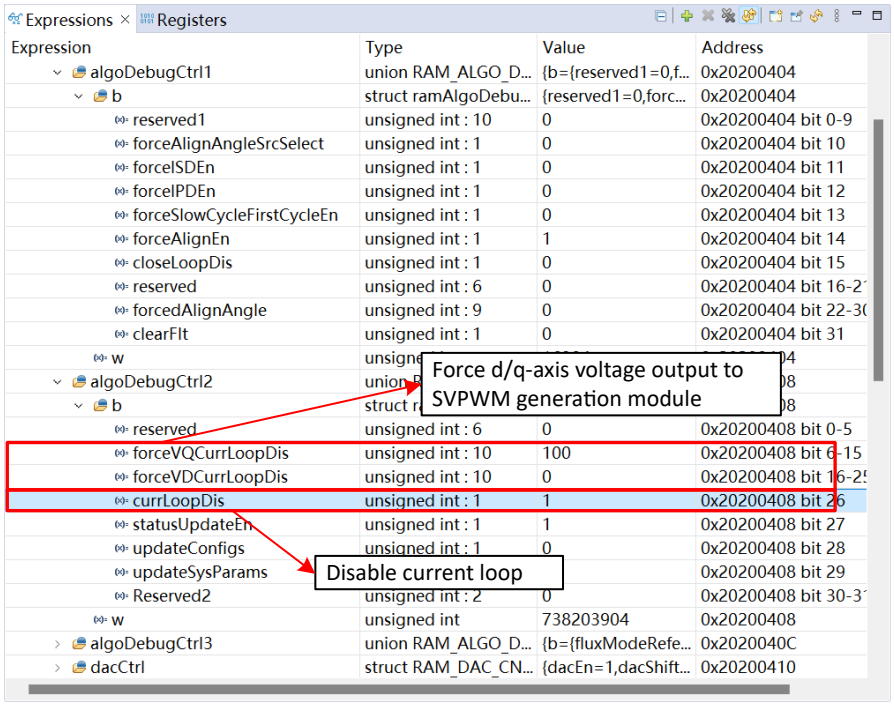


图 7-18. CCS 调试窗口中的寄存器设置

图 7-19 展示了强制对齐模式下的 PWM 输出波形。

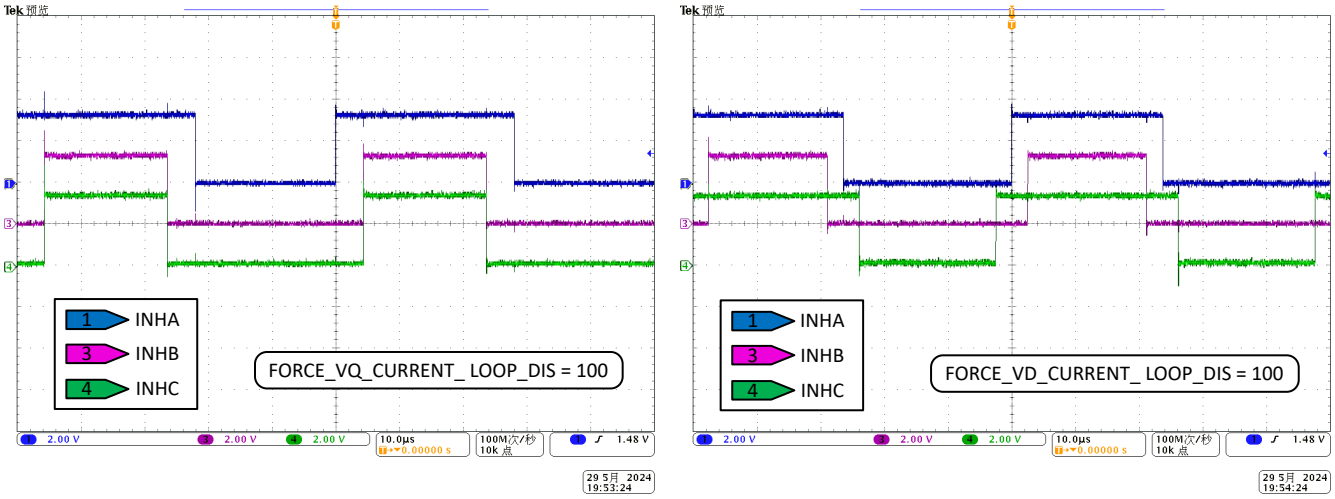


图 7-19. PWM 环路中的强制对齐模式

备注

TI 建议在测试 PWM 环路时断开电机，因为小占空比可能会在电机相位输出高电流。如果连接了电机，用户应小心地从零开始增加 PWM 周期。

7.7.1.1.2 双对齐模式

双对齐方法可避免电机对齐过程中因转子位置与对齐角度相差 180° 电角度而出现的临界稳定状态。

FOC 应用不支持双对齐模式，将 mtrStartUpOption 设置为 1h 也表示对齐模式。

7.7.1.1.3 初始位置检测 (IPD) 模式

对齐或双对齐方法可能会使电机在开始开环加速之前反向旋转。IPD 可用于不允许电机反转的应用。IPD 不会等待电机与换向对齐，因此可以实现更快的电机启动序列。当电机的电感作为位置的函数变化时，IPD 效果很好。IPD 在工作时会将脉冲电流输入电机，因此会产生噪声，在确定特定应用的最佳启动方法时必须考虑这一点。

IPD 算法利用 ADC 电流检测路径来识别相电流上升时间，从而检测转子位置。ADC 的窗口比较器根据预设的 IPD 阈值电流限制持续监测相电流，从而生成电流脉冲。使用计时器捕获不同扇区的这些不同脉冲的上升时间，并进行比较以检测转子位置。

计时器和 ADC 配置会在 IPD 初始化期间根据 SysConfig 进行更新。算法会根据所选的电流检测方法（节 4.2.3）和电流检测类型（节 7.1.2）来配置所需的 WCOMP 设置。

备注

在 IPD 脉冲时间内，其余的算法中断将暂停，以便以非常高的采样率持续监测 ADC 电流。IPD 运行完成后，将恢复正常的中断运行。

将 mtrStartUpOption 设置为 2h，即可选择 IPD 模式进行启动。按照以下顺序配置 IPD 参数：

1. 选择 IPD 电流阈值 [pUserInputRegs->mtrStartUp1.b.ipdCurrThresh]。根据电机的电感饱和点来选择 IPD 电流阈值。电流越高，准确检测初始位置的机率就越高。但是，较高的电流会导致转子运动、振动和噪声。建议先设置为电机额定电流的 50%。如果电机启动失败，则增加阈值，直到电机成功启动。不要将 IPD 电流阈值设置为高于电机的额定电流。
2. 选择 IPD 时钟值 [pUserInputRegs->mtrStartUp1.b.ipdClkFreq]。IPD 时钟定义施加 IPD 脉冲的速度。对于具有更高时间常数和更高电流阈值的电机，需要更长的时间来衰减电流，然后才会激励下一个 IPD 脉冲。因此，将时钟设置为较慢的时间作为起始值，然后逐渐增加，直到不触发 IPD TIME OUT 故障。较慢的时钟会使 IPD 噪声更柔和，但持续时间更长，因此应设置时钟，使 IPD 故障不被触发且噪声在合理可接受范围内。

   备注

   如果电机时间常数非常高或电机未连接，该器件会触发 IPD 超时故障 IPD_FAULT_CLOCK_TIMEOUT (controllerRawFaultStatus)。如果触发了此故障，请确保电机连接到器件。如果故障仍然存在，则减小 IPD 时钟频率 (ipdClkFreq)。
3. 选择 IPD 超前角 [pUserInputRegs->mtrStartUp1.b.ipdAdvAngle]。从 90° 开始，以实现最大启动扭矩。如果在启动过程中观察到急冲，请将该角度减小到 60° 或 30°，以实现更平稳的启动。
4. 选择执行 IPD 的次数 [pUserInputRegs->mtrStartUp1.b.ipdRepeat]。增加的 IPD 执行次数可提高 IPD 精度，但也会增加时间消耗。用户可以从 1 次开始。

将 ***pUserCtrlRegs*** ->algoDebugCrtl1.b.forceIPDEn 设置为 1b 以启用强制 IPD 模式。这对于用户调试 IPD 性能和调优参数很有用。

7.7.1.1.3.1 高分辨率 IPD

FOC 应用通过将 ***pUserInputRegs*** ->miscAlgo.b.ipdHiResolEn 设置为 1b 来支持高分辨率 IPD 功能（仅在无传感器 FOC 中可用）。下表展示了 IPD 与高分辨率 IPD 之间的差异。

表 7-12. ISD 功能比较

算法	说明
IPD	基本 ISD 功能支持以 30 电角度以内的精度检测初始电机转子角度
高分辨率 IPD	高分辨率 IPD 功能支持以 10 电角度以内的精度检测初始电机转子角度

7.7.1.1.4 慢速首循环 (SFC) 模式

将 mtrStartUpOption 设置为 3h 即可选择慢速首循环模式。在慢速首循环启动中，FOC 算法在 ***pUserInputRegs*** ->mtrStartUp2.b.slowFirstCycFreq 定义的频率下启动电机换向。配置的频率仅用于第一个周期，然后电机换向遵

循由开环参数所配置的加速曲线，如下图所示。必须将慢速首循环配置得足够慢，以便允许电机与换向序列同步。当需要快速启动时，该模式很有用，因为它可以显著减少对齐时间。

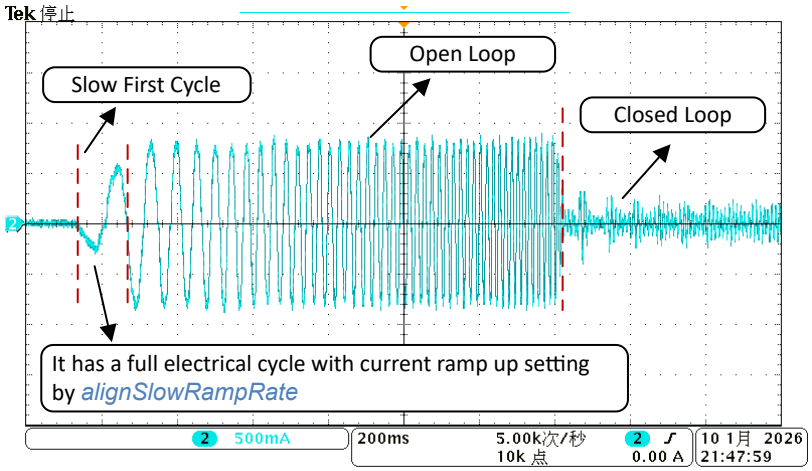


图 7-20. 慢速首循环模式下的相电流

将 ***pUserCtrlRegs*** ->algoDebugCrtl1.b.forceSlowCycleFirstCycleEn 设置为 1b，即可启用强制慢速首循环模式来进行调试。

7.7.1.2 开环模式

使用对齐、双对齐、IPD 或慢速首循环完成电机位置初始化后，FOC 算法开始在开环中加速电机。在开环期间，速度在固定的电流限制下增加。在开环中，Iq 和 Id 的控制 PI 环路主动控制电流。开环期间的角度由斜坡发生器提供。

按照以下步骤调优开环参数：

1. 设置 pUserInputRegs->mtrStartUp2.b.olILimit，以进行开环电流限制设置。

   备注

   将电流限制配置为高于电机失速电流的值时，可能会导致电机过热或损坏。

2. 使用 pUserInputRegs->mtrStartUp2.b.olAcc1 和 pUserInputRegs->mtrStartUp2.b.olAcc2 设置电机的加速度和 Jerk 参数，以确定电机加速时间（等于开环时间），请参阅方程式 40。
3. 使用 pUserInputRegs->mtrStartUp2.b.olClHandOffThr 设置从开环模式到闭环模式的开关速度。如果电机无法达到该开关速度，请提高电流限制 (olILimit)。

$$\text{Speed}(t) = \text{olAcc1} \times t + \text{olAcc2} \times t^2 \quad (40)$$

用户必须为反电动势观测器设置足够的开关速度，以估算电机的角度和速度。对于大多数电机，15~25% 的额定速度应能正常工作并获得稳定的开关性能。

7.7.1.2.1 自动转换

如果电机有足够的 BEMF 供观测器使用，用户可以将 ***pUserInputRegs*** ->mtrStartUp2.b.autoHandOffEn 设置为 1b 来启用自动切换。

可使用 ***pUserInputRegs*** ->miscAlgo.b.autoHandoffMinBemf 来配置自动切换的 BEMF 阈值 (mV)。

7.7.1.2.2 强制开环模式

为了进行调试，FOC 算法提供了一个选项，可强制电机控制状态机保持开环模式，以调整开环模式参数并高效验证性能。在强制开环模式下，用户可以通过调整电流环路控制参数（currLoopKp 和 currLoopKi）来直接观测电机相电流响应。

将 ***pUserCtrlRegs*** ->algoDebugCrtl1.b.closeLoopDis 设置为 1b 即可启用强制开环模式。下图展示了强制开环模式下的电机相电流波形。

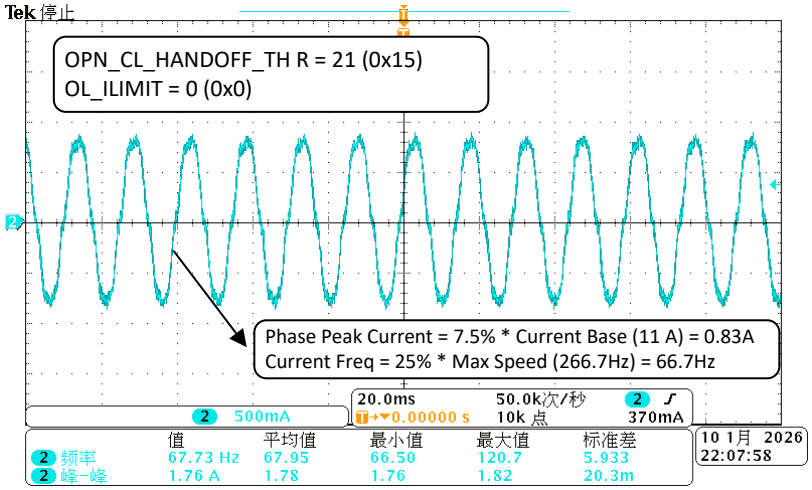


图 7-21. 强制开环模式下的相电流

7.7.1.3 从开环转换到闭环

在电机达到足够的速度，使反电动势观测器能够估算电机角度和速度时，FOC 算法就会切换到闭环模式。

为了实现平稳切换并避免速度瞬变，在切换之后 theta_error（开环角度 - 估算角度）呈线性减小。可以使用 ***pUserInputRegs*** ->mtrStartUp2.b.thetaErrRampRate 来配置 theta_error 减小的斜率。

如果在开环期间设置的电流限制 (olILimit) 较高，并且在切换到闭环之前未降低该电流限制，则电机速度可能会在切换到闭环后瞬间升至高于速度基准 (speedInput) 的值。为了避免这种速度变化，可将 ***pUserInputRegs*** ->mtrStartUp2.b.iqRampEn 配置为 1b，以便 iq_ref 在切换至闭环之前减小，如图 7-22 所示。然而，如果速度基准 (speedInput) 是开闭环切换速度 (olClHandOffThr) 的两倍以上，则无论 iqRampEn 设置如何，iq_ref 都不会减小，以实现更快的电机加速。

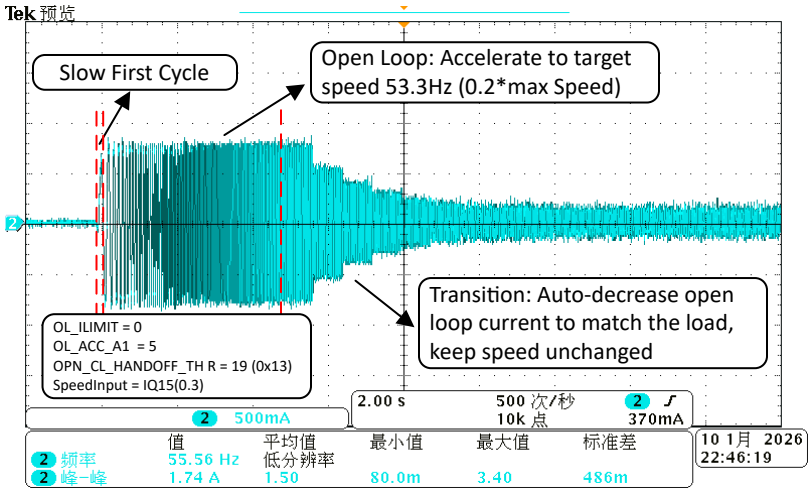
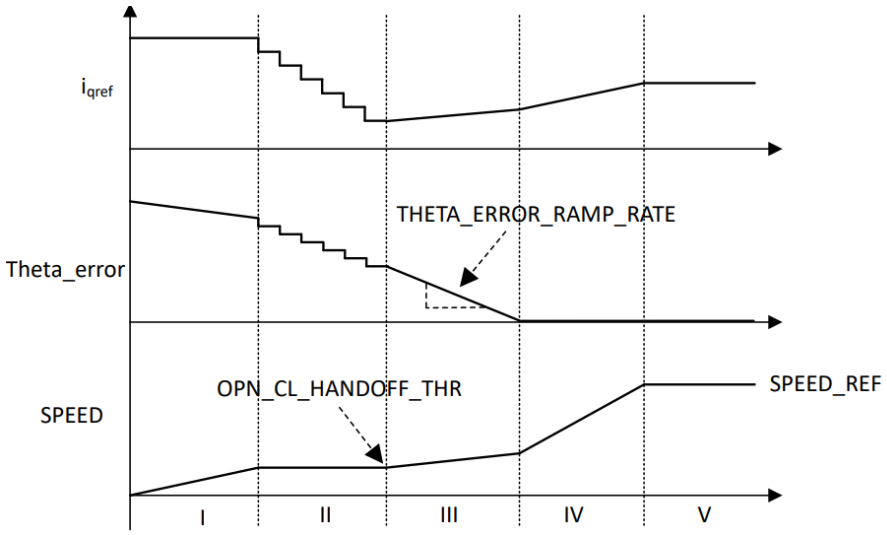


图 7-22. 启用 IqRamp 时的相电流

在以足够的速度切换至闭环后，可能仍然存在一些 θ 误差，因为估算器可能没有完全对齐。在开环转换到闭环之后可以使用缓慢加速，来确保 θ 误差减小为零。可以使用 ***pUserInputRegs*** ->miscAlgo.b.clSlowAcc 来配置缓慢加速。

图 7-23 展示了开闭环缓慢切换中的控制序列。



I. Open Loop Acceleration, II. Current Decay, III. Closed loop slow acceleration
IV. Closed loop acceleration, V. Closed loop steady state

图 7-23. 开闭环转换中的控制序列

7.7.1.4 闭环模式

在闭环运行模式下，电机角度和电气速度通过反电动势观测器进行估算。速度和电流调节是使用 PI 控制环路实现的。为了提高效率，直轴电流被设置为零 (Id_ref = 0)，这将确保定子和转子磁场相互正交（相位差为 90 度）。

要让电机在闭环模式下稳定旋转，用户应保持足够的速度，让反电动势观测器能估算电机的角度和速度。对于闭环运行，建议保持高于额定电机速度的 15%。

7.7.1.4.1 调优控制参数

集成式速度（或功率）控制环路有助于在不同的运行条件下保持恒定的速度（或功率）。Kp 和 Ki 系数通过 speedLoopKp 和 speedLoopKi 进行配置。速度环路的输出受到限制以实现电流限制。可以通过配置 iLimit 来设置电流限制。

按照以下步骤在闭环中对电机进行调优：

1. 使用 pUserInputRegs->closeLoop1.b.iLimit 来设置闭环最大电流限制。与开环不同，闭环中的 q 轴电流取决于电机负载。如果电机速度小于基准速度，则增大电流限制。
2. 使用 pUserInputRegs->closeLoop1.b.clAcc 和 pUserInputRegs->closeLoop1.b.clDec 设置电机的加速和减速转换率参数。即使在速度基准输入 (speedInput) 发生阶跃变化时，这种方式也能使速度基准输入实现线性变化，有助于防止施加到电机上的扭矩发生突变，从而降低噪声。

7.7.1.4.2 调优 PI 参数

要调优速度环路的 Kp 和 Ki 值，请执行以下操作：

1. 通过将 closeLoopDis 设置为 1b，将电机配置为在开环中持续旋转。
2. 如果通过用户应用代码启用了自动切换，则通过将 autoHandOffEn 设置为 0b 来将其禁用。
3. 使用 olClHandOffThr 将闭环切换阈值设置为最大速度的约 50%。
4. 将 iqRampEn 位设置为 1b，以便让 iq_ref 在切换到闭环之前减小
5. 电流基准逐渐降低，并稳定至可能的最低 Iqref，以便以给定的阈值速度运行。
6. 使用方程式 41 计算速度环路 Kp (speedLoopKp)。

$$\text{Speed Loop Kp} = \text{Current Reference at olClHandOffThr (Amps)} / \text{olClHandOffThr (Hz)} \quad (41)$$

7. 使用方程式 42 计算速度环路 Ki (speedLoopKi)。

$$\text{Speed Loop Ki} = \text{Speed Loop Kp} \times 0.1 \quad (42)$$

8. 通过将 closedloopDis 清除为 0b 来启用闭环。

备注

速度环路 Kp 和 Ki 的调优是试验性的。如果上述建议不起作用，则建议手动调优速度环路 Kp 和 Ki，直到实现所需的结果。

7.7.1.5 停止模式

将 speedInput 设置为 0，可以生成停止命令。FOC 算法提供了用于停止电机的不同选项，可通过 ***pUserInputRegs*** ->closeLoop1.b.mtrStopOption 配置这些选项。

