参考文献

Design Guide: TIDM-02018

适用于 AM263x 基于 Arm® 的 MCU 器件的通用电机控制参考设计

说明

此参考设计提供适用于 TI 基于 AM263x Arm® 的 MCU 的通用电机控制设计。该设计展示了如何将 AM263x MCU 用于各种类型的 FOC 电机控制技术，例如无传感器 (eSMO) 和有传感器（增量编码器、霍尔传感器）。该设计支持两种主要的硬件设置：一种是使用 AM263x LaunchPad™ 和 3PHGANINV BoosterPack™ 的低压设置，另一种是使用 AM263x controlCARD™ 和 TMDSHVMTRINSPIN 电机控制套件的高压设置。本文档还提供有关将设计迁移到定制电路板以及将工程移植到新器件的说明。

资源

TIDM-02018	设计文件夹
AM2634-Q1	产品文件夹
AM263x MCU+ SDK	工具文件夹
AM263x Academy	培训材料

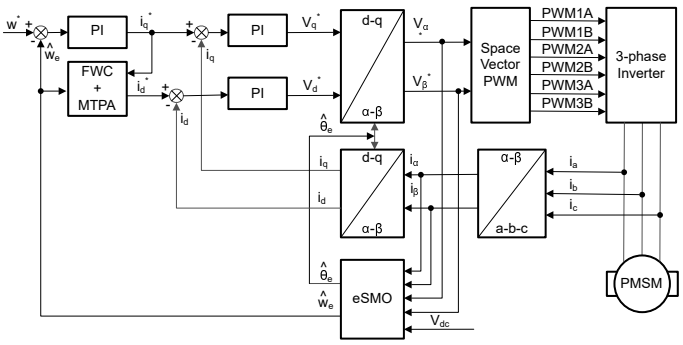
请咨询我司 TI E2E™ 支持专家

特性

- 此参考设计包含丰富的软件包、工具和文档，有助于缩短基于 AM263x MCU 的电机控制系统的开发时间。
- 各种 FOC 电机控制方法：支持无传感器 (eSMO) 和含传感器（增量编码器、霍尔传感器）控制。
- 包含与许多三相逆变器电机评估套件兼容的 TI 系统功能和调试接口。
- 基于 SysConfig 的工程可在不同的器件和电路板之间更轻松地进行迁移。这是通过用户友好的图形用户界面实现的，该界面允许用户调整引脚、外设、软件栈、时钟树和其他元件，从而加速软件开发。

应用

- 混合动力汽车/电动汽车逆变器和电机控制
- 电机驱动器
- 交流逆变器和 VF 驱动器
- 交流驱动器控制模块



English Document: TIDUF67

1 系统说明

此参考设计中介绍的通用电机控制工程不仅能让您对各种电机控制算法进行实验，还可用作您自己设计的参考。这个具有不同 *Build Configuration* 选项的工程提供不同的 FOC 电机控制技术，包括无传感器 (eSMO) 和含传感器（增量编码器、霍尔传感器）以及 TI 的系统特性和调试接口。此工程提供各种功能，例如：数据记录、软件频率响应分析器 (SFRA)、电机 PI 调优、磁场减弱、每安培最大扭矩 (MPTA)、CPU 时间计算、ePWM DAC 模式、阶跃响应模块和相位调整。该参考设计评估可从 TI.com 订购的两款硬件套件：

- BOOSTXL-3PHGANINV + LVSERVOMTR + LP-AM263
- TMDSHVMTRINSPIN + HVPMSMMTR + TMDSCNCD263

1.1 术语

FOC	场定向控制
eSMO	增强型滑模观测器
PMSM	永久磁性同步电机
EEMF	扩展电动势
PLL	锁相环
IPMS	内部 PMSM
FW	磁场削弱
MTPA	每安培最大扭矩
SVPWM	空间矢量调制
PWM	脉宽调制
RPM	每分钟转数