7.7.1.5.1 滑行（高阻态）模式

可以通过将 mtrStopOption 设置为 0h 来配置滑行（高阻态）模式。

当收到电机停止命令时，FOC 应用将通过关断所有 MOSFET 而切换到高阻抗 (Hi-Z) 状态。当 FOC 应用从驱动电机切换到高阻态时，电机绕组中的电感电流会继续流动，能量通过 MOSFET 输出级中的体二极管返回到电源中（请参阅示例）。

下图展示了高组态模式工作原理的示例。在该示例中，电流通过高侧 A 相 MOSFET (HSA)、高侧 B 相 MOSFET (HSB) 施加到电机，并通过低侧 C 相 MOSFET (LSC) 返回。

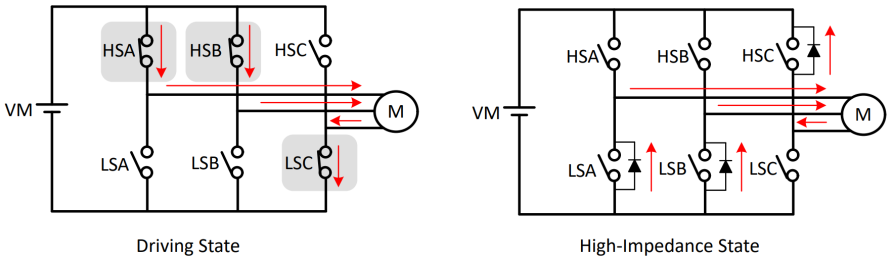


图 7-24. 滑行（高阻态）模式

在滑行模式下，反向电流会注入直流总线，并可能导致直流总线电压过冲。

7.7.1.5.2 主动降速模式

可以通过将 mtrStopOption 设置为 1h 来配置主动降速模式。当收到电机停止命令时，FOC 应用将电机速度基准降至 ***pUserInputRegs*** ->closeLoop2.b.actSpinThr，然后通过关断所有 MOSFET 来切换到高阻态。

该模式的优点是通过降低电机速度基准，电机减速至较低的速度，从而在进入高阻态之前减小相电流。现在，当电机切换到高阻态时，传输到电源的能量将减少。

在不需要快速停止，但接受一定量的电感能量回输到电源的应用中，可以使用主动降速作为电机停止选项。如果电源发生电压过冲，请减小 ACT_SPIN_THR，直到电压过冲达到可接受的限制。请注意，阈值 actSpinThr 需要配置得足够高，以使 FOC 应用不会在闭环中与电机失去同步。

7.7.1.5.3 制动模式

可以通过将 mtrStopOption 设置为 2h 来配置制动模式。按照以下说明设置制动模式：

1. 通过将 pUserInputRegs->pinCfg.b.brakeInput 设置为 2h，应用软件配置来启用制动模式。FOC 应用在此设置下使用停止命令进入制动模式。
2. 将 brakeInput 设置为 1h，以直接进入制动状态而无需停止命令。FOC 应用保持在制动状态，直到 brakeInput 设置为 2h。如果 brakeInput 恢复至 1h 时 speedInput 不为零值，电机将立即开始旋转。

3. 通过将 brakeInput 设置为 0h 或 3h，配置硬件输入引脚 (BRAKE) 来启用制动模式。只要 BRAKE 引脚驱动为高电平，电机就会保持在制动状态。

进入制动状态前，FOC 应用会将输出速度降至 ***pUserInputRegs*** ->closeLoop2.b.brkSpeedThr 定义的值。如果在收到停止命令前电机速度低于 brkSpeedThr，FOC 应用将直接切换至制动状态。

可以通过 ***pUserInputRegs*** ->pinCfg.b.brakePinMode 将制动状态进一步配置为低侧制动 (Low-Side Braking) 或对齐制动 (Align Braking)。

7.7.1.5.3.1 低侧制动

将 brakePinMode 设置为 0b 以选择低侧制动模式。下图展示了低侧制动状态的电路状态。

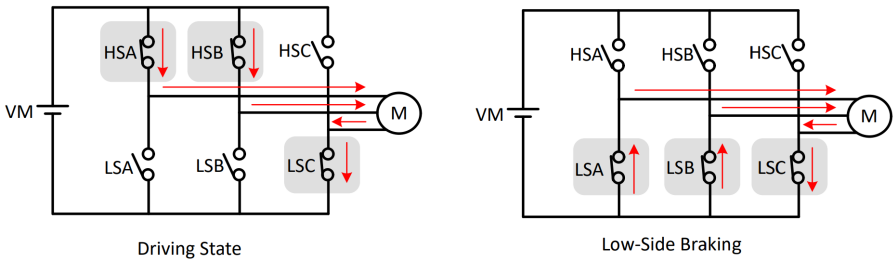


图 7-25. 低侧制动

7.7.1.5.3.2 对齐制动

可以通过将 brakePinMode 设置为 1b 来配置对齐制动模式。FOC 应用还可以通过 BRAKE 引脚进入对齐制动状态。在该模式下，FOC 应用在由 ***pUserInputRegs*** ->miscAlgo.b.brkCurrPersist 配置的特定时间内通过特定的相位模式注入直流电流来对齐电机。

对齐期间的相位模式是根据需要执行对齐的角度生成的，该角度可通过 ***pUserInputRegs*** ->mtrStartUp2.b.alignAngle 进行配置。对齐制动期间的电流限制阈值可以通过 ***pUserInputRegs*** ->mtrStartUp1.b.alignOrSlowCurrLimit 来配置。下图展示了对齐制动模式下的相电流波形。

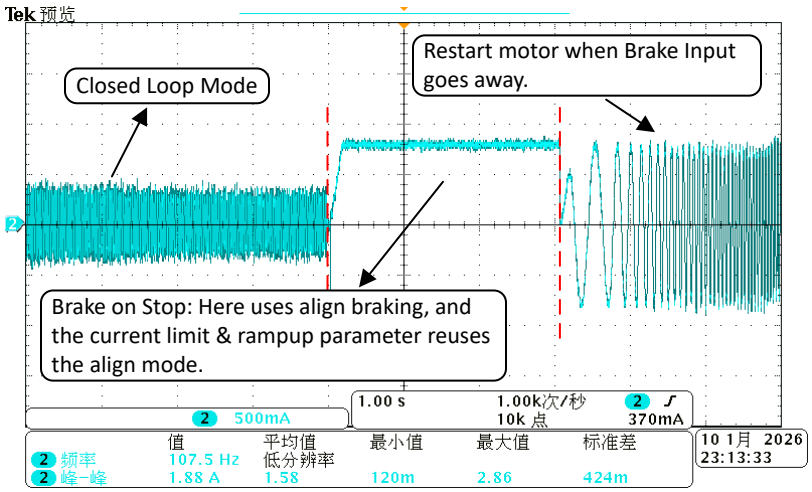


图 7-26. 对齐制动

7.7.1.6 故障处理

故障状态记录在 ***pUserStatusRegs*** 中（请参阅节 5.3.3）。下表汇总了可根据故障配置触发的故障。

表 7-13. 故障报告汇总

系统故障状态	FOC 故障状态	说明
NO_FAULTS	NO_FAULTS (0x00000000)	未发生故障。

表 7-13. 故障报告汇总（续）

系统故障状态	FOC 故障状态	说明
MOTOR_STALL	ABN_SPEED_FAULT_INDEX	电机估算速度超过异常速度锁定阈值（占最大速度的百分比）
	ABN_BEMF_FAULT_INDEX	估算的 BEMF 电压降至编程的异常 BEMF 锁定阈值（占预期 BEMF 的百分比）以下
	NO_MOTOR_FAULT_INDEX	电机相电流低于基极电流的无电机锁定阈值百分比。
VOLTAGE_OUT_OF_BOUNDS	OVER_VOLTAGE_FAULT_INDEX	直流总线电压超过编程的最大电压（以基极电压的百分比表示）。
	UNDER_VOLTAGE_FAULT_INDEX	直流总线电压降至编程的最小电压（以基极电压的百分比表示）以下。
LOAD_STALL	IPD_CLOCK_TIMEOUT_FAULT_INDEX	发生 IPD 超时故障。电流斜升至 IPD 阈值电流之前发生时间溢出。
	IPD_DECAY_TIME_FAULT_INDEX	发生 IPD 衰减时间故障。在下一个 IPD 脉冲之前未完成电流的完全衰减。
HARDWARE_OVER_CURRENT	BUS_CURRENT_LIMIT_INDEX	未应用。
HV_DIE	HV_DIE_FAULT_INDEX	故障输入引脚（PWM 故障引脚）为低电平。存在外部故障事件。
不适用	HW_LOCK_ILIMIT_FAULT_INDEX	未应用。

7.7.1.6.1 MOTOR_STALL

MOTOR_STALL 故障包括无电机故障、异常速度故障和异常 BEMF 故障。可使用 ***pUserStatusRegs*** ->controllerFaultStatus 确定在 MOTOR_STALL 故障下触发了哪个故障。

FOC 应用为使用 ***pUserInputRegs*** ->faultCfg1.b.mtrLckMode 和 ***pUserInputRegs*** ->faultCfg1.b.lockRetry 来触发 MOTOR_STALL 提供了各种锁定和重试方法。有关详细说明，请参阅节 5.3.2.7。

7.7.1.6.1.1 ABN_SPEED_FAULT

FOC 应用会持续监测速度，以及速度超过 ***pUserInputRegs*** ->faultCfg2.b.lockAbnSpeed 中定义的异常速度锁定阈值（最大速度的百分比）的情况。

将 ***pUserInputRegs*** ->faultCfg2.b.lock1En 设置为 1b 可启用异常速度保护。

7.7.1.6.1.2 ABN_BEMF_FAULT

将 ***pUserInputRegs*** ->faultCfg2.b.lock2En 设置为 1b 可启用异常 BEMF 保护。当估算的 BEMF 电压降至 ***pUserInputRegs*** ->faultCfg2.b.abnBemfThr 中定义的异常 BEMF 阈值百分比以下时，就会触发此故障。

例如，如果估算或测得的 Ke 为 5mV/Hz，编程的异常 BEMF 阈值为 40%，则当估算的 Ke 降至 2mV/Hz 以下时，就会触发此故障。

有两种触发 ABN_BEMF_FAULT 的典型情况：

1. 注册的 Ke 不准确。可参阅节 7.2.5 来计算或获得准确的 Ke。
2. 估算的 BEMF 电压随着电机转速下降而下降。负载动态变化（负载突变）可导致电机转速下降。对于具有动态负载的应用，转速会下降然后恢复。对于此类应用，建议将异常 BEMF 阈值设置为 10%，从而避免触发此故障。

备注

异常 BEMF 保护还会监测从开环模式到闭环模式的切换阶段结果，如果切换失败，将会报告 ABN_BEMF_FAULT。

7.7.1.6.1.3 NO_MOTOR_FAULT

将 ***pUserInputRegs*** ->faultCfg2.b.lock3En 设置为 1b 可启用无电机保护。当相电流低于 ***pUserInputRegs*** ->faultCfg2.b.noMtrThr 中定义的基极电流的无电机锁定阈值百分比时，就会触发此故障。

如果发生 NO_MOTOR_FAULT，请按照以下步骤操作：

1. 确保电机相位与端子连接牢固。
2. 如果故障仍然存在，可使用 noMtrThr 增大无电机锁定电流阈值。

7.7.1.6.2 VOLTAGE_OUT_OF_BOUNDS

在电源存在波动的应用中，用户需要指定最小和最大电源电压范围。在欠压情况下，电机可能会在过调制区运行，以实现目标转速，从而导致电流失真、效率低下或噪声。在过压情况下，MOSFET 和电机会持续以高电压运行，从而导致负荷过大。

使用 ***pUserInputRegs*** ->faultCfg2.b.minVmMtr 和 minVmMtr 设置适当的直流总线电压范围。使用 ***pUserStatusRegs*** ->controllerFaultStatus 确定是否触发了欠压或过压。

FOC 应用提供了欠压恢复模式 [***pUserInputRegs*** ->faultCfg2.b.minVmMode] 和过压恢复模式 [***pUserInputRegs*** ->faultCfg2.b.maxVmMode]。可以这样配置欠压或过压恢复模式：将 minVmMode 或 minVmMode 设置为 1b 来自动清除欠压，将 minVmMode 或 minVmMode 设置为 0b 来锁存故障。

7.7.1.6.3 LOAD_STALL

当电机启动配置为 IPD（ 节 7.7.1.1.3 ）时，FOC 应用使用在 80MHz 下运行的 16 位计时器来估算 IPD 期间电流斜升和斜降的时间。

在 IPD 期间，该算法会检查电流是否成功斜升至 ipdCurrThresh。如果 IPD 计时器溢出（电流未达到 ipdCurrThresh），将会触发 IPD_CLOCK_TIMEOUT_FAULT。如果发生 IPD_CLOCK_TIMEOUT_FAULT，请使用 ***pUserInputRegs*** ->miscAlgo.b.ipdMaxOverflow 设置更长的超时周期，或使用 ***pUserInputRegs*** ->mtrStartUp1.b.ipdClkFreq 设置更低的 IPD 频率。

类似地，该算法会检查在 IPD 电流斜降期间电流是否成功衰减到零。如果在当前 IPD 脉冲导致电流完全衰减之前命令发送下一个 IPD 脉冲，则 IPD 会给出不正确的结果。如果 IPD 计时器溢出（电流未斜降到零），则会触发 IPD_DECAY_TIME_FAULT。如果发生 IPD_DECAY_TIME_FAULT，请设置更低的 ipdClkFreq。

7.7.1.6.4 HARDWARE_OVER_CURRENT

此功能不适合当前 FOC 算法版本。

7.7.1.6.5 HV_DIE

FOC 应用实现了 PWM 故障引脚保护，让硬件能够快速对外部故障做出反应（禁用 PWM），并将输出信号保持在安全状态。

当 PWM 故障输入引脚为低电平时，MCU 硬件首先在没有软件参与的情况下禁用 PWM 输出（设置为低电平），然后 FOC 应用会触发 HV_DIE 故障并在状态机中处理保护逻辑。

7.7.1.7 电机旋转方向

可使用 ***pUserInputRegs*** ->periphCfg1.b.dirInput 设置电机旋转方向。FOC 应用提供了以下两种方法来设置方向：

1. 通过将 dirInput 设置为 1h（顺时针）或 2h（逆时针）来应用软件配置。
   - 当 dirInput 设置为 1h 时，FOC 应用驱动顺时针旋转： OUTA-OUTB-OUTC
   - 当 dirInput 设置为 2h 时，FOC 应用驱动逆时针旋转： OUTA-OUTC-OUTB
2. 配置硬件输入引脚 (DIR)，通过将 dirInput 设置为 0h 或 3h 来设置电机旋转方向。
   - DIR 输入为高电平时，FOC 应用驱动顺时针旋转：OUTA-OUTB-OUTC
   - DIR 输入为低电平时，FOC 应用驱动逆时针旋转：OUTA-OUTC-OUTB

可使用 ***pUserInputRegs*** ->periphCfg1.b.dirChangeMode 来确定 FOC 应用对电机旋转方向变化的响应：

- 当 dirChangeMode 设置为 0b 时，FOC 应用在电机方向变化时遵循电机停止选项和 ISD 例程。
- 当 dirChangeMode 设置为 1b 时，FOC 应用在持续驱动电机的同时通过反向驱动（节 7.7.2.7.2）改变方向。

7.7.1.8 PWM 配置

7.7.1.8.1 PWM 频率

可使用 ***pUserInputRegs*** ->closeLoop1.b.pwmFreqOut 为电机控制设置不同的 PWM 频率。默认情况下，固件将 PWM 频率设置为 16kHz。

7.7.1.8.2 PWM 死区时间

FOC 应用的硬件电路需要防止 MOSFET 出现任何跨导。通过插入死区时间，可精密控制高侧和低侧 MOSFET，从而避免发生任何击穿事件。

可使用 ***pUserInputRegs*** ->periphCfg1.b.mcuDeadTimeto 设置不同的 PWM 死区时间。

设置更大的死区时间

目前，可以在 ***pUserInputRegs*** ->periphCfg1 寄存器中将死区时间设置为 6 位可配置值。因此，该寄存器最大可提供 3.2uS。如果需要更大的死区时间，用户可以更新此寄存器，将 ***appInputCtrlInterface.h*** 文件中的 MCU_DEAD_TIME 大小增加到的更高位数，从而实现更大的死区时间。

图 7-27 展示了将死区时间配置为 8 位的示例。

```
458 /*! @brief userInputPeriCfg1 structure */
459 typedef struct
460 {
461     uint32_t
462     /*! Response to change of DIR pin status */
463     dirChangeMode:              1,
464     /*! DIR pin override */
465     dirInput:                   2,
466     /*! Bus current limit enable */
467     busCurrLimitEnable:         1,
468     /*! Bus current limit */
469     busCurrLimit:               5,
470     /*! deadtime for PWM outputs */
471     mcuDeadTime:                8,
472     /*! Reserved */
473     reserved:                   15;
474 }userInputPeriCfg1;
```

图 7-27. 将死区时间寄存器设置为 8 位

7.7.1.9 FOC 环路频率

在中断例程中定期执行 FOC 算法，以便更新转子角度，从而使电机达到出色的效率。用户可以根据应用带宽要求来配置此 FOC 速率。

将 ***pUserInputRegs*** ->closeLoop1.b.highFreqFOCEn 设置为 0b，可获得 16kHz 的最大 FOC 执行速率。将 highFreqFOCEn 设置为 1b，可将最大 FOC 执行速率降低一半。

备注

FOC 例程只能在 PWM 频率的倍数值时执行，因此，16kHz 的最大可实现 FOC 速率适用的 PWM 频率为 16 的倍数（例如 16kHz、32kHz、48kHz）。对于 20KHz、40kHz 等 PWM 频率，最大 FOC 速率限制为 10kHz（20KHz/2、40KHz/4 等）。

7.7.1.10 用于基本调优的硬代码

下图展示了调优 LVBLDC 电机的硬代码。

```
    focPeriphInit();  /* Does foc application specific Peripheral configurations */

    /* Configure the Motor Params */
#ifdef  DT42040
    /* DT42040 */
    pUserInputRegs->systemParams.mtrResist        = 750;
    pUserInputRegs->systemParams.mtrInductance    = 1050;
    pUserInputRegs->systemParams.mtrBemfConst     = 367;
    pUserInputRegs->systemParams.maxMotorSpeed = 266.7;
#elif defined (DT4260)
    /* DT4260 */
    pUserInputRegs->systemParams.mtrResist        = 400;
    pUserInputRegs->systemParams.mtrInductance    = 600;
    pUserInputRegs->systemParams.mtrBemfConst     = 357;
    pUserInputRegs->systemParams.maxMotorSpeed = 266.7;
#endif

    /* When set as 0, the algorithm will automatically calculate current loop parameters */
    pUserInputRegs->systemParams.currLoopKp     = 0;
    pUserInputRegs->systemParams.currLoopKi     = 0;
    /* Configure the Current Loop Params - copy from auto calculated */
//    pUserInputRegs->systemParams.currLoopKp     = 5.9847188;
//    pUserInputRegs->systemParams.currLoopKi     = 3989.8125;

    /* StartUp Parameter */
    pUserInputRegs->mtrStartUp1.b.alignOrSlowCurrLimit = 0x0;   //7.5%*base_current
    pUserInputRegs->mtrStartUp1.b.alignTime = 0x6;  /*500ms*/
    pUserInputRegs->mtrStartUp2.b.olILimit = 0x0;       //7.5%*base_current
    pUserInputRegs->mtrStartUp2.b.olAcc1 = 0x5;         // 10Hz/s
    pUserInputRegs->mtrStartUp2.b.olClHandOffThr = 0x15;    //25%*base_speed

    /* Close Loop Parameter */
    pUserInputRegs->closeLoop1.b.controlMode = 0x0; // Closed Loop Speed Control
    pUserInputRegs->closeLoop1.b.iLimit = 0xE;      // 20%*base_current limitation
    pUserInputRegs->closeLoop1.b.clAcc  = 0x5;      // 10Hz/s
    pUserInputRegs->closeLoop1.b.clDec  = 0x5;      // 10Hz/s
//    pUserInputRegs->closeLoop1.b.pwmFreqOut = 0x3;  // Default is 16kHz

    /* Fault Handling */
    pUserInputRegs->faultCfg2.b.lockAbnSpeed    = 0;    // 130% max speed lock
    pUserInputRegs->faultCfg2.b.lock1En         = 0;    // Not enabled
    pUserInputRegs->faultCfg2.b.noMtrThr        = 0;    // 7.5% no motor threshold
    pUserInputRegs->faultCfg2.b.lock3En         = 0;    // Not enabled
    pUserInputRegs->faultCfg2.b.abnBemfThr      = 2;    // 50% Ke threshold
    pUserInputRegs->faultCfg2.b.lock2En         = 0;    // Not enabled

    pUserInputRegs->faultCfg2.b.maxVmMtr        = 0x7;  // No max Vm limitation
    pUserInputRegs->faultCfg2.b.minVmMtr        = 0x0;  // No min Vm limitation

    pUserInputRegs->periphCfg1.b.busCurrLimit   = 0xE;  // 20%*base_current limitation
    pUserInputRegs->periphCfg1.b.busCurrLimitEnable = 0x0;  // Not enabled

    pUserInputRegs->faultCfg1.b.mtrLckMode      = 0;    // Latch up the fault

    while (1)
    {
        if(gdReadTestEn)
        {
            regData =  gateDriverRegRead(regAddr);
        }
        UART_checkForCommand(pUART);

        updateConfigs();
    }

}
```

图 7-28. 用于基本调优的硬代码

7.7.2 高级调优

本节帮助用户调优 FOC 应用的高级特性。

7.7.2.1 控制模式设置

可使用变量 ***pUserInputRegs*** ->closeLoop1.b.controlMode 在以下四种模式下控制 FOC 应用。速度/功率/扭矩/电压的基准输入可通过 speedInput 寄存器来配置。

7.7.2.1.1 闭环速度控制模式

速度闭环控制是一种广泛使用的控制方法。在速度控制模式下，根据速度控制寄存器中设置为 speedInput 值（IQ15 格式的 P.U 值）的输入基准，使用闭环 PI 控制来控制电机速度（以 Hz 表示的电气速度）。P.U 速度计算为 systemParams 中配置的 ACTUAL_MOTOR_SPEED / MOTOR_MAX_SPEED 值。

示例：如果 maxMotorSpeed 设置为 100Hz，则将 speedInput 中的基准输入设置为 0x3FFFh（IQ15 格式为 0.5P.U），可将电机速度设置为 50Hz。

默认情况下，FOC 算法通过将 controlMode 设置为 0h 来使用速度闭环控制。

7.7.2.1.2 闭环功率控制模式

在功率控制模式下，根据速度控制寄存器中设置为 speedInput 值（IQ15 格式的 P.U 值）的输入基准，使用闭环 PI 控制来控制电机的输入电功率（以瓦特为单位）。P.U 功率计算为 systemParams 中配置的 ACTUAL_MOTOR_POWER / MOTOR_MAX_POWER 值。

示例：如果 maxMotorPower 设置为 100W，则将 speedInput 中的基准输入设置为 0x3FFFh（IQ15 格式为 0.5P.U），系统会以 50W 的恒定电功率运行电机。

FOC 算法通过将 controlMode 设置为 1h 来使用功率闭环控制。

7.7.2.1.3 闭环扭矩控制模式

在扭矩（电流）控制模式下，根据速度控制寄存器中设置为 speedInput 值（IQ15 格式的 P.U 值）的输入基准，使用闭环 PI 控制来控制电机的扭矩分量电流 Iq（以安培为单位）。q 轴电流的 P.U 扭矩分量计算为 systemParams 中配置的 TORQUE_CURRENT / CURRENT_BASE 值。

示例：如果 currentBase 设置为 10 安培，则将 speedInput 中的基准输入设置为 0x3FFFh（IQ15 格式为 0.5P.U），系统会以 5A 的恒定 q 轴电流运行电机。

备注

当电机在扭矩模式下运行时，必须将适当的负载连接到电机。

FOC 算法通过将 controlMode 设置为 2h 来使用扭矩闭环控制。

7.7.2.1.4 开环电压控制模式

在电压控制模式下，速度和电机 d-q 轴电流不受控制。FOC 算法直接将 d-q 轴电压输出到 SVPWM 模块。根据在速度控制寄存器中设置为 speedInput 值（IQ15 格式的 P.U 值）的输入基准来控制电机的调制指数。

示例：将 speedInput 中的基准输入设置为 0x3FFFh（IQ15 格式为 0.5P.U），以 0.5 的恒定调制指数运行电机。

FOC 算法通过将 controlMode 设置为 3h 来使用电压开环控制。

7.7.2.1.4.1 超前角控制

在电压控制模式下，可以调节超前角，以在给定速度下获得出色的电机效率。使用 ***pUserInputRegs*** ->closeLoop2.b.leadAngle 设置超前角。

对于给定超前角（θ），施加的电压 Vq 和 Vd 定义为：

$$Vq = MODULATION\_INDEX \times Cos\theta \quad (43)$$

$$Vq = MODULATION\_INDEX \times Sin\theta \quad (44)$$

7.7.2.2 每安培最大扭矩 (MTPA) 控制

场定向控制 (FOC) 内的 MTPA（每安培最大扭矩）控制是一种适用于凸极电机的高级策略。MTPA 可优化电流注入，以在极小的输入电流下产生更大的扭矩，从而更大限度地提高电机控制效率。用户可以按照凸极参数说明中的详细描述，将电机的凸极配置为非零值（节 7.2.3）。

可以通过将 ***pUserInputRegs*** ->fieldCtrl.b.mtpaEnable 设置为 1b 来启用该特性。

7.7.2.3 弱磁控制 (FWC)

弱磁控制通过降低电机激励磁场的强度来扩大电机的速度范围。尤其是在超过额定速度后，通过注入与电机 d 轴磁场相对的电流分量，总 d 轴磁场会减弱，允许电机在电压限制下继续加速。

备注

在磁通减弱运行期间，电机无法提供额定扭矩。扭矩限值 Iq 根据 ILIMIT = SQRT(Id^2 + Iq^2) 定义的循环电机电流限制自动调整。

可以通过将 ***pUserInputRegs*** ->fieldCtrl.b.fluxWeakeningEn 设置为 1b 来启用该特性。

按照以下步骤调优弱磁控制方法：

1. 将 pUserInputRegs->fieldCtrl.b.fluxWeakeningEn 设置为 1b 以启用弱磁
2. 使用 pUserInputRegs->fieldCtrl.b.fluxWeakCurrRatio 来确定最大弱磁 d 轴电流。设置更大的弱磁 d 轴电流会带来电机永磁体退磁的风险。
3. 使用 pUserInputRegs->fieldCtrl.b.fluxWeakeningReference 确定最大调制指数，超过该指数时启用弱磁。设置较小的速度阈值会导致效率降低。

启用弱磁控制后，会应用一个单独的 PI 控制器进行 Id 基准设置。可使用 ***pUserInputRegs*** ->systemParams.fluxWeakeningKp 和 ***pUserInputRegs*** ->systemParams.fluxWeakeningKi 调优弱磁控制。

7.7.2.4 死区时间补偿

在半桥桥臂中高侧和低侧 MOSFET 的开关瞬间之间插入了死区时间，以避免发生击穿情况。

由于存在死区时间插入，相节点上的预期电压与施加的电压会因相电流方向而异。相节点电压失真会在相电流中引入不必要的失真，进而导致可闻噪声。由于死区时间而导致的电流波形失真在 d-q 轴坐标系中显示为基频的六次谐波。

FOC 算法集成了专有死区时间补偿技术，利用谐振控制器将相电流中的六次谐波分量控制为零，从而确保缓解因死区时间导致的电流失真。iq 和 id 控制路径中都包含谐波控制器。可以通过配置 ***pUserInputRegs*** ->closeLoop1.b.deadTimeCompEn 来启用或禁用死区时间补偿。

7.7.2.5 PWM 生成模式

FOC 算法支持两种不同的调制方案，即连续和不连续空间矢量 PWM 调制方案。

在连续 PWM 调制中，全部三个相位始终会按照定义的开关频率进行开关。在非连续 PWM 调制中，其中一个相位会在 120° 电气周期内钳位至地，而另外两个相位会进行脉宽调制。调制方案使用 ***pUserInputRegs*** ->closeLoop1.b.pwmMode 配置。

下图展示了不同调制方案的调制后平均相电压。

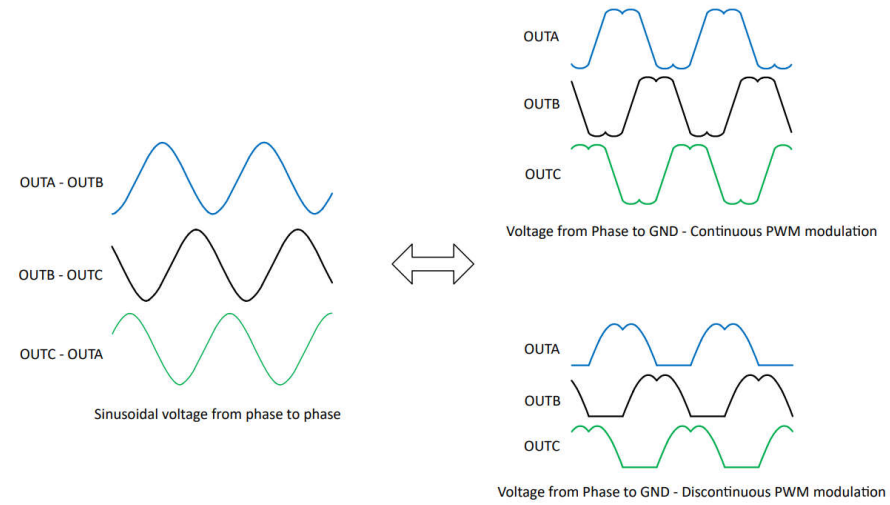


图 7-29. 连续和非连续 PWM 调制相位电压

连续调制有助于减小具有低电感的电机的电流纹波，但由于所有三个相位都在开关，因此会导致更高的开关损耗。由于一次只有两个相位互相交错，因此非连续调制具有更低的开关损耗，但其电流波纹更高。

7.7.2.6 过调制模式

FOC 算法提供了过调制选项，可通过适当修改应用的 PWM 模式来增加施加的基波相电压，从而在相同 VM 电压下以更高的速度运行电机 — 更高的基波相电压伴随着更高次谐波增加。

可以通过将 ***pUserInputRegs*** ->closeLoop1.b.overModEnable 设置为 1b 来启用该特性。

7.7.2.7 初始速度检测 (ISD) 模式

ISD 功能用于标识电机的初始状态，可以通过将 ***pUserInputRegs*** ->isdCfg.b.isdEn 设置为 1b 来启用。初始速度、位置和方向通过由 MSPM0 ADC 对相电压进行采样来决定。此功能对风扇应用很有用。

如果禁用了该功能则 FOC 算法不会执行初始速度检测功能，而继续检查制动例程 [***pUserInputRegs*** ->isdCfg.b.brakeEn] 是否已启用。

7.7.2.7.1 电机重新同步

当同时启用 ISD 和重新同步功能并且器件判断电机初始状态为正向旋转（与命令方向相同）时，电机重新同步功能起作用。ISD 期间测量的速度和位置信息用于初始化驱动状态，该驱动状态可以直接切换至闭环（如果电机速度不足以进行闭环运行，则为开环）状态，无需将电机停机。

将 ***pUserInputRegs*** ->isdCfg.b.resyncEn 设置为 1b 可启用电机重新同步。如果电机重新同步被禁用，那么固件将继续检查是否启用了电机滑行（高阻态）例程 [***pUserInputRegs*** ->isdCfg.b.hiZEn]。

7.7.2.7.2 反向驱动

当 ***pUserInputRegs*** ->isdCfg.b.rvsDrEn 和 isdEn 都设置为 1b 并且 ISD 确定电机旋转方向与命令方向相反时，FOC 算法使用反向驱动功能来改变电机的旋转方向。此功能仅在**无传感器 FOC** 中有效。

反向驱动包括在相反的方向与电机速度同步，对电机进行反向减速至越过零速，改变方向，以及在正向（或命令方向）在开环中加速，直到器件在正向切换为闭环。

FOC 算法通过配置 ***pUserInputRegs*** ->rvsDrvCfg.b.revDrvConfig 来提供使用正向参数或一组单独的反向驱动参数的选项。

如果电机无法在反方向重新同步，请遵循以下建议：

1. 增加反向减速速度阈值，以切换至开环 [pUserInputRegs->isdCfg.b.revDrvHandoffThr]
2. 启用开环反向驱动配置 (revDrvConfig)
3. 增加反向驱动开环电流基准 [pUserInputRegs->isdCfg.b.revDrvOpenLoopCurr]
4. 减小反向驱动期间的开环加速系数 A1 和 A2 [pUserInputRegs->rvsDrvCfg.b.revDrvOpenLoopAccelA1 和 revDrvOpenLoopAccelA2]

下图展示了反向驱动切换下的电机速度曲线。

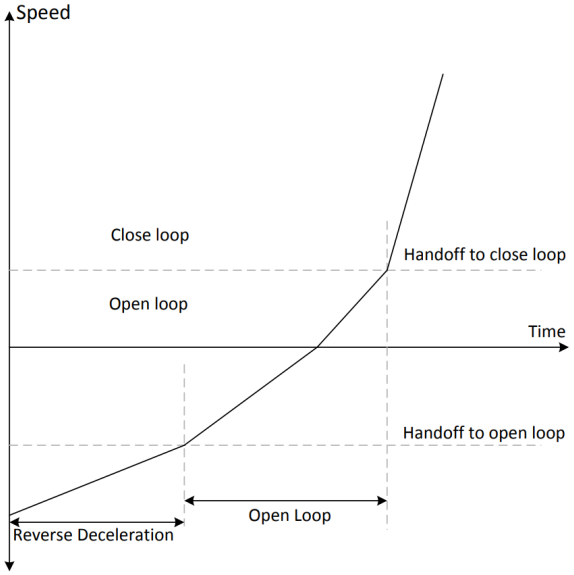


图 7-30. 反向驱动切换下的电机速度曲线

7.7.2.7.3 快速 ISD

FOC 应用通过将 ***pUserInputRegs*** ->miscAlgo.b.fastIsdEnable 设置为 1b 来支持快速 ISD 功能（仅在**无传感器 FOC** 中可用）。下表展示了 ISD 与快速 ISD 之间的差异。

表 7-14. ISD 功能比较

算法	说明
ISD	基本 ISD 功能支持在 1 个电气周期内重新同步电机位置。
快速 ISD	快速 ISD 功能支持在 60 度电角度范围内快速重新同步电机位置。

7.7.2.8 防电压浪涌

驱动电机时，能量从电源传输到电机。其中一些能量以电感能量和机械能的形式进行存储。如果命令的速度骤降，使得电机产生的 BEMF 电压大于施加到电机上的电压，电机的机械能将返回到电源，从而产生 VM 电压浪涌。

AVS 特性旨在防止在 VM 上产生该电压浪涌，可以通过 ***pUserInputRegs*** ->closeLoop1.b.avsEn 来启用。当禁用 AVS 时，电机减速率通过 ***pUserInputRegs*** ->closeLoop1.b.clDec 进行配置。

7.8 覆盖用户输入寄存器表

在 FOC 应用中，用户根据寄存器配置来实现功能，寄存器的值代表不同的物理含义，如寄存器宏（节 5.3.2）所示。FOC 应用在“ ***configTables.c*** ”文件中以常数数组格式提供部分寄存器映射，如图 7-31 所示。用户可以覆盖数组中的值，为 FOC 应用设置其自定义寄存器表。

```c
#include "configTables.h"

/* ISD Config Tables*/
/*! @brief Table for forward and reverse drive */
const uint16_t tbl_fwRevDrv_pMil[16]    = {50,100,150,200,250,300,350,400,450,500,550,600,700,800,900,1000};

/*! @brief Table for HiZ brake time */
const uint16_t tbl_hiZ_brk_Time_ms[16] = {10,50,100,200,300,400,500,750,1000,2000,3000,4000,5000,7500,10000,15000};

/*! @brief Table for Stall detection threshold */
const _iq tbl_StatDetectThr_pu[8] = {_IQ(0.00104),_IQ(0.003125),_IQ(0.0041),_IQ(0.0104),_IQ(0.0208),_IQ(0.03125),_IQ(0.0416),_IQ(0.0625)};

/*Closed Loop PWM Frequency Table */
const uint16_t tbl1_clPWMFreqKHz[16] = {5,10,16,20,25,32,40,48,50,64,80,80,80,80,80,80};

/* Open Loop Speed and Acceleration Slew Rates */
/*! @brief Table 1 for open loop acceleration */
const uint16_t tbl1_olAccA1A2_centiHzPerSec[8] = {1, 5, 100, 250, 500, 1000, 2500, 5000};
/*! @brief Table 2 for open loop acceleration */
const uint16_t tbl2_olAccA1A2_HzPerSec[8] = {75, 100, 250, 500, 750, 1000, 5000, 10000};

/* Close Loop Speed and Acceleration Slew Rates */
/*! @brief Table 1 for close loop acceleration and deceleration */
const uint16_t tbl1_clDecClAcc_deciHzPerSec[16] = {5,10,25,50,75,100,200,400,600,800,1000,2000,3000,4000,5000,6000};
/*! @brief Table 2 for close loop acceleration and deceleration */
const uint16_t tbl2_clDecClAcc_HzPerSec[14] = {700,800,900,1000,2000,4000,6000,8000,10000,20000,30000,40000,50000,60000};

/* Close Loop 2 */
/*! @brief Table for brake speed threshold and active spin threshold */
const uint16_t tbl_brkDutyActSPinThr_pMil[16] = {1000,900,800,700,600,500,450,400,350,300,250,200,150,100,50,25};

/* Motor StartUp1 */
/*! @brief Table for ipd clock frequency */
const uint16_t tbl_ipdClkFreq_Hz[8] = {50,100,250,500,1000,2000,5000,10000};

/*! @brief Table for align time */
const uint16_t tbl_alignTime_msec[16] ={10,50,100,200,300,400,500,750,1000,1500,2000,3000,\
                                        4000,5000,7500,10000};

/*! @brief Table for align and slow cycle start ramp rate */
const uint16_t tbl_alignSlowRampRate[16] = {1,10,50,100,150,250,500,1000,1500,2000,2500,5000,10000,20000,50000,0};

/* Motor Start Up2 */
/*! @brief Table for theta error ramp rate */
const uint16_t tbl_thetaErrRampRate_mili[8]  = {10,50,100,150,200,500,1000,2000};

/*! @brief Table for align angle */
const uint16_t tbl_alignAngle[32] = {0,10,20,30,45,60,70,80,90,110,120,135,150,160,170,180,190, \
                                     210,225,240,250,260,270,280,290,315,330,340,350,350,350,350};

/*! @brief Table for slow first cycle frequency */
const uint16_t tbl_slowFirstCycFreqPerMil[16]  = {10,20,30,50,75,100,125,150,175,200,250,300,350,400,450,500};

/* Fault Config 1 and Config2 */
/*! @brief Table for abnormal speed lock */
const uint16_t tbl_lckAbnormalSpeed_pMil[8] = {1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000};

/*! @brief Table for abnormal backemf threshold */
const uint16_t tbl_AbnormalBEMFThr_pMil[8] = {10,20,30,50,80,100,120,150};

/* Internal algo 1 Params*/

/*! @brief Table for isd stop time and isd run time */
const uint16_t tbl_isdStopTime_msec[4]     = {1,5,50,100};

/*! @brief Table for isd timeout */
const uint16_t tbl_isdRunTime_msec[4]     = {100,250,500,1000};

/*! @brief Table for autohandoff minimum backemf */
const uint16_t tbl_autoHandOffMinBemf_mV[8]   = {0,50,100,250,500,1000,1250,1500};

/*! @brief Table for persistent brake current */
const uint16_t tbl_brakeCurrPersist_msec[4]   = {50,100,250,500};

/* Internal algo 2 Params*/
/*! @brief Table for close loop alow acceleration and deceleration */
const uint16_t tbl_clSlowAcc_deci[16]     = {1,10,20,30,50,100,200,300,400,500,1000,2000,5000,7500,10000,20000};

/*! @brief Table for current */
const _iq tbl_pu[32] = {_IQ(0.075),_IQ(0.080),_IQ(0.085),_IQ(0.090),_IQ(0.095),_IQ(0.100),_IQ(0.110),_IQ(0.120),   \
                        _IQ(0.130),_IQ(0.140),_IQ(0.150),_IQ(0.160),_IQ(0.170),_IQ(0.180),_IQ(0.200),_IQ(0.225),   \
                        _IQ(0.250),_IQ(0.275),_IQ(0.300),_IQ(0.350),_IQ(0.400),_IQ(0.450),_IQ(0.500),_IQ(0.550),   \
                        _IQ(0.600),_IQ(0.700),_IQ(0.750),_IQ(0.800),_IQ(0.850),_IQ(0.900),_IQ(0.950),_IQ(1.000)};

/*! @brief Table for minimum VM */
const _iq tbl_minVm_pMil[8] = {_IQ(0.0),_IQ(0.05),_IQ(0.10),_IQ(0.12),_IQ(0.15),_IQ(0.18),_IQ(0.20),_IQ(0.25)};

/*! @brief Table for maximum VM */
const _iq tbl_maxVm_pMil[8] = {_IQ(0.60),_IQ(0.65),_IQ(0.70),_IQ(0.75),_IQ(0.80),_IQ(0.85),_IQ(0.90),_IQ(1.0)};

/*! @brief Table for Modulation Limit Beyond which Flux weakening is enabled */
const _iq tbl_mSqrRef[4] = {_IQ(0.49), _IQ(0.64),_IQ(0.81), _IQ(0.902)};

/* Misc algo 1 Params*/
/*! @brief Table for IPD maximum overflow */
const uint8_t tbl_ipdMaxOverflow[4] = {5, 10, 20, 40};
```

图 7-31. 配置表文件概述

例如，如果用户希望根据应用要求将对齐时间设置为 150ms，而默认寄存器映射仅提供 100ms 或 200ms 选项。那么用户可以将 tbl_alignTime_msec 放入 ***configTables.c*** 中，并将 tbl_alignTime_msec[3]从 200 重写为 150。这样，现在通过将 ***pUserInputRegs*** ->mtrStartUp1.b.alignTime 设置为 3h，就可以将对齐时间设置为 150ms。