1.2 主要系统规格

- 要了解带有基于采样电阻的内嵌式电机相电流检测评估模块的 48V 三相逆变器，请参阅 BOOSTXL-3PHGANINV 设计文件和技术文档。
- 有关高压电机控制套件的信息，请参阅 TMDSHVMTRINSPIN。
- 有关低压伺服电机、编码器和线束的信息，请参考 LVSERVOMTR 数据表。
- 有关高压永磁同步电机的信息，请参阅 HVPMSMMTR 数据表。

2 系统概述

2.1 方框图

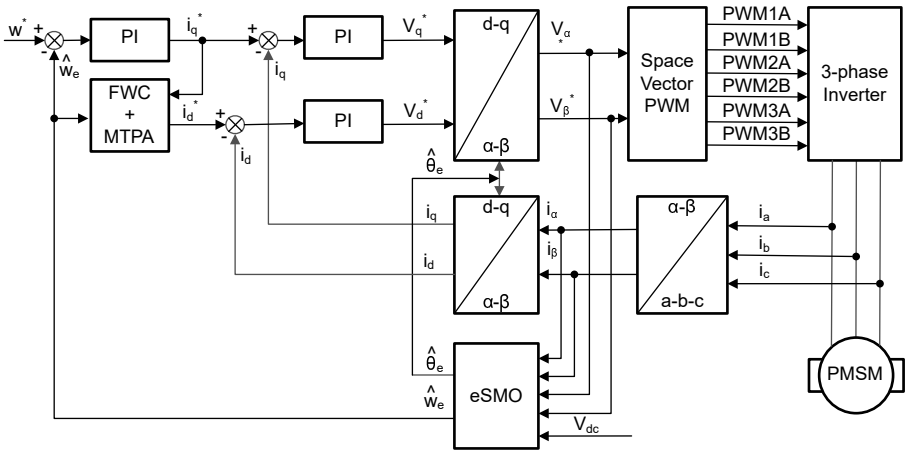


图 2-1. 基于 eSMO 的 PMSM 电机无传感器 FOC 方框图

2.2 主要产品

2.2.1 AM263x 微控制器

AM263x 基于 Arm® 的微控制器旨在满足下一代工业和汽车嵌入式产品复杂的实时处理需求。AM263x MCU 系列包含多个具有多达四个 400MHz Arm® Cortex®-R5F 内核的引脚对引脚兼容器件。对于不同的功能安全配置，可以选择将多个 Arm® 内核编程为在锁步选项中运行。工业通信子系统 (ICSS) 支持集成工业以太网通信，例如 PROFINET IRT、TSN 或 EtherCAT® (以及许多其他工业以太网) ，或用于标准以太网连接或自定义 I/O 接口。AM263x 系列设计用于先进的电机控制和带有先进模拟模块的数字电源控制应用。

2.2.1.1 TMDSCNCD263

AM263x 控制卡评估模块 (EVM) 是一款适用于德州仪器 (TI) Sitara™ AM263x 系列微控制器 (MCU) 的评估和开发板。此 EVM 具有用于编程和调试的板载仿真功能以及用于简化用户界面的按钮和 LED，可让您在 AM263x MCU 上轻松开始开发牵引逆变器设计。该控制卡还支持通过接头引脚访问重要信号，以进行快速原型设计。

图 2-2. AM263x controlCARD™

2.2.1.2 LP-AM263

LP-AM263 是一款适用于 AM263x 系列 Sitara™ 高性能微控制器 (MCU) 的成本优化型开发板。该板提供易于使用的标准化平台来开发下一个应用，非常适用于初始评估和原型设计。

LP-AM263 配备 Sitara AM2634 处理器以及其他元件，使用户可以利用各种器件接口，包括工业以太网 (IE)、标准以太网、快速串行接口 (FSI) 等，从而轻松创建原型。AM2634 支持各种 IE 协议，例如 EtherCAT、EtherNet/IP 和 PROFINET®。