8 硬件迁移指南

本节帮助用户将 SDK 示例工程迁移到自己的客户板。

8.1 硬件层概述

下图展示了 FOC 工程的硬件配置关系。硬件层包含三个部分：板层、SysConfig 层和 HAL 层。

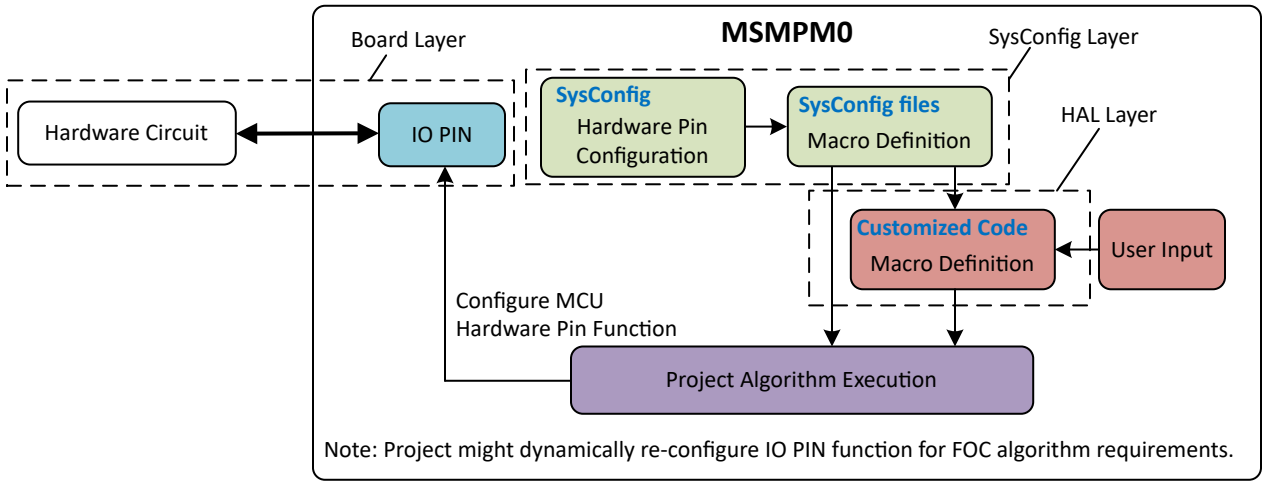


图 8-1. 硬件配置关系

板层是用户将 MCU 与硬件电路中其他元件相连接的层。对于 FOC 应用，它主要用于连接所用的栅极驱动器器件和 ADC 采样通道。

SysConfig 层在 . ***syscfg*** 文件中设置 MCU 外设初始化特性。连接到外部电路的 IO 引脚需要设置为相应的外设功能，才能成功运行 FOC 应用。SDK 工程提供了多种外设默认配置来适应 DRV EVM 板或 TIDA 参考板。用户需要配置 . ***syscfg*** 文件来手动调整硬件电路。SysConfig 工具会自动生成 MCU 外设初始化文件，包括映射到 MCU 硬件外设的宏定义。

第二个 HAL 层是 SDK FOC 工程中的自定义宏定义。许多 FOC 函数会调用自定义宏定义来确定算法的实现。因此，用户需要在头文件中手动管理这些自定义宏定义，以适应硬件电路和 SysConfig 生成的宏定义。

可按照以下步骤为自定义电路板迁移硬件配置：

1. 检查自定义电路板上每个模块使用的 MSPM0 IO 引脚。
2. 修改 SysConfig 配置以适合硬件设计。
3. 修改 HAL 层宏定义以适应硬件设计。

8.2 栅极驱动器模块

FOC 应用使用预定义符号来确定使用哪个栅极驱动器板，以正确配置电路板参数和 HAL 层，如下图所示。

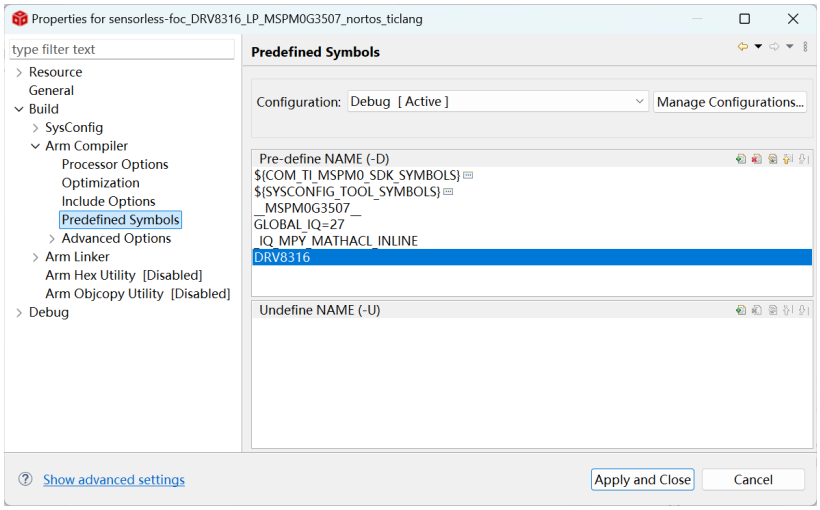


图 8-2. 工程属性中的 DRV8316 预定义符号

默认栅极驱动器接口是为 EVM 硬件板配置的。用户可以根据自己的电路板配置，覆盖为 DRV8316 设置的相关配置。

同时，如果使用另一个栅极驱动器，建议在用户的特定硬件设计中使用 **CUSTOM** 符号，以便更好地管理代码。按照以下步骤使用此方法配置定制栅极驱动器模块。

8.2.1 选择参考工程

MSPM0 SDK 为不同的硬件板提供各种 FOC 示例工程。请参阅下表，选择首选的示例工程来为您自己的硬件板进行迁移。

表 8-1. 关于迁移的工程建议

FOC 类型	自定义硬件板		工程建议	迁移工作
	栅极驱动器	电流检测类型		
无传感器/通用 FOC	DRV8323	单分流器 (1)	sensorless-foc_DRV8323RS	移植单分流器配置。
			sensorless-foc_DRV8329	移植栅极驱动器配置。
		双或三分流器	sensorless-foc_DRV8323RS	无需大量工作。
	DRV8316	单分流器 (1)	sensorless-foc_DRV8316	移植单分流器配置。
			sensorless-foc_DRV8329	移植栅极驱动器配置。
		双或三分流器	sensorless-foc_DRV8316	无需大量工作。
	其他	单分流器	sensorless-foc_DRV8329	无需大量工作。
		双或三分流器	sensorless-foc_TIDA010250	无需大量工作。
带传感器 FOC	DRV8316	单分流器 (1)	hall_sensored-foc_DRV8316	移植单分流器配置。
			hall_sensored-foc_TIDA010251	移植栅极驱动器配置。
		双或三分流器	hall_sensored-foc_DRV8316	无需大量工作。
	其他	单分流器	hall_sensored-foc_TIDA010251	无需大量工作。
		双或三分流器	hall_sensored-foc_DRV8316	无需大量工作。

1. 如果用户的硬件板具有单分流器电路，建议使用具有单分流器配置的示例工程开始迁移。

以下章节将以 TIDA010250 为例，介绍自定义栅极驱动器模块迁移的流程。

8.2.2 修改预定义符号

在工程属性中，修改默认的预定义符号：TIDA010250->CUSTOM。

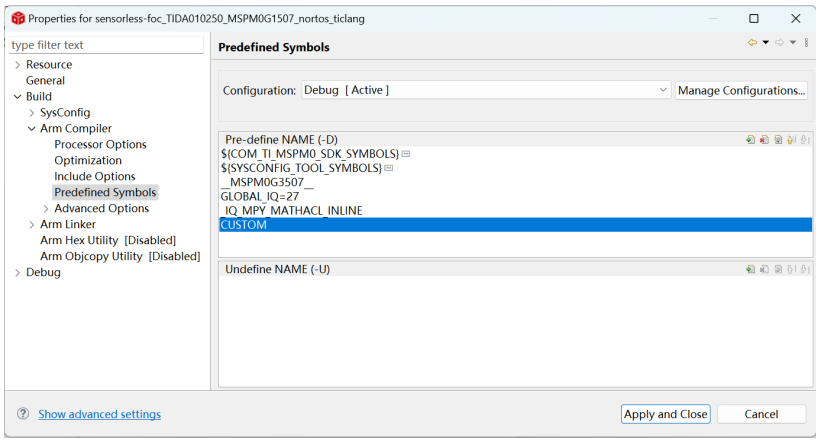


图 8-3. 在工程属性中修改预定义符号

8.2.3 添加自定义源文件

默认情况下，FOC 工程在 FOC 库中为栅极驱动器集成了 SPI 通信功能，如果使用自定义栅极驱动器，用户可以保留用于栅极驱动器通信的源代码。为了帮助用户快速删除默认 SPI 通信，FOC 工程在 SDK 文件夹中提供了自定义源文件。用户可以使用 **CUSTOM** 预定义符号手动将源文件添加到自定义工程中。

8.2.3.1 栅极驱动器通信文件夹

栅极驱动器通信的预设自定义源文件位于以下 SDK 路径：

```
...\ti\mspm0_sdk_<SDK_Version>\source\ti\motor_control_pmsm_foc\common_modules\hal\LP_MSPM0Gx5xx\gateDriverInterface\gateDriverLib
```

图 8-4 展示了 CUSTOM 栅极驱动器库文件夹。该库删除了 FOC 应用中用于栅极驱动器 SPI 通信的默认代码，以便用户可以在其自定义文件中添加自己的栅极驱动器通信代码（节 8.2.4）。

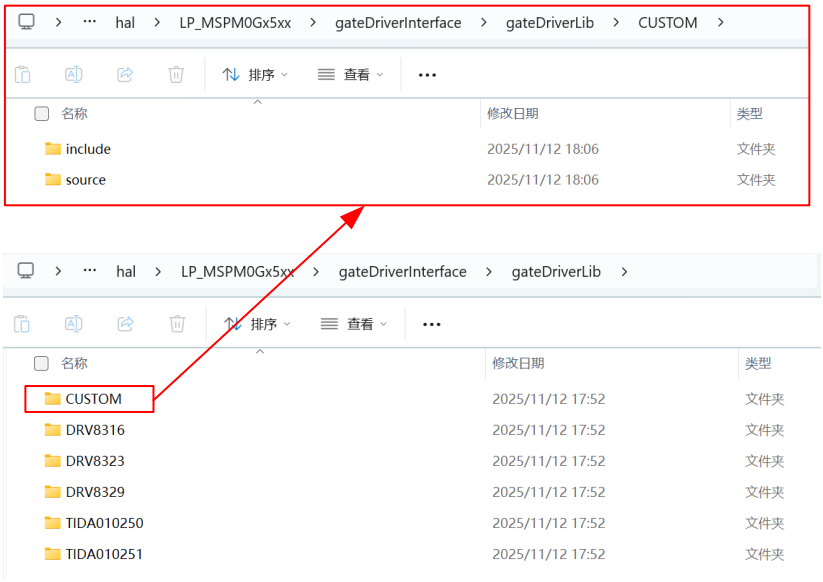


图 8-4. 自定义栅极驱动器库

将 CUSTOM 文件夹（复制并粘贴）添加到示例 FOC 工程中，如图 8-5 所示。可以删除默认的栅极驱动器文件夹 (TIDA010250)（可选）。

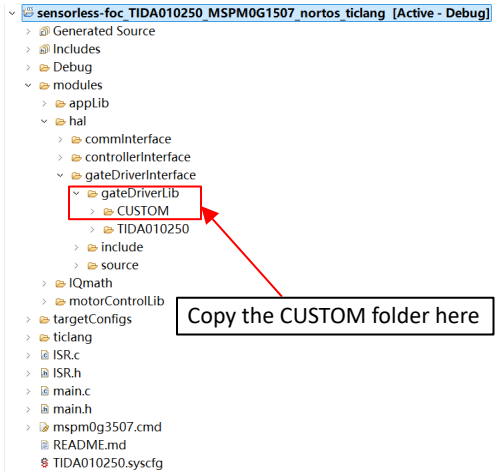


图 8-5. 将 CUSTOM 文件夹添加到 FOC 工程中

在下方工程属性中的 Include Options 中添加 CUSTOM 文件夹路径：

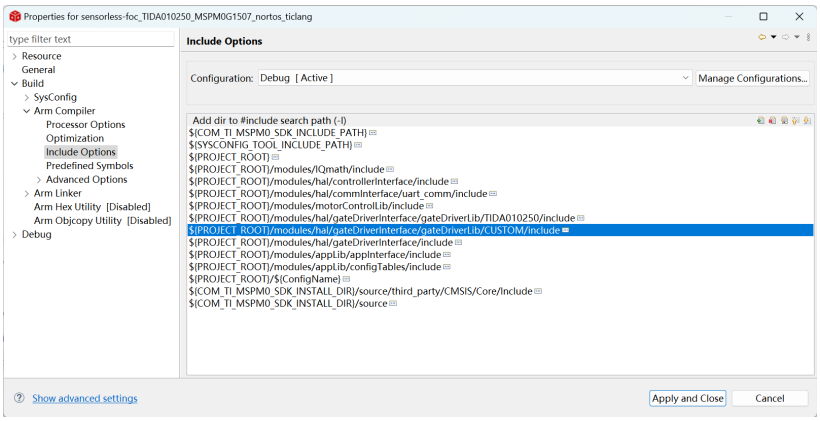


图 8-6. 在 Include Options 中添加 CUSTOM 文件夹

8.2.3.2 HAL 层文件

MSPM0 SDK 在以下路径为 HAL 层提供了预设的自定义源文件：

```
...\ti\mspm0_sdk_<SDK_Version>\source\ti\motor_control_pmsm_foc\common_modules\hal\LP_MSPM0Gx5xx\gateDriverInterface\[LAUNCHPAD]\source
```

图 8-7 展示了该自定义 HAL 层文件。

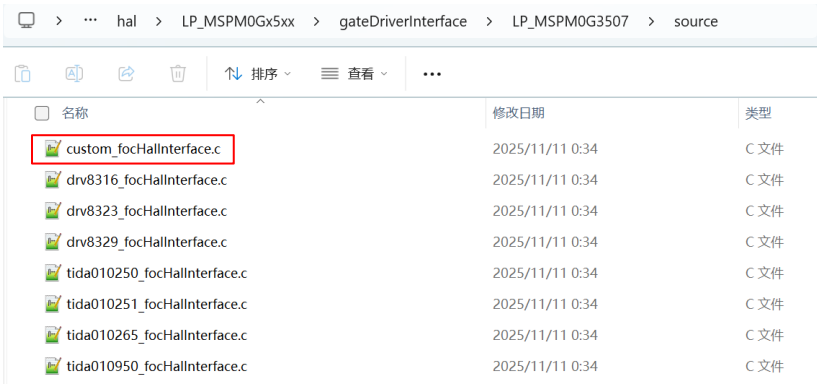


图 8-7. 自定义 HAL 层文件

将 ***custom_focHallInterface.c*** 文件添加到 FOC 工程的 ***gateDriverLib*** 文件夹，如下所示：

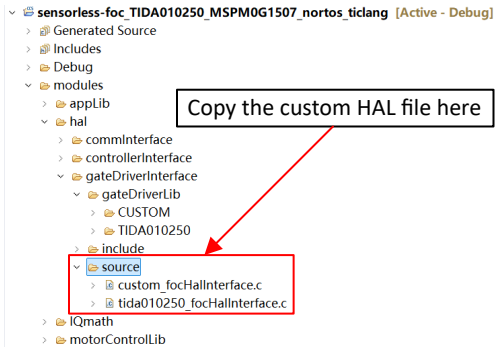


图 8-8. 将自定义 HAL 文件添加到 FOC 工程中

8.2.4 添加自定义通信接口

栅极驱动器支持配置不同的通信接口，包括基于电阻器、基于 SPI、基于 IIC 的接口等。对于用户来说，将新通信接口集成到 FOC HAL 库并进行调试是很复杂的。该设计引入了一种更轻松的方法来禁用默认的基于 SPI 的通信（DRV8316 或 DRV8323 FOC 工程），以便 FOC 算法不会使用任何 MCU 引脚或外设来进行通信。

可按照节 8.2.3.1 中的步骤使用自定义通信接口。然后将栅极驱动器通信 API 设置为空，使其不会影响正常的 FOC 应用，如图 8-9 所示。用户还可以手动将栅极驱动器通信接口的源代码修改为空，然后在 FOC 应用中添加自己的栅极驱动器文件，以适配自己的硬件板。

custom.c

```
#include <math.h>
#include "custom.h"

#include "main.h"
#include "ISR.h"
#include "focHALInterface.h"

#ifdef CUSTOM
uint32_t gateDrivernFaultReport = 0;
uint32_t gateDriverFaultAction = 0;

void gateDriverConfig(void)
{


}

void gateDriverClearFault(void)
{

}


uint32_t gateDriverGetFaultStatus(void)
{
    return 0;
}
/*Update the Gate Driver Parameters when the Config Enable Flg is Set */
void gateDriverParamsUpdate(HV_DIE_EEPROM_INTERFACE_T *pGateDriverConfig)
{

}

```

There provide an empty code implementation.

图 8-9. 自定义栅极驱动器文件

备注

直接删除上图中所示的 API 会导致编译器错误，因为无传感器 FOC 库会在静态库中调用这些 API。

8.2.5 覆盖默认宏定义

8.2.5.1 main.h 文件

在此 SDK 版本中， ***main.h*** 缺失用于用户自定义的 CUSTOM 宏。添加以下相关代码（从其他宏（例如 TIDA-010250）复制并使用 CUSTOM 覆盖该值）：

Manually add CUSTOM parts

```
217
218#elif defined CUSTOM
219
220/*! @brief TIDA010250 has inverting isense */
221#define _INVERT_ISEN
222/*! @brief IPD feature is enabled */
223#define __IPD_ENABLE
224/*! @brief TIDA010250 propagation delay */
225#define DRIVER_PROPAGATION_DELAY_nS                500
226/*! @brief TIDA010250 minimum on time (rise time + settling time) */
227#define DRIVER_MIN_ON_TIME_nS                      8000
228/*! @brief DC voltage base value */
229#define DC_VOLTAGE_BASE                             448.0
230/*! @brief Full scale readable current used as current base value,
231calculated using (FULL Scale Voltage(3.3)/2* CSA Gain) */
232#define FULL_SCALE_CURRENT_BASE                     8.25
233/*! @brief Current shunt configuration */
234#define __CURRENT_THREE_SHUNT_DYNAMIC
235/*! @brief Enable dynamic current shunt changing */
236#define DYNAMIC_CURRENT_SHUNT_CONFIG_EN             FALSE//TRUE
237
238#endif
```

图 8-10. 将自定义宏添加到 main.h 文件中

用户应根据硬件板功能修改宏定义，如表 8-2 所示。

表 8-2. FOC 应用的硬件宏定义

硬件特性	宏	说明
电流检测类型	_INVERT_ISEN	该电路板具有一个反相或非反相电流检测硬件电路。请参阅节 8.3.2.1。
	_NONINVERT_ISEN	
电流检测方法	__CURRENT_XX_SHUNT	电流检测方法包括单分流器、双分流器和三分流器。请参阅节 8.3.2.2。
电路板参数	DC_VOLTAGE_BASE	最大可测量总线电压。请参阅节 7.1.1。
	FULL_SCALE_CURRENT_BASE	基极电流。请参阅节 7.1.2。
	DRIVER_PROPAGATION_DELAY_nS	定义馈送到栅极驱动器的输入 PWM 逻辑边沿与实际栅极驱动器输出之间的时间延迟，以 ns 为单位。
	DRIVER_MIN_ON_TIME_nS	定义电流检测放大输出的合并上升时间和趋稳时间。
IPD 功能	__IPD_ENABLE	为 FOC 应用启用 IPD 功能。应正确设置 IPD 配置寄存器，请参阅节 7.7.1.1.3。

8.2.5.1.1 电流检测路径中的延迟分量

下图展示了电流测量路径中的延迟分量。

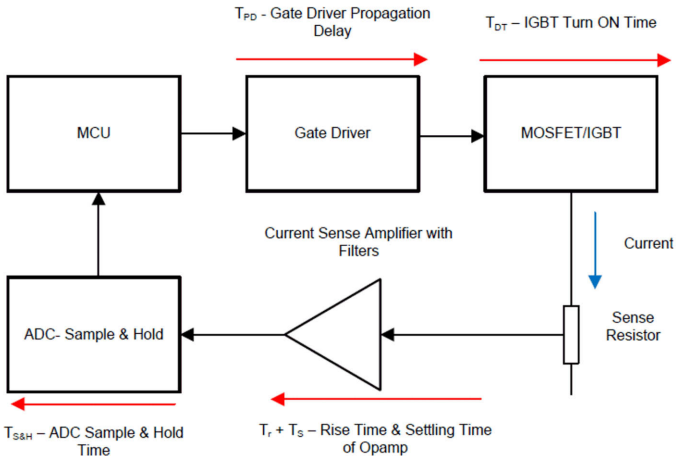


图 8-11. 电流检测环路中的延迟

FOC 应用使用 DRIVER_PROPAGATION_DELAY 来定义馈送到栅极驱动器的输入 PWM 逻辑边沿与实际栅极驱动器输出 PWM 之间的时间延迟，以 ns 为单位。此延迟影响实际栅极驱动器输出的电流检测采样实例，必须馈送到算法以进行精确的电流检测。用户可以从所用栅极驱动器的数据表中获取传播延迟，或使用硬件测量该时间。

FOC 应用使用 DRIVER_MIN_ON_TIME 来定义电流检测放大输出的合并上升时间和趋稳时间。如果电流分流器两端的电压发生满量程变化，则必须单独采集此值。为了获得准确的电流检测读数，在捕获电流信号之前电流检测放大器输出应稳定下来。请参阅方程式 45 来计算 DRIVER_MIN_ON_TIME。

$$\text{DRIVER\_MIN\_ON\_TIME} = \text{CSASettlingTime} + \text{CASRiseTime} \tag{45}$$

备注

DRIVER_MIN_ON_TIME 限制最大 FOC 输出 PWM 占空比，以便为 CSA 保留足够的时间来建立稳定的电流信号馈入 ADC 采样通道。

8.2.5.2 gateDriver.h 文件

gateDriver.h 文件定义了 HAL 层宏。用户必须根据其硬件电路板电路来覆盖这些宏。

- 有关与 PWM 相关的宏定义覆盖，请参阅节 8.3.1。
- 有关与 ADC 相关的宏定义覆盖，请参阅节 8.3.2。

8.3 MCU 外设配置

8.3.1 PWM 模块

使用 MSPM0 的 TIMA0 为 FOC 应用启用三对互补的 PWM 输出和死区时间插入。TIMA0 有四个输出通道，用户可以为 FOC 应用选择其中任意三个通道。

硬件电路最终确定后，用户应注意电机相位序列和 PWM 通道设置之间的对应关系。**sensorless-foc_DRV8316_LP_MSPM0G3507** 工程的默认映射关系如下表所示。

表 8-3. PWM 映射

板层		HAL 层		说明
	SysConfig 层			
板宏	IO 引脚	PWM 通道	HAL 层宏	
INHA	PB4	TIMA0_C2	FOC_PWMA0_U_IDX	A 相 PWM 输出。

表 8-3. PWM 映射（续）

板层		HAL 层		说明
	SysConfig 层			
板宏	IO 引脚	PWM 通道	HAL 层宏	
INHB	PA28	TIMA0_C3	FOC_PWMA0_V_IDX	B 相 PWM 输出。
INHC	PB20	TIMA0_C1	FOC_PWMA0_W_IDX	C 相 PWM 输出。
FAULT	PA27	故障引脚 2	未使用	栅极驱动器输出故障引脚。
不适用	不适用	TIMA0_C0	FOC_PWMA0_ADC_TRIG_IDX	从 PWM 到 ADC 的触发通道。

HAL 层宏始终保持与板宏的相同映射关系，并且应相应地修改 SysConfig 层。

备注

在当前 FOC 应用中，未使用的 PWM 通道配置为 ADC 触发通道，但工程中启用了 PIN 输出功能。用户可以在主循环代码中手动禁用它，或者使用非输出通道（CC4 或 CC5）来消除这种 PWM 引脚输出影响。

8.3.1.1 用于 PWM 输出的不同引脚

如果用户仅在硬件中更改不同的 PWM 输出引脚，而映射关系保持不变，则用户只需在 SysConfig 工具中修改 TIMA0output 通道的 PINMUX 选择，如下图所示。

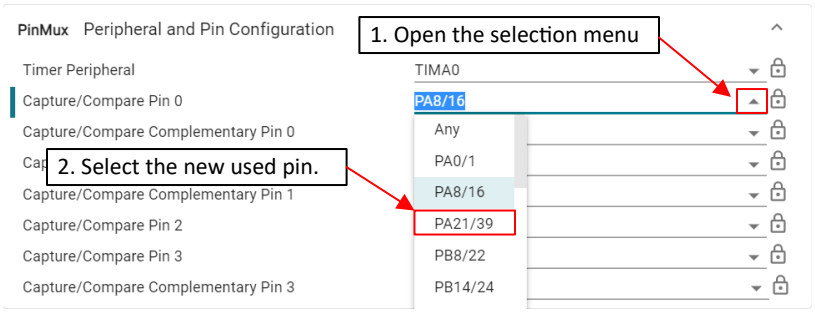


图 8-12. 在 SysConfig 中修改 PWM 输出引脚

8.3.1.2 用于 PWM 故障输入的不同引脚

如果用户在硬件中更改不同的 PWM 故障引脚，用户只能在 SysConfig 工具中修改 TIMA0fault 引脚的 PINMUX 选择，如图 8-13 所示。

如果硬件电路不支持故障保护，请取消选择“Enable Fault Handler”以将其禁用。不正确的故障信号（悬空或低脉冲）可能会在旋转电机时触发 HV_DIE 故障。

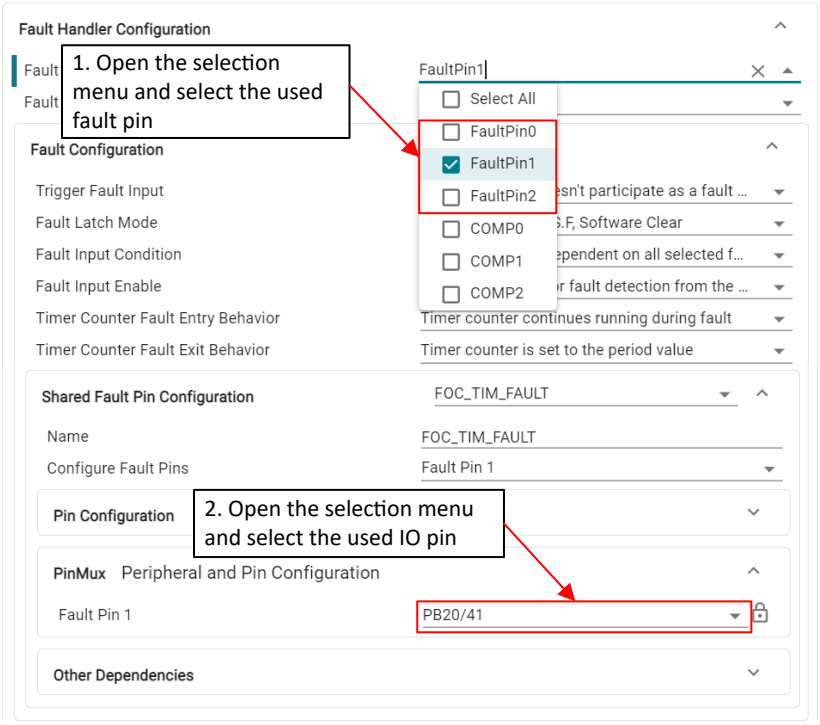


图 8-13. 在 SysConfig 中修改 PWM 故障引脚

8.3.1.3 与 PWM 输出通道的不同映射

对于硬件中的 PWM 输出通道映射不同的情况，用户应首先参阅节 8.3.1.1 来修改 SysConfig 中使用的 IO 引脚。然后定位到 ***gateDriver.h*** 文件，以覆盖 HAL 层宏。

例如，如果用户拥有下面的新映射表：

表 8-4. 自定义 PWM 映射

板宏	IO 引脚	PWM 通道	HAL 层宏
INHA	待定（在 SysConfig 中设置）	TIMA0_C0	FOC_PWMA0_U_IDX
INHB	待定（在 SysConfig 中设置）	TIMA0_C1	FOC_PWMA0_V_IDX
INHC	待定（在 SysConfig 中设置）	TIMA0_C2	FOC_PWMA0_W_IDX
不适用	不适用	TIMA0_C3	FOC_PWMA0_ADC_TRIG_IDX

下图展示了 ***gateDriver.h*** 文件中的相应修改。

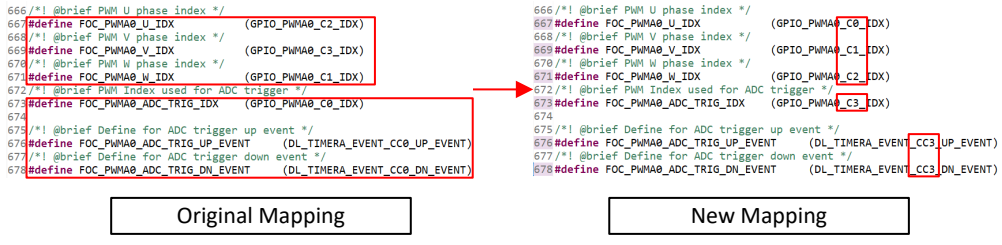


图 8-14. 修改 HAL 层宏

其余未使用的 PWM 输出通道始终用于在双分流器或三分流器电流检测方法中触发 ADC 采样。并且 ADC 触发事件应设置为触发 PWM 通道的事件。

对于单分流器电流检测方法，FOC 应用使用辅助 TIMA1 来触发 ADC，因此无需修改触发通道。

8.3.2 ADC 模块

将 ADC 模块迁移到自定义设置时，用户应注意硬件电路中使用的映射关系以及电流检测方法。

8.3.2.1 电流检测类型

用户应根据 ***main.h*** 文件中的硬件检测电路类型覆盖工程中的宏定义，如下表所示。

表 8-5. 电流检测类型

文件	宏定义	说明
main.h	_NONINVERT_ISEN	硬件板使用非反相电流检测电路
	_INVERT_ISEN	硬件板使用反相电流检测电路

图 8-15 展示了典型双向电流检测电路方框图，可供用户参考。

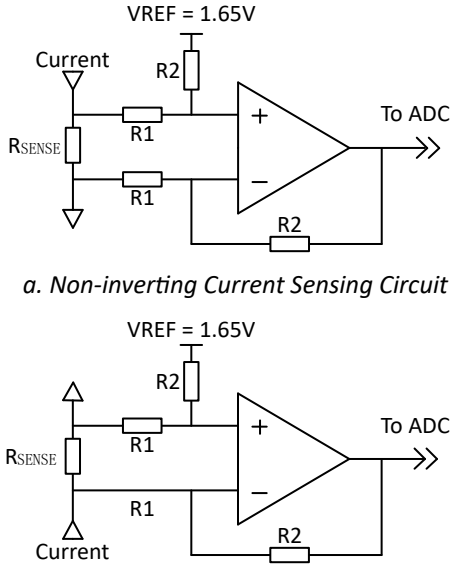


图 8-15. 电流检测类型

8.3.2.2 电流检测方法

SDK FOC 示例可针对单分流器、双分流器和三分流器等各种分流器配置选项进行配置。为确保算法正常运行，必须根据硬件设计选择适当的分流器配置。FOC 应用支持在给定实例中同时对两个相位采样，从而优化电流采样时间。默认情况下，在所有分流器配置中，同时通过两个 ADC 实例来使用同步采样功能。

8.3.2.2.1 三分流器配置

下表展示了三分流器配置的宏。

表 8-6. 用于三分流器检测的宏

文件	宏定义	说明
main.h	__CURRENT_THREE_SHUNT_AB_C	如果通过 ADC0 检测 A 相和 B 相，而通过 ADC1 检测 C 相，则选择此配置。
	__CURRENT_THREE_SHUNT_A_BC	如果通过 ADC0 检测 A 相，而通过 ADC1 检测 B 相和 C 相，则选择此配置。

8.3.2.2.2 采用同步采样的三分流器配置

用户也可以将其中一个相位（比如“B”）同时路由到 ADC0 和 ADC1 实例，并将其他两个相位路由到两个不同的 ADC 实例。

例如，A 相路由到 ADC0，C 相路由到 ADC1。B 相同时路由到 ADC0 和 ADC1 实例。然后，算法可以动态切换到两个样本，如果启用了同步采样，这样可以根据给定的扇区提供最佳的电流采样时间。如果用户希望减少 1 个用于同步采样功能的 ADC 引脚，请参阅节 9.3。

在同步采样模式下，FOC 应用支持将电流检测估算动态地转移到两个相位，从而更大限度地提高调制指数。与平衡三相电机一样，三相电流中的任何一个都可以使用方程式 46 中的另外两个相电流来估算。

$$I_a + I_b + I_c = 0 \tag{46}$$

根据运行的扇区，选择具有最低调制指数的两个相位进行电流测量，并使用这两个相电流估算具有最高调制指数的第三个相位。此方法有助于通过持续 SVM 运行将调制指数扩展到更高的限值。

下表给出了启用同步采样的示例。

表 8-7. 用于同步采样的宏

文件	宏定义	说明
main.h	__CURRENT_THREE_SHUNT_DYNAMIC	将宏添加到文件中。
	DYNAMIC_CURRENT_SHUNT_CONFIG_EN	将宏添加到文件并将其设置为 TRUE。

8.3.2.2.3 双分流器配置

下表展示了双分流器配置的宏。

表 8-8. 用于三分流器检测的宏

文件	宏定义	说明
main.h	__CURRENT_TWO_SHUNT_A_B	如果只在 A 相和 B 相有两个分流检测通道可用于进行电流采样，并且 A 路由到 ADC0，B 路由到 ADC1，则选择此配置。
	__CURRENT_TWO_SHUNT_B_C	如果只在 B 相和 C 相上有两个分流检测通道可用于进行电流采样，并且 B 路由到 ADC0，C 路由到 ADC1，则选择此配置。
	__CURRENT_TWO_SHUNT_A_C	如果只在 A 相和 B 相上有两个分流检测通道可用于进行电流采样，并且 A 路由到 ADC0，B 路由到 ADC1，则选择此配置。

如果用户将相位组合连接到 ADC 0 和 ADC1 实例的方式与 SDK 中的默认连接不同。需要进行适当的更改，请参阅节 8.3.2.4.1.2。

8.3.2.2.4 单分流器配置

使用 **__CURRENT_SINGLE_SHUNT** 进行单分流器配置。

单分流器配置需要用户添加额外的 TIMA1 实例，启用 TIMA0 的交叉触发功能。建议使用单分流器配置从 SDK 示例工程移植配置（如表 8-1 所示），并按照节 8.3.2.4.1.3 中的说明进行操作。

8.3.2.3 CSA 偏移比例因子

FOC 应用根据可通过 ADC 检测到的最大电流通过 ADC 将采样电流转换为 PU 系统值。这取决于从放大器引入的 CSA 偏移。

通常，对于双极性电流检测测量，满量程值 ADC 3.3V / 2 = 1.65V 作为偏移给出。对于电流检测始终为单极的应用，将偏移值设置为小于 0.5V，以便使用最大满量程 ADC 输出来测量正电流，并留出较小的裕度来测量负电流。

为实现适当的功能，FOC 应用要求指定此标度。当 ADC 12 位值转换为 PU 值时，如果偏移设置为 0：然后，将比例因子设置为 _IQ(1)。如果对于双极电流检测测量，硬件中的 CSA 偏移设置为 1.65V (3.3V/2)，则比例因子设置为 _IQ(2)。对于任意偏移值，比例值应根据方程式 47 指定：

$$\_IQ(CSA\_OFFSET[PU]) = \_IQ(3.3V/(3.3V - CSA\_OFFSET[V])) \quad (47)$$

FOC 应用为用户提供了一个宏，可更改为单分流器电流检测方法定义的预设 CSA 偏移，如下表所示。

表 8-9. 处理 CSA 偏移的宏

文件	宏定义	说明
Drv8329_focHalInterface.c	DRV8329_CURRENT_SF_IQ	定义单分流器电流检测方法的 CSA 偏移比例因子。

对于双或三分流器电流检测方法，FOC 应用通过应用格式转换 API (_IQ11toIQ)，在 ***focH ALInterface.c*** 文件中将比例因子硬编码为 _IQ(2)。

8.3.2.4 通道映射

表 8-10 展示了 **sensorless-foc_DRV8316_LP_MSPM0G3507** 工程的默认映射关系。

表 8-10. ADC 映射

板层		HAL 层		SysConfig 配置
	SysConfig 层			
电路板连接	**ADC 通道数**	**ADC 存储器**	**HAL 层宏**	
A 相电流	A0_3	A0_MEM0	ADC0_CURRENT_U_CH	初始化 ADC0/ADC1 的存储器 0/1。通道选择无关紧要，因为 FOC 应用会在运行时根据 HAL 层的宏来动态覆盖通道设置。
B 相电流 1	A0_2	A0_MEM1	ADC0_CURRENT_V_CH	
B 相电流 2	A1_2	A1_MEM0	ADC1_CURRENT_V_CH	
C 相电流	A1_1	A1_MEM0	ADC1_CURRENT_W_CH	
总线电压	A1_3	A1_MEM2	FOC_ADC_VOLT_DC_INST ADC_VOLT_DC_IDX	将 ADC1 通道 3 分配至存储器 2。
A 相电压	A1_6	A1_MEM0	ADC_VOLTAGE_U_INST ADC_VOLTAGE_U_IDX ADC_VOLTAGE_U_CH	如果运行 ISD 函数，则不会在 SysConfig 中初始化，而在 FOC 应用中动态初始化。
B 相电压	A0_7	A0_MEM0	ADC_VOLTAGE_V_INST ADC_VOLTAGE_V_IDX ADC_VOLTAGE_V_CH	
C 相电压	A1_5	A1_MEM1	ADC_VOLTAGE_W_INST ADC_VOLTAGE_W_IDX ADC_VOLTAGE_W_CH	
ADC 中断	不适用	A0_MEM1	FOC_ADC_ISR_INST	利用加载的 MEM1 结果初始化 ADC0 中断配置。
			FOC_ADC_MEM_RES_LOAD	

FOC 应用会针对不同模式动态修改 ADC 通道和 ADC 存储器结果寄存器映射。在 ISD 模式下，FOC 应用使用 ADC0 MEM0 和 ADC1 MEM0/1 存储采样的相电压。在其他模式下，FOC 应用使用 ADC0 MEM0/1 和 ADC1 MEM0/1 存储采样的相电流。表 8-11 展示了 FOC 应用使用的 ***Drv8316_focHalInterface.c*** 文件中的 API。

表 8-11. 用于修改 ADC 映射的 API

FOC API	HAL 层宏	说明
HAL_GD_ConfigureVoltageChannels()	ADC_VOLTAGE_X_INST ADC_VOLTAGE_X_IDX ADC_VOLTAGE_X_CH	将相电压通道 (X_CH) 映射到 ADC 实例 (X_INST) 的特定 ADC MEM (X_IDX)。 X = U / V / W。

表 8-11. 用于修改 ADC 映射的 API（续）

FOC API	HAL 层宏	说明
HAL_GD_ReadVoltages()	ADC_VOLTAGE_U_INST ADC_VOLTAGE_U_IDX	从 ADC 实例 (X_INST) 的 ADC MEM (X_IDX) 读取相电压。 X = U / V / W。
HAL_GD_ConfigureCurrentChannels()	ADC0_CURRENT_X_CH	将相电流通道 (X_CH) 映射到**硬编码的** ADC 实例和 ADC 存储器。 X = U / V / W。
HAL_GD_ReadCurrents()	不适用	从 ADC 实例的硬编码 ADC 存储器读取相电流。
HAL_GD_ReadDCVBusVoltage()	FOC_ADC_VOLT_DC_INST ADC_VOLT_DC_IDX	从 ADC 实例的预定义 ADC 存储器读取总线电压。

以下章节将介绍用户硬件不遵循 SDK 工程默认映射时的流程。

8.3.2.4.1 相电流通道

相电流通道的可修改宏是 ADC 通道，其余配置（ADC 实例和 ADC 索引）均在 ***HAL_GD_ConfigureCurrentChannels()*** API 中进行硬编码，如图 8-16 所示。

```
168 void HAL_GD_ConfigureCurrentChannels(CURRENT_SHUNT_TYPES currentShunt)
169 {
170     /* Configure the ADC channels based on the ADC pin configurations */
171     switch(currentShunt)
172     {
173           case CURRENT_THREE_SHUNT_DYNAMIC:
174           case CURRENT_THREE_SHUNT_AB_C:
175               HAL_setADCIdxChannel(FOC_CURR_ADC0_INST, ADC0_IDX_0, ADC0_CURRENT_U_CH);
176               HAL_setADCIdxChannel(FOC_CURR_ADC0_INST, ADC0_IDX_1, ADC0_CURRENT_V_CH);
177               HAL_setADCIdxChannel(FOC_CURR_ADC1_INST, ADC1_IDX_0, ADC1_CURRENT_W_CH);
178               break;
179           case CURRENT_THREE_SHUNT_A_BC:
180               HAL_setADCIdxChannel(FOC_CURR_ADC0_INST, ADC0_IDX_0, ADC0_CURRENT_U_CH);
181               HAL_setADCIdxChannel(FOC_CURR_ADC1_INST, ADC1_IDX_0, ADC1_CURRENT_V_CH);
182               HAL_setADCIdxChannel(FOC_CURR_ADC1_INST, ADC1_IDX_1, ADC1_CURRENT_W_CH);
183               break;
184
185           case CURRENT_TWO_SHUNT_A_B:
186               HAL_setADCIdxChannel(FOC_CURR_ADC0_INST, ADC0_IDX_0, ADC0_CURRENT_U_CH);
187               HAL_setADCIdxChannel(FOC_CURR_ADC1_INST, ADC1_IDX_0, ADC1_CURRENT_V_CH);
188               break;
189
190           case CURRENT_TWO_SHUNT_A_C:
191               HAL_setADCIdxChannel(FOC_CURR_ADC0_INST, ADC0_IDX_0, ADC0_CURRENT_U_CH);
192               HAL_setADCIdxChannel(FOC_CURR_ADC1_INST, ADC1_IDX_0, ADC1_CURRENT_W_CH);
193               break;
194
195           case CURRENT_TWO_SHUNT_B_C:
196               HAL_setADCIdxChannel(FOC_CURR_ADC0_INST, ADC0_IDX_0, ADC0_CURRENT_V_CH);
197               HAL_setADCIdxChannel(FOC_CURR_ADC1_INST, ADC1_IDX_0, ADC1_CURRENT_W_CH);
198               break;
199
200           case CURRENT_SINGLE_SHUNT:
201               break;
202           default:
203               break;
204     }
205 }
```

With different current shunt configurations, foc application set the ADC channels accordingly

图 8-16. HAL_GD_ConfigureCurrentChannels 的源代码

图 8-17 展示了相电流的 ADC 通道宏（在 gateDriver.h 文件中）。

```
680 /*! @brief Instance for ADC Interrupt */
681 #define FOC_ADC_ISR_INST          (ADC0_INST)
682 /*! @brief Memory Load Register for ADC Interrupt */
683 #define FOC_ADC_MEM_RES_LOAD      DL_ADC12_IIDX_MEM1_RESULT_LOADED
684
685 /*! @brief ADC instance for DC bus voltage */
686 #define FOC_ADC_VOLT_DC_INST      (ADC0_INST)
687 /*! @brief IDX for the DC bus voltage */
688 #define ADC_VOLT_DC_IDX           (ADC0_ADCMEM_3)
689
690 /*! @brief Channel for Phase U current */
691 #define ADC0_CURRENT_U_CH         (DL_ADC12_INPUT_CHAN_3)
692 /*! @brief Channel for Phase V current */
693 #define ADC0_CURRENT_V_CH         (DL_ADC12_INPUT_CHAN_1)
694 /*! @brief Channel for Phase V current */
695 #define ADC1_CURRENT_V_CH         (DL_ADC12_INPUT_CHAN_2)
696 /*! @brief Channel for Phase W current */
697 #define ADC1_CURRENT_W_CH         (DL_ADC12_INPUT_CHAN_1)
```

Define the ADC interrupt trigger source. Use the last ADC MEM for phase current as the trigger source

Define the ADC channel used by users' hardware circuit. Modify the CHAN accordingly.