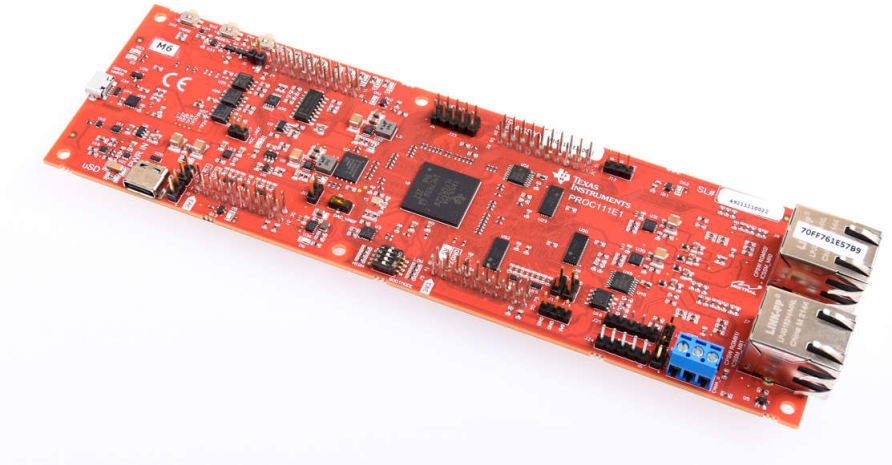


图 2-3. AM263x LaunchPad™

3 系统设计原理

3.1 三相 PMSM 驱动器

永磁同步电机 (PMSM) 具有一个绕线定子、一个永磁转子组件和用于检测转子位置的内部或外部器件。感测器件提供位置反馈以适当地调整定子基准电压的频率和振幅，从而使磁体组件保持旋转。一个内部永磁转子和外部绕组的组合提供低转子惯性、有效散热和电机尺寸减少等优势。

- 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，请控制定子电流。
- 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。永磁体非常适合范围高达几千瓦的同步电机。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，因此转子的旋转会产生一个有用的机械输出，如图 3-1 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，并且在机器部件上产生过多的机械抖动、噪声和机械应力。此外，如果转子惯性使转子不能对这些摆动做出响应，那么转子在同步频率上停止转动，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。

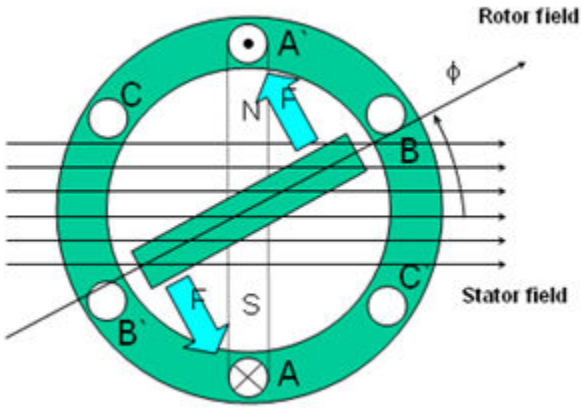


图 3-1. 旋转的定子磁通和转子磁通之间的相互作用产生扭矩

3.1.1 PMSM 的数学模型和 FOC 结构

PMSM 的 FOC 结构如图 2-1 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。

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方程式 1 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 是 q 轴定子端电压
- v_q 是 d 轴定子端电压
- i_d 是 d 轴定子电流
- i_q 是 q 轴定子电流
- L_d 是 q 轴电感
- L_q 是 d 轴电感
- p 是导数算子
- 方程式 2 的简写表示永磁体产生的磁链
- R_S 是定子绕组的电阻
- ω_e 是转子的电角速度

$$\frac{d}{dt};\lambda_{pm} \tag{2}$$

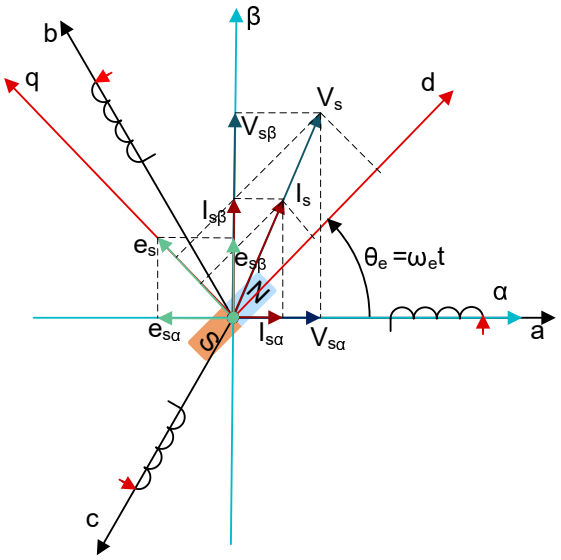


图 3-2. PMSM 建模坐标系的定义

通过使用如图 3-2 所示的 Park 逆变换，PMSM 的动力学可以在 α - β 静止坐标系中建模为：

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \begin{bmatrix} R_s + pL_d & \omega_e(L_d - L_q) \\ -\omega_e(L_d - L_q) & R_s + pL_q \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} \tag{3}$$