图 8-17. 相电流 ADC 通道宏

根据硬件设计的电流检测方法，用户应覆盖 ***gateDriver.h*** 文件中 ***HAL_GD_ConfigureCurrentChannels()*** API 的源代码和相电流的 ADC 通道宏，以便正确地对相电流进行采样。

8.3.2.4.1.1 三分流器配置

图 8-18 展示了三或双分流器配置信号链。建议使用两个 ADC 实例来对 MSPM0G 系列的三相电流进行采样和转换。

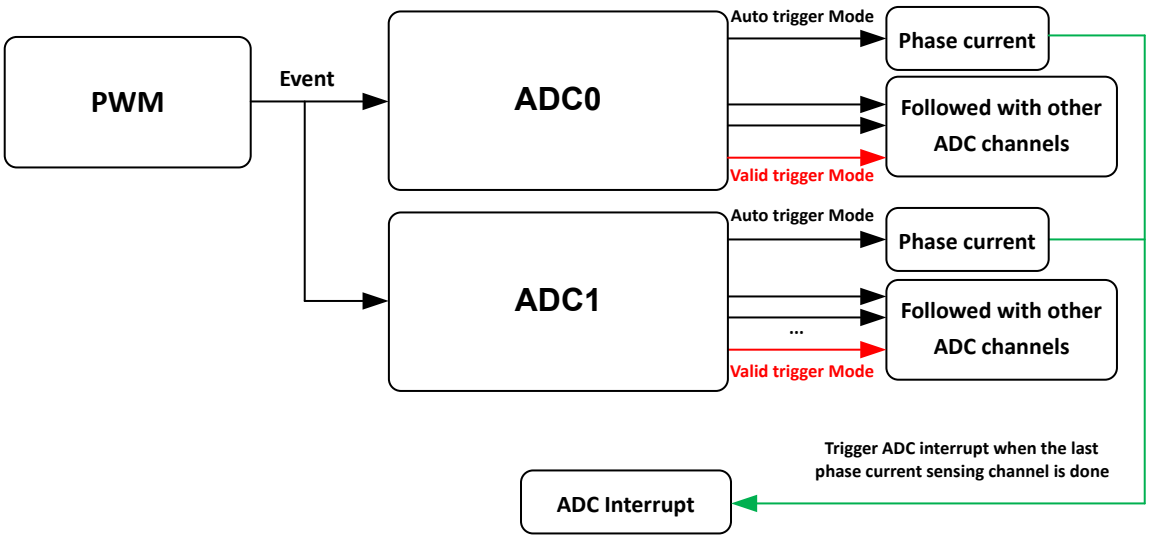


图 8-18. 三或双分流器配置中的外设连接

按照以下步骤，根据硬件电路修改相电流通道。

1. 将相应的宏定义（节 8.3.2.2）添加到 main.h 文件中
2. 根据硬件设计修改 ***gateDriver.h*** 文件中的 ADC 通道
3. 如果未启用同步采样，用户可以将 ADC0_CURRENT_V_CH 和 ADC1_CURRENT_V_CH 设置为相同
4. 选择用于相电流采样的最后一个 ADC MEM 索引作为中断触发源。例如，如果用户将所有三个相电流均设置为 ADC0，则 FOC 应用使用三个 ADC MEM 索引（MEM0、MEM1、MEM2），因此 FOC_ADC_MEM_RES_LOAD 应设置为 ADC MEM2
5. 如果为 ADC1 实例使用了 ADC 中断，请将宏 FOC_ISR_ADC1 添加到 ***gateDriver.h*** 文件中

6. 相应地初始化 ADC 存储器并在 SysConfig 中设置 ADC 中断源。ADC 通道设置是可选的，因为 FOC 应用会在运行时调用 HAL_GD_ ConfigureCurrentChannels() 来重新配置它
7. 覆盖 HAL_GD_ConfigureCurrentChannels() API 中的源代码，以满足第 2~6 步中实现的 ADC 实例和 ADC 存储器索引配置要求，因为 ADC 实例设置和存储器读取选择已硬编码
8. 覆盖 HAL_GD_HAL_GD_ReadCurrents() API 中的源代码，以满足第 7 步中实现的 ADC 实例和 ADC 存储器索引配置要求

图 8-19 展示了修改与 ADC1 的所有相电流通道映射的示例。ADC 通道设置示例如下：

- U 相电流通道： ADC1.1 -> ADC MEM0
- V 相电流通道：ADC1.2 -> ADC MEM1
- U 相电流通道：ADC1.3 -> ADC MEM2 -> 触发 ADC1 中断

备注

仅用作示例。不建议将一个 ADC 实例用于全部三个电流采样。

Step 1: Add the current shunt configuration

```
#elif defined CUSTOM

/*! @brief TIDA010250 has inverting isense */
#define _INVERT_ISEN
/*! @brief IPD feature is enabled */
#define __IPD_ENABLE
/*! @brief TIDA010250 propagation delay */
#define DRIVER_PROPAGATION_DELAY_nS                500
/*! @brief TIDA010250 minimum on time (rise time + settling time) */
#define DRIVER_MIN_ON_TIME_nS                      8000
/*! @brief DC voltage base value */
#define DC_VOLTAGE_BASE                             448.0
/*! @brief Full scale readable current used as current base value,
calculated using (FULL Scale Voltage(3.3)/2* CSA Gain) */
#define FULL_SCALE_CURRENT_BASE                     8.25
/*! @brief Current shunt configuration */
#define __CURRENT_THREE_SHUNT_DYNAMIC
/*! @brief Enable dynamic current shunt changing */
#define DYNAMIC_CURRENT_SHUNT_CONFIG_EN             FALSE//TRUE

#endif
```

Step 2 & 3: Set ADC Current Channel

```
/*! @brief Channel for Phase U current */
#define ADC0_CURRENT_U_CH         (DL_ADC12_INPUT_CHAN_1)
/*! @brief Channel for Phase V current */
#define ADC0_CURRENT_V_CH         (DL_ADC12_INPUT_CHAN_2)
/*! @brief Channel for Phase V current */
#define ADC1_CURRENT_V_CH         (DL_ADC12_INPUT_CHAN_2)
/*! @brief Channel for Phase W current */
#define ADC1_CURRENT_W_CH         (DL_ADC12_INPUT_CHAN_3)
```

This example uses all ADC1 channels.
And simultaneously sampling is not enabled.

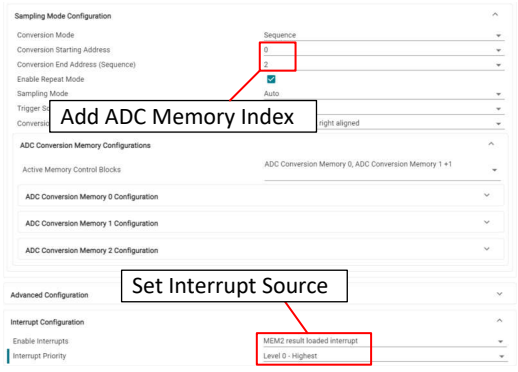
Step 4 & 5: Set ADC Interrupt Source

```
/*! @brief Instance for ADC Interrupt */
#define FOC_ADC_ISR_INST          (ADC1_INST)
/*! @brief Memory Load Register for ADC Interrupt */
#define FOC_ADC_MEM_RES_LOAD       DL_ADC12_IIDX_MEM2_RESULT_LOADED

#define FOC_ISR_ADC1
```

This example uses all ADC1 channels, so the ADC MEM is increased to 2 (0,1,2)
Due to ADC1 is used with larger Memory Index for phase current smapling, there add FOC_ISR_ADC1 macro for FOC application

Step 6: Configure SysConfig



Step 7: HAL_GD_ConfigureCurrentChannels()

```
void HAL_GD_ConfigureCurrentChannels(CURRENT_SHUNT_TYPES currentShunt)
{
    /* Configure the ADC channels based on the ADC pin configurations */
    switch(currentShunt)
    {
        case CURRENT_THREE_SHUNT_DYNAMIC:
        case CURRENT_THREE_SHUNT_AB_C:
            HAL_setADCIdxChannel(FOC_CURR_ADC1_INST, ADC0_IDX_0, ADC0_CURRENT_U_CH);
            HAL_setADCIdxChannel(FOC_CURR_ADC1_INST, ADC0_IDX_1, ADC0_CURRENT_V_CH);
            HAL_setADCIdxChannel(FOC_CURR_ADC1_INST, ADC1_IDX_2, ADC1_CURRENT_W_CH);
            break;
        case CURRENT_THREE_SHUNT_A_BC:
            HAL_setADCIdxChannel(FOC_CURR_ADC1_INST, ADC0_IDX_0, ADC0_CURRENT_U_CH);
            HAL_setADCIdxChannel(FOC_CURR_ADC1_INST, ADC1_IDX_1, ADC1_CURRENT_V_CH);
            HAL_setADCIdxChannel(FOC_CURR_ADC1_INST, ADC1_IDX_2, ADC1_CURRENT_W_CH);
            break;

        case CURRENT_TWO_SHUNT_A_B:
            HAL_setADCIdxChannel(FOC_CURR_ADC0_INST, ADC0_IDX_0, ADC0_CURRENT_U_CH);
            HAL_setADCIdxChannel(FOC_CURR_ADC1_INST, ADC1_IDX_0, ADC1_CURRENT_V_CH);
            break;

        case CURRENT_TWO_SHUNT_A_C:
            HAL_setADCIdxChannel(FOC_CURR_ADC0_INST, ADC0_IDX_0, ADC0_CURRENT_U_CH);
            HAL_setADCIdxChannel(FOC_CURR_ADC1_INST, ADC1_IDX_0, ADC1_CURRENT_W_CH);
            break;

        case CURRENT_TWO_SHUNT_B_C:
            HAL_setADCIdxChannel(FOC_CURR_ADC0_INST, ADC0_IDX_0, ADC0_CURRENT_V_CH);
            HAL_setADCIdxChannel(FOC_CURR_ADC1_INST, ADC1_IDX_0, ADC1_CURRENT_W_CH);
            break;

        case CURRENT_SINGLE_SHUNT:
            break;
```

This example uses all ADC1 channels, so the ADC MEM is increased to 2 (0,1,2). The ADCX_INST and ADC_IDX should be both overwritten according to the hardware circuit. Users only requires to modify the used THREE_SHUNT_XX related branch. Here modify both and is optional.

Step 8: HAL_GD_ReadCurrents()

```
void HAL_GD_ReadCurrents(HAL_MEASURE_CURRENT_T *pCurrent)
{
    int32_t adc0Idx0,adc0Idx1,adc1Idx0,adc1Idx1;

    MC_ABC_T* iabcRaw = &pCurrent->iabcRaw;

    CURRENT_SHUNT_TYPES currentShunt = pCurrent->currentShunt;

    //ADC1-MEM0
    adc0Idx0 = DL_ADC12_getMemResult(FOC_CURR_ADC1_INST, FOC_CURR_ADC0_IDX_0);
    //ADC1-MEM1
    adc0Idx1 = DL_ADC12_getMemResult(FOC_CURR_ADC1_INST, FOC_CURR_ADC0_IDX_1);
    adc1Idx0 = DL_ADC12_getMemResult(FOC_CURR_ADC1_INST, FOC_CURR_ADC1_IDX_1);
    //ADC1-MEM2
    adc1Idx1 = DL_ADC12_getMemResult(FOC_CURR_ADC1_INST, FOC_CURR_ADC1_IDX_2);

    switch(pCurrent->currentShunt)
    {
        case CURRENT_THREE_SHUNT_DYNAMIC:
        case CURRENT_THREE_SHUNT_AB_C:
            iabcRaw->a = adc0Idx0;   //ADC1-MEM0
            iabcRaw->b = adc0Idx1;   //ADC1-MEM1
            iabcRaw->c = adc1Idx1;   //ADC1-MEM2
        break;

        case CURRENT_THREE_SHUNT_A_BC:
            iabcRaw->a = adc0Idx0;   //ADC1-MEM0
            iabcRaw->b = adc1Idx0;   //ADC1-MEM1
            iabcRaw->c = adc1Idx1;   //ADC1-MEM2
        break;

        case CURRENT_TWO_SHUNT_A_B:
            iabcRaw->a = adc0Idx0;
            iabcRaw->b = adc1Idx0;
        break;

        case CURRENT_TWO_SHUNT_A_C:
            break;
        default:
```

This example uses all ADC1 channels, so ADC MEM read back code should be modified according to Step 7.
The Phase U is read from ADC1 – MEM0 (iabcRaw->a)
The Phase V is read from ADC1 – MEM1 (iabcRaw->b)
The Phase W is read from ADC1 – MEM0 (iabcRaw->c)

图 8-19. 在三分流器配置中修改 ADC 电流通道

8.3.2.4.1.2 双分流器配置

双分流配置请参考三分流配置流程图（节 8.3.2.2.1），细微区别如下：

- 对于第 1 步，用户可从可用的 CURRENT_TWO_SHUNT_X_Y（X、Y = A、B、C）中选择双分流器配置宏。

- 对于第 7 和 8 步，只需覆盖使用的 CURRENT_TWO_SHUNT_X_Y (节 8.3.2.2.3) 分支，其他分支可以保持不变。

8.3.2.4.1.3 单分流器配置

图 8-20 展示了单分流器配置信号链。

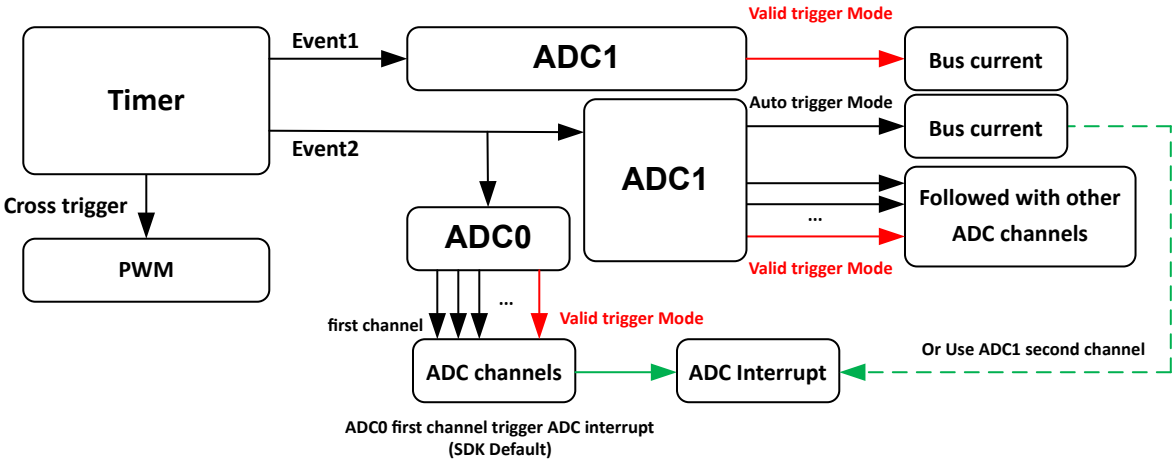


图 8-20. 单分流器配置中的外设连接

按照以下步骤，根据硬件电路修改相电流通道。

1. 将相应的宏定义 __CURRENT_SINGLE_SHUNT 添加到 main.h 文件中。
2. 根据硬件设计修改 gateDriver.h 文件中的 ADC 通道。单分流器需要一个 ADC 通道和两个 ADC MEM 索引，因为它在一个 PWM 周期内进行两次采样。
3. 选择其他 ADC 实例的 ADC MEM0 索引作为中断触发源。例如，如果用户使用 ADC1 进行总线电流采样，则使用 ADC0 实例作为中断触发器。
   a. 如果使用 ADC1 中断，则将宏 FOC_ISR_ADC1 添加到 gateDriver.h 文件中。
4. 相应地初始化 ADC 存储器并在 SysConfig 中设置 ADC 中断源。ADC 通道设置是可选的，因为 FOC 应用会调用 HAL_GD_ ConfigureCurrentChannels() 来重新配置它。
   a. 如果使用 ADC1 中断，则切换 TIMA1 中 ADC0 和 ADC1 的事件触发器。

图 8-21 展示了在单分流器配置中修改 ADC 电流通道映射的示例。

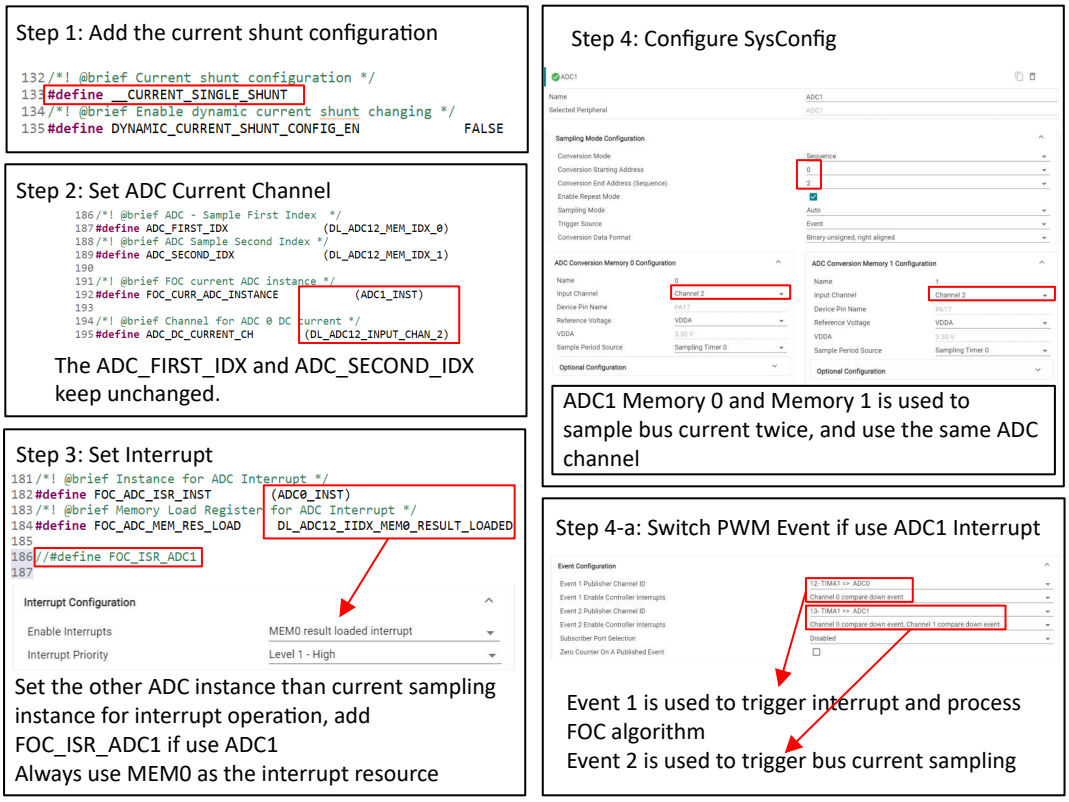


图 8-21. 在单分流器配置中修改 ADC 电流通道

备注

单分流器的迁移不限于一种特定模式。用户应注意中断设置，包括在计时器中设置正确的事件触发器，以及在 ADC 中设置正确的触发模式和中断触发器，以满足图 8-20 中定义的要求。

8.3.2.4.2 总线电压通道

按照以下步骤，根据硬件电路修改总线电压通道。

1. 在 SysConfig 中将总线电压通道分配给所需的 ADC 实例和 ADC 存储器
2. 在 gateDriver.h 文件中相应地覆盖 HAL 层宏

下图展示了一个修改电压通道映射的示例：将总线电压通道（在新硬件电路中定义的 ADC0_4）分配给 ADC0 存储器 3。

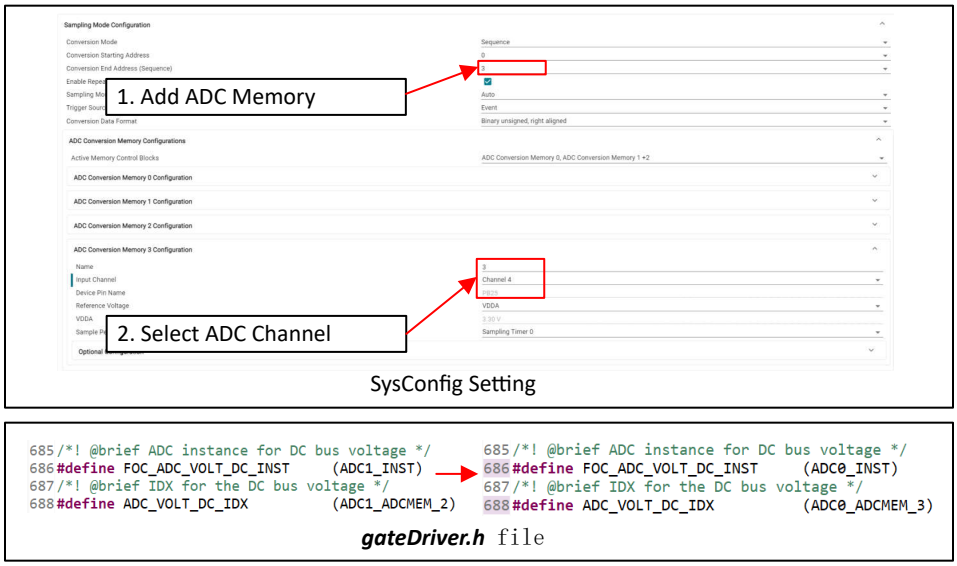


图 8-22. 修改总线电压通道

TI 建议使用 ADC0 或 ADC1 MEM2 进行总线电压采样（在相电流采样后），并保持其他 ADC 通道位于总线电压通道之前。

8.3.2.4.3 相电压通道

按照以下步骤，根据硬件电路修改相电压通道。

1. 确保 SysConfig 已初始化 ADC 存储器索引 (ADCx_ADCMEM_y)。无需设置相应的硬件电压 ADC 通道，因为 FOC 应用会调用 HAL_GD_ConfigureVoltageChannels () 来重新配置它。
2. 根据硬件电路覆盖 gateDriver.h 文件中的 HAL 层宏。