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：

$$\begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} = \left(\lambda_{pm} + (L_d - L_q)i_d\right)\omega_e \begin{bmatrix} -\sin(\theta_e) \\ \cos(\theta_e) \end{bmatrix} \tag{4}$$

根据方程式 3 和方程式 4，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解耦出来，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 电压公式方程式 5 改写为状态公式：

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \frac{1}{L_d}\begin{bmatrix} -R_s & -\omega_e(L_d - L_q) \\ \omega_e(L_d - L_q) & -R_s \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \frac{1}{L_d}\begin{bmatrix} V_\alpha - e_\alpha \\ V_\beta - e_\beta \end{bmatrix} \tag{5}$$

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：

$$S(x) = \begin{bmatrix} \hat{i}_\alpha - i_\alpha \\ \hat{i}_\beta - i_\beta \end{bmatrix} = \begin{bmatrix} \tilde{i}_\alpha \\ \tilde{i}_\beta \end{bmatrix} \quad (6)$$

其中

- 方程式 7 和方程式 8 是估算的电流
- 上标 ^ 表示变量为估算值
- 上标 ~ 表示变量为变量误差，即观测值与实际测量值之间的差异

$$\hat{i}_\alpha \quad (7)$$

$$\hat{i}_\beta \quad (8)$$

3.1.2 PM 同步电机的磁场定向控制

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 3-3 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)。

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 3-3 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。

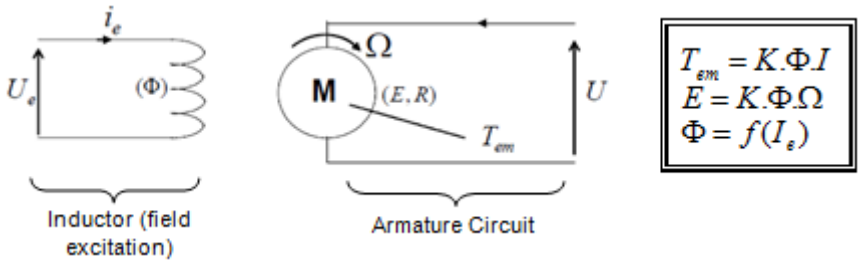


图 3-3. 在直流电机模型中磁通和扭矩是独立控制的

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，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 9 所示。

$$\tau_{em} = \vec{B}_{stator} \times \vec{B}_{rotor} \quad (9)$$

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 10 所示。

$$\tau_{em} \propto \psi_R \times i_{sq} \tag{10}$$

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。

3.1.2.1 (a, b) → (α, β) 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 3-4 所示的矢量图。

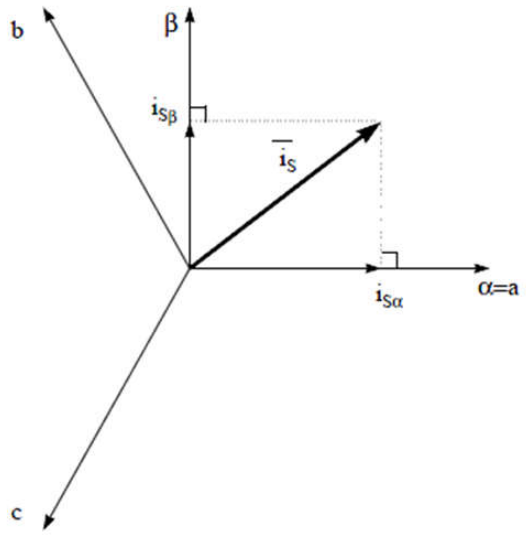


图 3-4. 静止坐标系中的定子电流空间矢量

将三相系统修改为（α，β）二维正交系统的投影如方程式 11 所示。

$$i_{s\alpha} = i_a$$
$$i_{s\beta} = \frac{1}{\sqrt{3}} i_a + \frac{2}{\sqrt{3}} i_b \tag{11}$$