图 8-23 展示了修改相电压通道映射的示例。

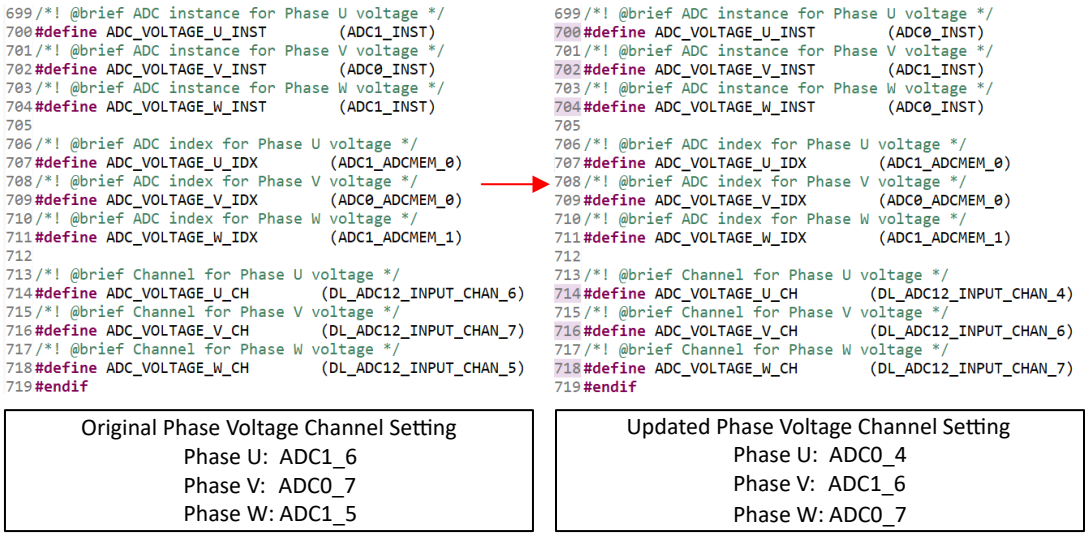


图 8-23. 修改相电压通道

除非所有三个索引都加载到同一 ADC 实例，否则 ADC 存储器索引可保持不变。

备注

如果未应用初始速度检测 (ISD) 功能，则不会在 FOC 应用中使用相电压通道。

8.3.2.5 触发模式

FOC 应用通过 PWM 模块生成的硬件事件来启动 ADC 转换，以避免软件延迟。ADC 重复模式与有效触发模式一起使用，可在每个 PWM 周期实现定期转换。下表介绍了自动步进和有效触发模式之间的差异。

表 8-12. ADC 触发模式行为

ADC 触发模式	说明
自动步进到下一次转换	当 ADC 完成当前存储器转换时，它会自动步进到下一个存储器并开始转换。
有效触发到下一次转换	当 ADC 完成当前存储器转换时，它会等待另一个触发信号，以在下一个存储器中开始转换。如果没有发生触发信号，ADC 会保持在当前 ADC 存储器中且不进行操作。

按照以下步骤为 FOC 应用设置 ADC 通道的合适触发模式。

8.3.2.5.1 三或双分流器配置

对于三或双分流器配置，FOC 应用可在一个 PWM 周期内转换一次相电流。用于相电流采样的 ADC 实例必须且仅将最后一个 ADC 存储器索引设置为有效触发模式。这样，在每个 PWM 周期，当低侧 MOSFET 全部导通时，PWM 模块会向 ADC 模块生成一个事件，触发 ADC 对相电流进行采样和转换。下图展示了 ***sensorless-foc_DRV8316*** 工程的 SysConfig 设置。

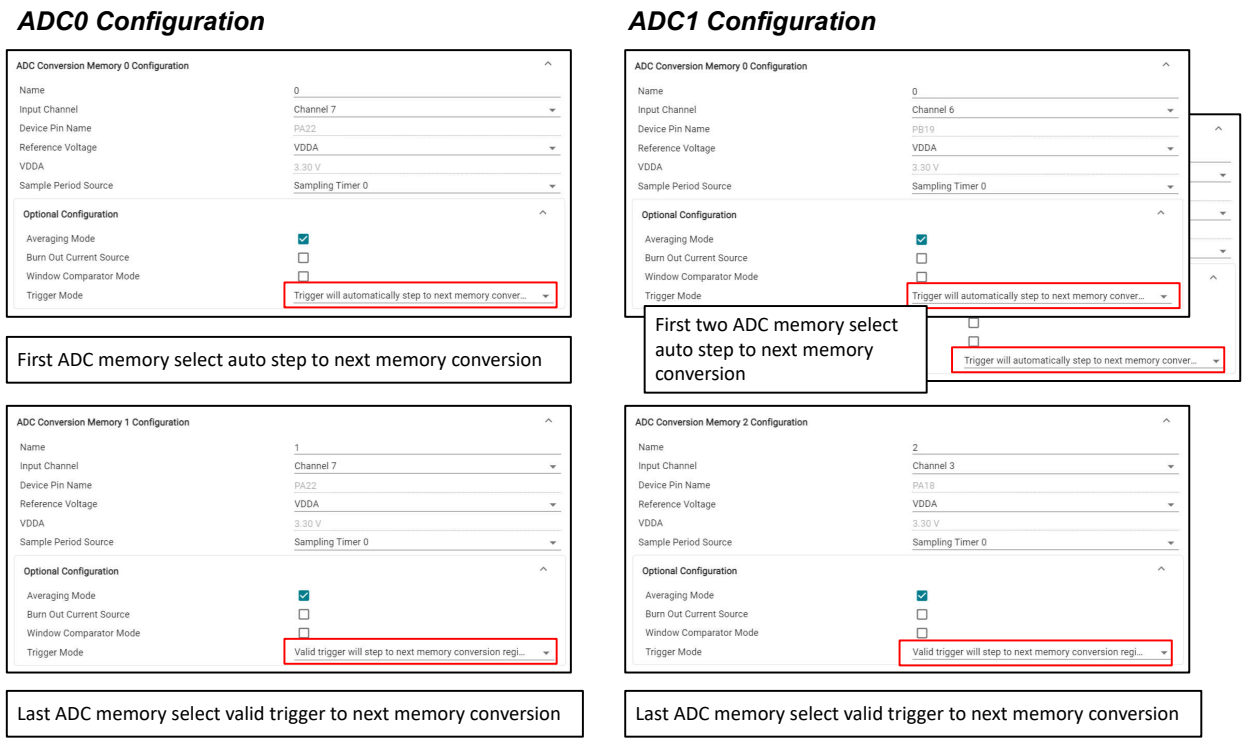


图 8-24. 三或双分流器配置中的 ADC 触发模式设置

8.3.2.5.2 单分流器配置

对于单分流器配置，FOC 应用在一个 PWM 周期内转换相电流两次，并在第二次时触发中断。用于相电流采样的 ADC 实例必须且仅将**第一个**和**最后一个** ADC 存储器索引设置为有效触发模式。因此，在每个 PWM 周期中，PWM 模块都会根据空间矢量状态向 ADC 模块生成两个事件，并且这些事件会触发 ADC 按顺序采样和转换总线电流。下图展示了 ***sensorless-foc_DRV8329*** 工程的 SysConfig 设置。

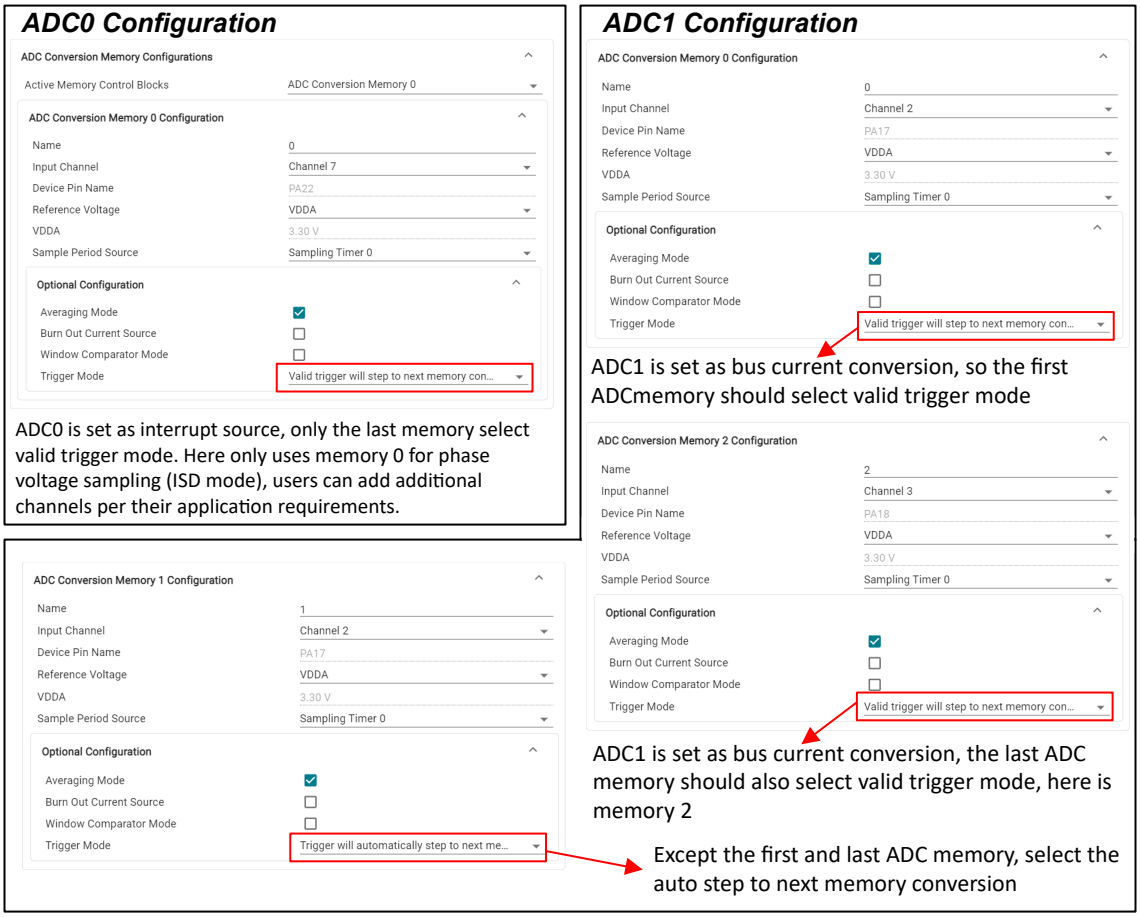


图 8-25. 单分流器配置中的 ADC 触发模式设置

备注

如果使用 ISD 功能，应为不对总线电流进行采样的 ADC 实例配置两个相电压通道；否则无法同时对全部三个相电压进行采样。

8.3.3 GPIO 引脚

FOC 工程中使用的 GPIO 主要包括以下部分：

- HALL_GPIO_IN：霍尔输入信号，详见节 8.3.4
- FOC_GPIO_IN：获取方向和制动命令输入
- FOC_GPIO_OUT：用于控制栅极驱动器电源或指示 FOC 故障
- TST：测试引脚，不适用于 FOC 算法

如果用户要保留 GPIO 功能并分配新 PINMUX，请参阅节 8.3.4。如果用户要删除 GPIO 引脚，可按照以下步骤操作：

1. 在 SysConfig 中删除 GPIO 引脚
2. 删除 ***focHALInterface.c*** 文件中使用相关 GPIO 宏的相关 API 函数，如下图所示。

```
927 void HAL_ClearNFault()
928 {
929 //    DL_GPIO_clearPins(FOC_GPIO_NFAULT_PORT, FOC_GPIO_NFAULT_PIN);
930 }
931
932 void HAL_SetNFault()
933 {
934 //    DL_GPIO_setPins(FOC_GPIO_NFAULT_PORT, FOC_GPIO_NFAULT_PIN);
935 }

1003 _Bool HAL_getDirPinStatus(void)
1004 {
1005     return 0; //DL_GPIO_readPins(FOC_GPIO_DIR_PORT, FOC_GPIO_DIR_PIN)?
1006               //          HAL_GPIO_HIGH : HAL_GPIO_LOW;
1007 }
1008
1009 _Bool HAL_getBrakePinStatus(void)
1010 {
1011     return 0; //DL_GPIO_readPins(FOC_GPIO_BRAKE_PORT, FOC_GPIO_BRAKE_PIN)?
1012               //          HAL_GPIO_HIGH : HAL_GPIO_LOW;
1013 }
```

图 8-26. 删除预设 GPIO 功能

8.3.4 HALL 模块

仅在带传感器 FOC 中，三个霍尔传感器输入信号通过通用输入/输出 (GPIO) 输入馈送，并根据 GPIO 电平变化生成事件，发送到特定计时器（采集模式）。根据您的硬件设计，使用 SysConfig 覆盖霍尔引脚。

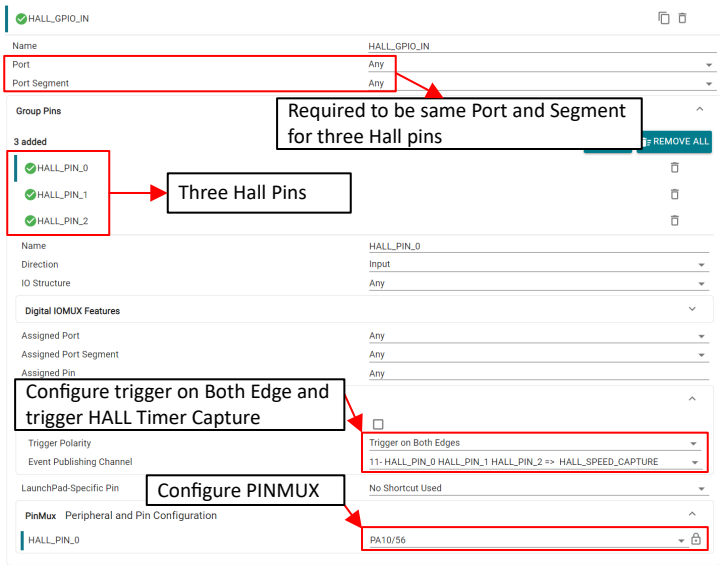


图 8-27. 修改霍尔引脚

8.3.5 UART 模块

FOC 工程使用 UART 与 GUI 通信，支持在没有 CCS IDE 的情况下进行电机调优。使用 DMA 提高 UART 传输效率，而且不受 FOC ISR 影响。

按照以下步骤移除 UART 模块：

1. 删除 SysConfig 中的 UART 模块和 CRC 模块
2. 从编译中排除 uart_comm.c 文件
3. 移除 main.c 和 ISR.c 文件中使用了相关 UART 或 DMA 宏的 API 函数

图 8-28 展示了移除 UART 模块的示例。然后，用户可以添加回 UART 模块并应用自己的通信协议。

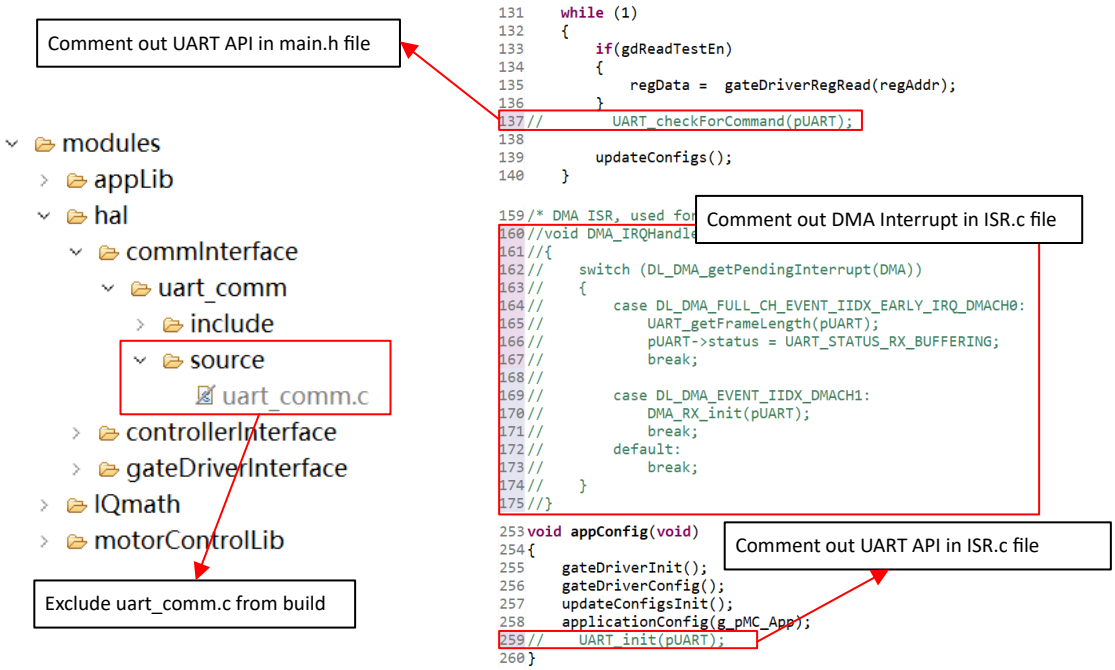


图 8-28. 移除 UART 模块

8.3.6 DAC12 模块

默认情况下，FOC 应用支持 DAC12 模块实时跟踪在 MCU 引脚上运行的变量。按照以下步骤移除 DAC12 模块：

1. 删除 SysConfig 中的 DAC12 模块。
2. 注释掉 FOC_ADC_ISR 中生成 DAC 输出的代码（位于 ***ISR.c*** 文件中）。

```
//    if(pUserCtrlRegs->dacCtrl.dacEn != 0)
//    {
//        if(pUserCtrlRegs->dacCtrl.dacScalingFactor != 0)
//        {
//            dacWriteData = _IQmpy_mathacl(
//                        *((uint32_t *)pUserCtrlRegs->dacCtrl.dacOutAddr),
//                         pUserCtrlRegs->dacCtrl.dacScalingFactor);
//            /* Adding offset value to half of reference voltage */
//            dacWriteData = _IQtoIQ12((dacWriteData >> 1) + _IQ(0.5));
//        }
//        else
//        {
//            dacWriteData = *((uint32_t *)pUserCtrlRegs->dacCtrl.dacOutAddr);
//            if((pUserCtrlRegs->dacCtrl.dacShift)>=0)
//            {
//                dacWriteData <<= pUserCtrlRegs->dacCtrl.dacShift;
//            }
//            else
//            {
//                dacWriteData >>= (-1*pUserCtrlRegs->dacCtrl.dacShift);
//            }
//        }
//        DL_DAC12_output12(DAC0, dacWriteData);
//    }
}
```

图 8-29. 移除 DAC12 模块

8.3.7 IPD 模块（捕获计时器）

在捕获模式下会启用计时器，用于 IPD 脉冲时间计数。如果不使用 IPD 功能，请按照以下步骤删除 CAPTURE 模块。

1. 删除 SysConfig 中的 CAPTURE 模块。

2. 将 main.h 文件中的宏 __IPD_ENABLE 注释掉。
3. 注释掉 IPD_ADC_init () API（位于 focHALInterface.c 文件中）。

main.h

```
159 /*! @brief DRV8316 has inverting isense */
160 #define _INVERT_ISEN
161 /*! @brief IPD feature is enabled */
162 //#define __IPD_ENABLE
163 /*! @brief DRV8323 propagation delay */
164 #define DRIVER_PROPAGATION_DELAY_nS                100
165 /*! @brief DRV8316 minimum on time (rise time + settling time) */
```

focHALInterface.c

```
1021 static void IPD_ADC_init(ADC12_Regs *adc12Inst, uint32_t chansel)
1022 {
1023 //    DL_ADC12_setClockConfig(adc12Inst, (DL_ADC12_ClockConfig *) &gADC_IPDClockConfig);
1024 //    DL_ADC12_initSingleSample(adc12Inst,
1025 //        DL_ADC12_REPEAT_MODE_ENABLED, DL_ADC12_SAMPLING_SOURCE_AUTO, DL_ADC12_TRIG_SRC_SOFTWARE,
1026 //        DL_ADC12_SAMP_CONV_RES_12_BIT, DL_ADC12_SAMP_CONV_DATA_FORMAT_UNSIGNED);
1027 //    DL_ADC12_configConversionMem(adc12Inst, DL_ADC12_MEM_IDX_0,
1028 //        chansel, DL_ADC12_REFERENCE_VOLTAGE_VDDA, DL_ADC12_SAMPLE_TIMER_SOURCE_SCOMP0, DL_ADC12_AVER
1029 //        DL_ADC12_BURN_OUT_SOURCE_DISABLED, DL_ADC12_TRIGGER_MODE_AUTO_NEXT, DL_ADC12_WINDOWS_COMP_MO
1030 //    DL_ADC12_setPowerDownMode(adc12Inst,DL_ADC12_POWER_DOWN_MODE_MANUAL);
1031 //    DL_ADC12_setSampleTime0(adc12Inst,2);
1032 //    DL_ADC12_setPublisherChanID(adc12Inst,FOC_IPD_EVENT_CH);
1033 //#ifdef _NONINVERT_ISEN
1034 //    DL_ADC12_enableEvent(adc12Inst,DL_ADC12_EVENT_WINDOW_COMP_HIGH);
1035 //#else
1036 //    DL_ADC12_enableEvent(adc12Inst,DL_ADC12_EVENT_WINDOW_COMP_LOW);
1037 //#endif
1038 //    DL_ADC12_enableConversions(adc12Inst);
1039 }
```

图 8-30. 移除 IPD 模块

8.4 自定义板验证

成功迁移 FOC 工程以匹配您的硬件配置后，请按照以下步骤验证软件功能。

ADC 中断验证

ADC 中断是 FOC 应用的关键任务，负责读取电机相电流并执行 FOC 算法。用户可以在 FOC_ADC_ISR() 中设置一个断点（或 GPIO 切换），以验证 ADC 中断触发频率是否与预期 FOC 环路频率一致。

在空闲状态下，ADC 存储器索引寄存器会定期更新转换结果，以反映硬件电路的基线状态。这些值可在电机运行之前提供适当的参考读数。例如，向 ADC 相电流输入通道施加 1.65V 电压，会得到一个大约为 2,047 的 ADC 存储器索引寄存器值。