两相（α，β）电流仍取决于时间和速度。

3.1.2.2 (α , β) → (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 3-5 显示了来自该二维坐标系的电流矢量的关系。

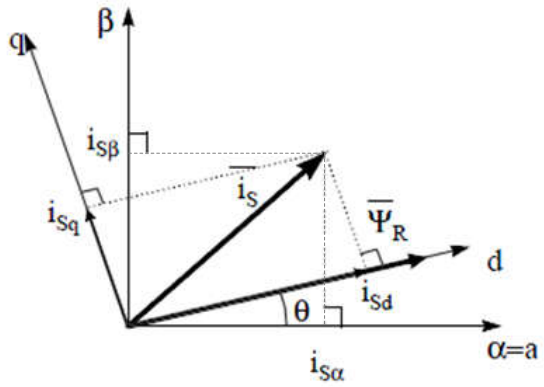


图 3-5. d,q 旋转坐标系中的定子电流空间矢量

电流矢量的磁通和扭矩分量由方程式 12 决定。

$$i_{sd} = i_{s\alpha}\cos(\theta) + i_{s\beta}\sin(\theta)$$
$$i_{sq} = -i_{s\alpha}\sin(\theta) + i_{s\beta}\cos(\theta) \quad (12)$$

在这里， 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。

3.1.2.3 交流电机 FOC 基本配置方案

图 3-6 总结了使用 FOC 进行扭矩控制的基本配置方案：

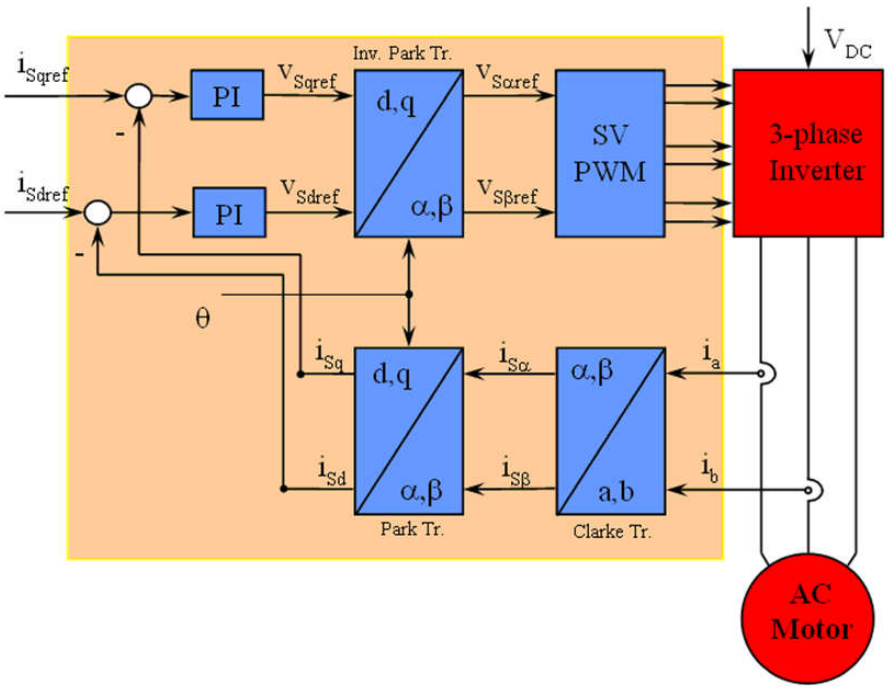


图 3-6. 交流电机 FOC 基本配置方案

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。此时，这个控制结构具有一个有意思的优势：只需改变磁通基准并获得转子磁通位置，该控制结构即可用于控制同步或感应电机。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} 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。

3.1.2.4 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 3-7 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。