如果未触发 ADC 中断，请检查以下配置：

- 如果使用 ADC1 中断，则添加 FOC_ISR_ADC1 宏
- 验证是否为 ADC 转换设置了合适的 PWM/计时器触发事件
- 验证是否为 ADC 模块设置了合适的 ADC 中断触发事件

如果在意外的频率下发生 ADC 中断，请检查以下配置：

- 验证是否为每个 ADC 存储器索引设置了合适的 ADC 触发模式
- 验证是否为 ADC 模块设置了合适的 ADC 中断触发事件
- 无传感器/通用 FOC 支持最大 10kHz 的 ADC 中断，而带传感器 FOC 支持最大 16kHz 的 ADC 中断

如果 ADC 转换值异常，请检查以下配置：

- 验证电机的 A/B/C 相电流输入信号（到 ADC 通道）是否正确
- 验证是否为 ADC 转换设置了合适的 PWM/计时器触发事件

PWM 输出与相电流输入映射

FOC 应用使用 U/V/W 分别表示电机的 A/B/C 相。尽管对维持默认关系没有限制，但用户必须确保 PWM 输出与相电流输入之间正确对齐。

例如，如果硬件电路分配了：

- FOC_PWMA0_U_IDX -> 电机 B 相
- FOC_PWMA0_V_IDX -> 电机 A 相

那么您必须相应地配置电机相电流：

- 电机 B 相 ADC 输入通道 -> ADCx_CURRENT_U_CH
- 电机 A 相 ADC 输入通道 -> ADCx_CURRENT_V_CH

这样可以确保 PWM 与电流之间的关系保持一致。如果处理有误，则在运行 FOC 应用时，电流闭环功能将会失败。

栅极驱动器输出验证

按照以下步骤验证栅极驱动器输出：

- 断开电机
- 使用示波器观测三相 PWM 输出
- 将 FOC 应用配置为使用 PWM 环路在强制对齐模式下运行（节 7.7.1.1.1.2）
- 将 FORCE_VQ_CURRENT_LOOP_DIS 或 FORCE_VQ_CURRENT_LOOP_DIS 设置为非零值
- 将 speedInput 设置为非零值以启动 FOC 状态机
- 验证三相 PWM 输出是否遵循 SVPWM 模式（图 7-19）
- 验证三个 PWM 输出是否遵循与电机相位对应的定义序列

电流检测电路验证

按照以下步骤验证电机相电流检测电路：

- 连接电机
- 配置直流电压源输出并将电流限制设置为适当的值，通常低于电机的额定电流规格
- 使用电流探针监测一个电机相电流，使用电压探针监测 ADC 输入通道上的此电机相电流信号
- 将 FOC 应用配置为使用电流环路在强制对齐模式下运行（节 7.7.1.1.1.1）
- 设置适当的对齐时间和对齐电流参数
- 将 speedInput 设置为非零值以启动 FOC 状态机
- 验证电机相电流是否遵循对齐模式（图 7-17）
- 验证电机相电流幅度是否与对齐模式参数匹配

如果对齐电流验证失败，请检查以下配置：

- 检查电机相位分流器电阻是否适合 ADC 采样
- 检查 ADC 输入通道（按比例缩放后）的电机相电流信号是否与电流探针信号匹配
- 检查 PWM 输出与相电流输入映射
- 检查电流检测类型和电流检测方法
- 根据所用的分流器配置检查 ADC 通道映射
- 检查电流环路 PI 参数（请参阅节 7.3.2）

电流控制环路验证

按照以下步骤验证电流控制环路：

- 验证电流检测电路是否正常工作
- 使用电流探针监测一个电机相电流，使用电压探针监测 ADC 输入通道上的此电机相电流信号
- 将 FOC 应用配置为在强制开环中运行（节 7.7.1.2.2）

- 设置适当的对齐模式参数（通常对于空载电机，将对齐电流设置为 0h）
- 设置适当的开环电流和加速率参数（通常对于空载电机，将开环电流设置为 0h）
- 将切换阈值设置为电机额定速度的 20-30%
- 配置电流控制环路参数，建议从自动计算的参数开始（节 7.3.2）
- 将 speedInput 设置为非零值以启动 FOC 状态机
- 验证电机相电流是否遵循开环模式（图 7-21）且是干净的正弦波
- 验证电机相电流幅度和频率是否与开环参数匹配

如果开环验证失败，请检查以下配置：

- 检查 ADC 输入通道（按比例缩放后）的电机相电流信号是否与电流探针信号匹配
- 检查电流环路 PI 参数（请参阅节 7.3.2）
- 请参阅跟踪实时变量，监测 ADC 原始数据并验证原始 ADC 数据是否与正弦波形匹配。

备注

通过这些步骤验证硬件功能后，请参阅调优 LVBLDC 电机，开始使用所有可用特性进行全面的电机调优。

9 常见问题解答 (FAQ)

9.1 MSPM0 无法连接

如果 MCU 无法与 DRV8316 通信，DRV8316 FOC 工程会生成 POR 复位，这可能会导致 MCU 处于定期复位场景并断开 SWD 连接。可将 DRV8316 EVM 板连接到具有合适范围的电压源来修复该问题。

如果这么做没有用，或者其他 DRV EVM 板上出现问题，请按照以下步骤进行恢复：

1. 按 S1 按钮 (PA18)，然后对 MSPM0 LaunchPad 重新供电。
2. 将新固件下载到 MSPM0 器件。

9.2 以硬编码形式启动电机

FOC 应用提供了一个状态位 [***pUserCtrlRegs*** ->algoDebugCrtl2.b.updateConfigs]，用于为用户提供配置更新状态。当调优配置由算法更新并且电机未旋转时，该位每 200ms 复位一次。为了确保用户配置在算法中反应出来，用户可以在进行所需的调优配置后设置该位并等待该状态位复位，然后再给出速度命令。

下面展示了参考代码流程：

```
  ... /* Set tuning parameter */
  pUserCtrlRegs->algoDebugCtrl2.b.updateConfigs = 0x1;
  /* After all tuning parameters updated to FOC algorithm, this bit reset to zero */
  while(pUserCtrlRegs->algoDebugCtrl2.b.updateConfigs){
    updateConfigs();
}
  ... /* Set speedInput as non-zero value to spin motor */
```

9.3 减少 1 个用于同步采样的 ADC 引脚

在 FOC 应用中，三分流器电流检测方法中的同步采样特性通过四个 ADC 通道实现。

对于 MSPM0GX50X 系列，有两个唯一的 ADC 通道同时映射到 ADC0 和 ADC1 实例，如下所示。选择其中一个通道用于 B 相电流采样。

CHANNEL[0:7]	SIGNAL NAME[2]		CHANNEL[8:15]	SIGNAL NAME[1] [2]	
	ADC0	ADC1		ADC0	ADC1
0	A0_0	A1_0 / DAC_OUT[4]	8	A1_7[3]	A0_7[3]
1	A0_1	A1_1	9	-	-
2	A0_2	A1_2	10	-	-
3	A0_3	A1_3	11	*Temperature Sensor*	-
4	A0_4	A1_4	12	A0_12	*Temperature Sensor*
5	A0_5	A1_5	13	*OPA0 output*	*OPA1 output*
6	A0_6	A1_6	14	*GPAMP output*	*GPAMP output*
7	A0_7	A1_7	15	*Supply/Battery Monitor*	*Supply/Battery Monitor*

图 9-1. MSPM0GX50X ADC 通道映射表

请参阅器件的特定数据表，确定是否启用了该双映射 ADC 特性。

9.4 调优实时控制参数

将 ***pUserCtrlRegs*** ->algoDebugCrtl2.b.updateSysParams 设置为 1b，可实现每 200ms 动态更新系统参数一次，例如速度/扭矩/电流环路的 PI 增益，从而通过调优达到所需性能。FOC 应用默认启用了参数动态更新方法，以供用户执行调试活动。

9.5 跟踪实时变量

9.5.1 DAC12 输出

可通过 DAC12 模块从 MCU 实时输出 32 位算法变量。通过将 ***pUserCtrlRegs*** ->dacCtrl.dacEn 设置为 1b，即可启用 DAC12 输出。MSPM0 中的 DAC12 为 12 位 DAC 模块，因此需要在输出之前进行比例调节。按照以下步骤配置 DAC12 输出：

1. 对于全局 IQ 格式的变量 (IQ27)，请参阅方程式 48。将 pUserCtrlRegs->dacCtrl.dacScalingFactor 设置为 IQ(1.0)，可使 DAC12 输出在 0V 至 3.3V 之间，表示 IQ(1.0) 至 IQ(-1.0) 之间的数据。要表示超过值 IQ(1.0) 的数据，可使用更高的 dacScalingFactor。例如，要表示 IQ(2.0) 至 IQ(-2.0) 之间的数据，可将 dacScalingFactor 设置为 IQ(0.5)。

$$DAC\_OUTPUT[V]=(VARIABLE\_VALUE[IQ\ format]\times SCALING\_FACTOR+1)\times 1.65V \quad (48)$$

2. 对于任何非 IQ27 的 IQ 格式输出，用户可以将变量左移位或右移位，以便在输出之前将数据置于 12 位范围内。可以通过将 dacScalingFactor 设置为 0 来选择此模式。
   a. 如果变量值小于 12 位值，请将 pUserCtrlRegs->dacCtrl.dacShift 设置为正，DAC 输出遵循方程式 49。
   b. 如果变量值大于 12 位值，请将 dacShift 设置为负，DAC 输出遵循方程式 50。

$$DAC\_OUTPUT[V]=(VARIABLE\_VALUE[IQ\ format]<<DAC\_SHIFT)\times 3.3V \quad (49)$$

$$DAC\_OUTPUT[V]=(VARIABLE\_VALUE[IQ\ format]>>DAC\_SHIFT)\times 3.3V \quad (50)$$

备注

通过将 dacEn 设置为 1b，可将变量输出馈送到 DAC 寄存器，但用户需要在 TI SysConfig 中启用 DAC 外设，DAC 外设才能正常运行。还要确保 DAC 输出引脚未被任何其他外设或 DRV 板载入。

表 9-1 列出了用于实时跟踪的 FOC 电机控制变量地址，每个 FOC 环路都会发生更新。相比之下，**pUserStatusRegs** 中的变量每 1ms 更新一次，因此无法显示实时变化。

表 9-1. 用于 DAC 监控的地址表

参数	变量地址/名称			IQ 类型
	无传感器 FOC	通用 FOC	带传感器 FOC	
A 相电流	g_pMotorInputs->current.iabc.a			IQ27
B 相电流	g_pMotorInputs->current.iabc.b			IQ27
C 相电流	g_pMotorInputs->current.iabc.c			IQ27
A 相电流原始 ADC 值	g_pMotorInputs->current.iabcRaw.a			IQ11
B 相电流原始 ADC 值	g_pMotorInputs->current.iabcRaw.b			IQ11
C 相电流原始 ADC 值	g_pMotorInputs->current.iabcRaw.c			IQ11
A 相电压	g_pMotorInputs->voltage.vabc.a			IQ27
B 相电压	g_pMotorInputs->voltage.vabc.b			IQ27
C 相电压	g_pMotorInputs->voltage.vabc.c			IQ27
A 相电压原始 ADC 值	g_pMotorInputs->voltage.vabcRaw.a			IQ12
B 相电压原始 ADC 值	g_pMotorInputs->voltage.vabcRaw.b			IQ12
C 相电压原始 ADC 值	g_pMotorInputs->voltage.vabcRaw.c			IQ12
D 轴电流	0x20200760	g_pMC_App->foc.idq.d		IQ27
Q 轴电流	0x20200764	g_pMC_App->foc.idq.q		IQ27
D 轴电压	0x20200768	g_pMC_App->foc.vdq.d		IQ27
Q 轴电压	0x2020076C	g_pMC_App->foc.vdq.q		IQ27
D 轴经过滤波的 BEMF	0x20200BEC	g_pMC_App->angleTrackingPLLEstim.EdqFilt.d	不适用	IQ27

表 9-1. 用于 DAC 监控的地址表 （续）

参数	变量地址/名称			IQ 类型
	无传感器 FOC	通用 FOC	带传感器 FOC	
Q 轴经过滤波的 BEMF	0x20200BF0	g_pMC_App->angleTrackingPLLEstim.EdqFilt.q	不适用	IQ27
滤波后的估算电机转速	0x20200C0C	g_pMC_App->angleTrackingPLLEstim.velocityFilt	g_pMC_App->foc.hallObj.hallEstimVelocityFilt	IQ27
估算的转子角度	0x20200C14	g_pMC_App->angleTrackingPLLEstim.fluxAngle	g_pMC_App->foc.hallObj.hallEstimFluxAngle	IQ27
功率反馈	0x20200940	g_pMC_App->foc.closeLoop.PowerFeedback		IQ27
SVM 输出占空比 A 相	0x20200730	g_pMC_App->foc.svm.Dabc.a		IQ0
SVM 输出占空比 B 相	0x20200734	g_pMC_App->foc.svm.Dabc.b		IQ0
SVM 输出占空比 C 相	0x20200738	g_pMC_App->foc.svm.Dabc.c		IQ0

备注

在不同的 FOC 算法版本中，变量地址可能有所不同。有关较新的无传感器 FOC 算法版本的变量地址，请参阅最新的 MSPM0 无传感器 FOC 调优指南。

图 9-2 和图 9-3 展示了在 DAC12 输出引脚上输出估算的转子角度变量的示例。

```
161    /* DAC12 output */
162    pUserCtrlRegs->dacCtrl.dacEn              = 1;
163    pUserCtrlRegs->dacCtrl.dacScalingFactor   = _IQ(2.0);     //
164    pUserCtrlRegs->dacCtrl.dacOutAddr         = 0x20200C14;   // Rotor angle
```

图 9-2. DAC12 模块的寄存器设置

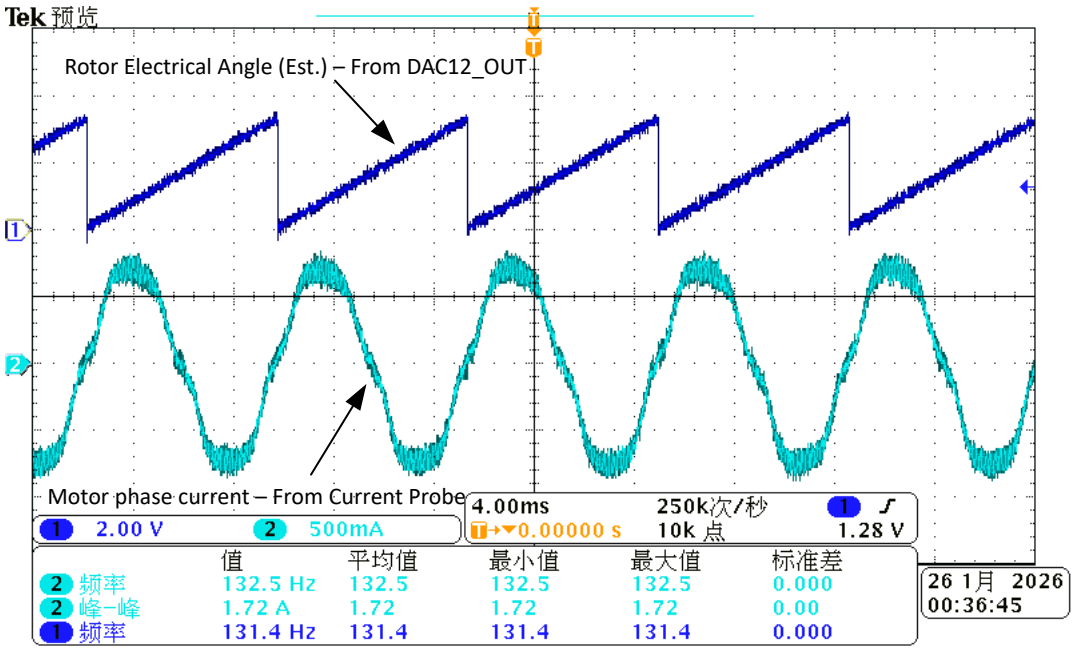


图 9-3. DAC12 模块输出的估算转子角度

备注

对于估算的转子角度，IQ(1.0) 表示 32 个电角度周期，以避免算法溢出。

9.5.2 J-Scope 工具

在运行时可使用 J-Scope 同时跟踪多个变量。请按照以下步骤操作：

1. 将所需变量（详见表 8-1）分配给预设的全局变量，以进行监测。
   a. 为预设变量添加属性： __attribute ((used))
2. 打开 J-Scope 并导入 FOC 工程的 .out（或 .elf）文件。
3. 将第 1 步中的预设变量添加到 J-Scope 观测面板中。
4. 运行 FOC 工程并实时跟踪变量。
5. 需要 UART（或其他通信接口）来动态控制电机，或者在应用代码中使用自动旋转机制。

图 9-4 和图 9-5 展示了一个在运行时同时监测多个变量的示例。

Global Vairable

```
177 volatile __attribute ((used)) int32_t gPhaseCurrentA = 0;
178 volatile __attribute ((used)) int32_t gPhaseCurrentB = 0;
179 volatile __attribute ((used)) int32_t gPhaseCurrentC = 0;
180 volatile __attribute ((used)) int32_t gEstimatedMotorSpeed = 0;
181 volatile __attribute ((used)) int32_t gEstimatedRotorAngle = 0;
```

ISR.c

```
void FOC_ADC_ISR(void)
225     /* Update variable for J-scope observation in @FOC_ADC_ISR() */
226     gPhaseCurrentA = *((int32_t*)(0x20200614));
227     gPhaseCurrentB = *((int32_t*)(0x20200618));
228     gPhaseCurrentC = *((int32_t*)(0x2020061C));
229
230     gEstimatedMotorSpeed = *((int32_t*)(0x20200C0C));
231     gEstimatedRotorAngle = *((int32_t*)(0x20200C14));
232 }
```

图 9-4. J-Scope 跟踪的全局变量设置

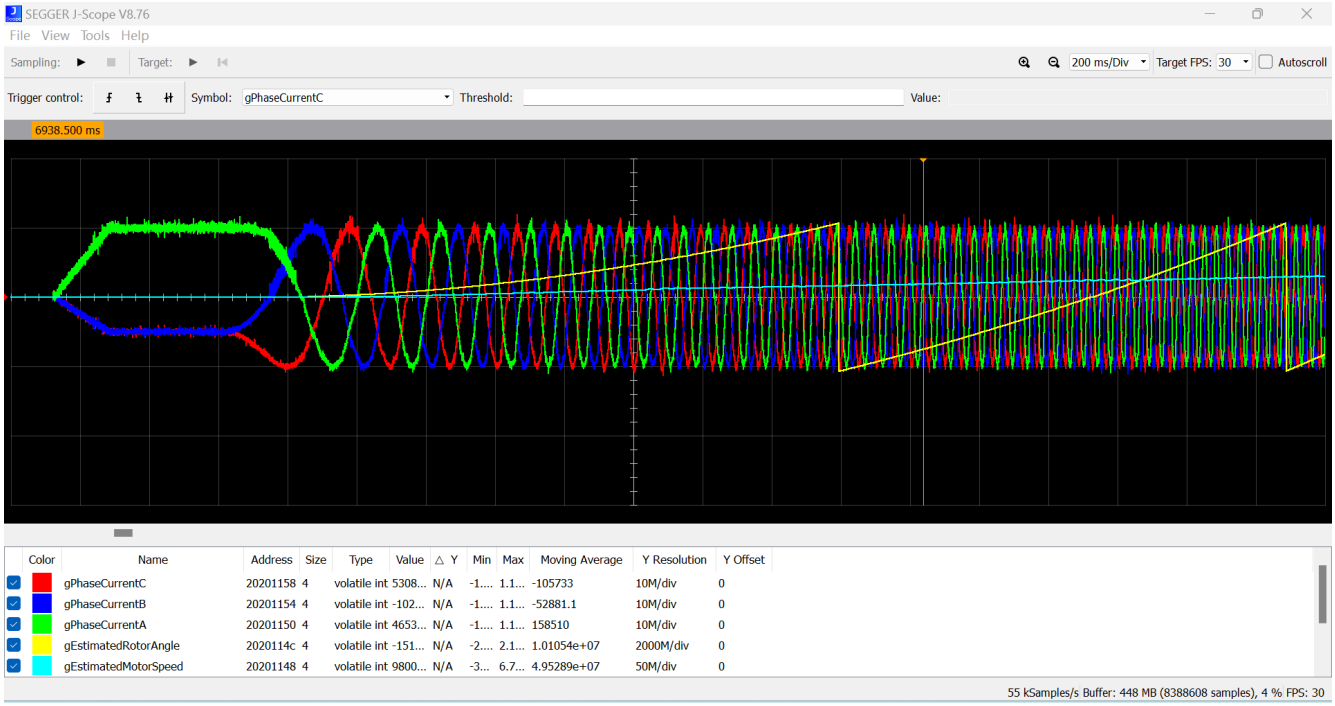


图 9-5. 在 J-Scope 中监测多个变量

10 总结

本文档为用户提供了 MSPM0 FOC 电机控制解决方案的全面指南，涵盖架构、硬件和软件部分。其中介绍了详细的参考工作流程，并提供了有关设置调试环境、调优电机和迁移硬件配置的分步说明和实际示例。

11 参考资料

- 无传感器 FOC SDK 用户指南
- 通用 FOC SDK 用户指南
- 带传感器 FOC SDK 用户指南
- 无传感器 FOC EVM 电路板连接指南
- 带传感器 FOC EVM 电路板连接指南
- UART 通信用户指南
- MSPM0 无传感器 FOC 调优指南
- MSPM0 通用 FOC 调优用户指南
- MSPM0 带传感器 FOC 调优指南
- 使用单一直流链路分流器的 PMSM 无传感器 FOC
- 三相永磁同步电机的无传感器场定向控制

12 修订历史记录

注：以前版本的页码可能与当前版本的页码不同

日期	修订版本	注释
2026 年 3 月	*	初始发行版

重要通知和免责声明

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、与某特定用途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安全、安保法规或其他要求。

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 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 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。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月