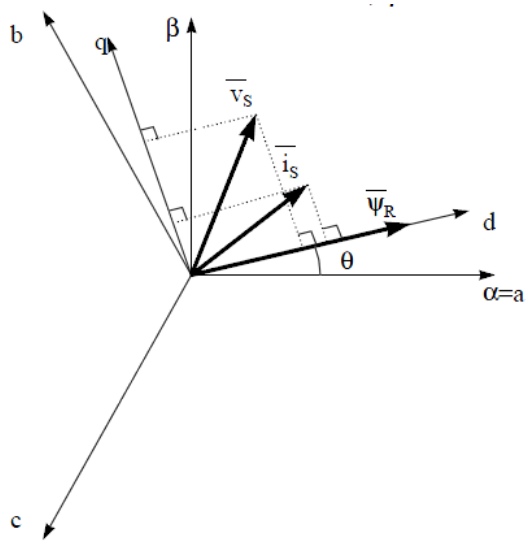


图 3-7. (d, q) 旋转坐标系中的电流、电压和转子磁通空间矢量

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。从静止基准框架到同步旋转基准框架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 轴电流控制扭矩，而 d 轴电流强制设置为零。因此，该系统的关键模块是使用增强型滑模观测器 (eSMO) 来估算转子位置。

3.1.3 PM 同步电机的无传感器控制

在家用电器应用中，如果使用机械传感器，将会导致成本、尺寸和可靠性问题增加。为了克服这些问题，无传感器控制方法应运而生，它可以通过多种估算方法在没有机械位置传感器的情况下获得转子转速和位置信息。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。

3.1.3.1 具有锁相环的增强型滑模观测器

基于模型的方法用于实现 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。其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。

3.1.3.1.1 PMSM 的 ESMO 设计

图 3-8 展示了集成在 SMO 中的传统 PLL。

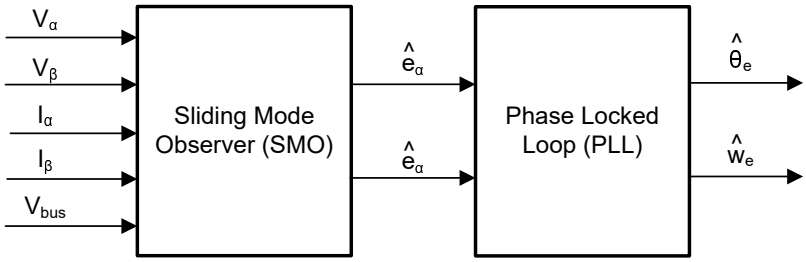


图 3-8. 包含用于 PMSM 的 PLL 的 eSMO 方框图

构建了传统的降阶滑模观测器，其数学模型如方程式 13 所示，方框图如图 3-9 所示。

$$\begin{bmatrix}\hat{i}_\alpha\\ \hat{i}_\beta\end{bmatrix}=\frac{1}{L_d}\begin{bmatrix}-R_s & -\hat{\omega}_e(L_d-L_q)\\ \hat{\omega}_e(L_d-L_q) & -R_s\end{bmatrix}\begin{bmatrix}\hat{i}_\alpha\\ \hat{i}_\beta\end{bmatrix}+\frac{1}{L_d}\begin{bmatrix}V_\alpha-\hat{e}_\alpha+z_\alpha\\ V_\beta-\hat{e}_\beta+z_\beta\end{bmatrix} \tag{13}$$

其中 z_α 和 z_β 是滑模反馈分量，其定义为：

$$\begin{bmatrix}z_\alpha\\ z_\beta\end{bmatrix}=\begin{bmatrix}k_\alpha \mathrm{sign}(\hat{i}_\alpha-i_\alpha)\\ k_\beta \mathrm{sign}(\hat{i}_\beta-i_\beta)\end{bmatrix} \tag{14}$$

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，以保证 SMO 的稳定运行，则 k_α 和 k_β 通常足够大，以保持 方程式 15 和 方程式 16。

$$k_\alpha > \max(|e_\alpha|) \tag{15}$$

$$k_\beta > \max(|e_\beta|) \tag{16}$$

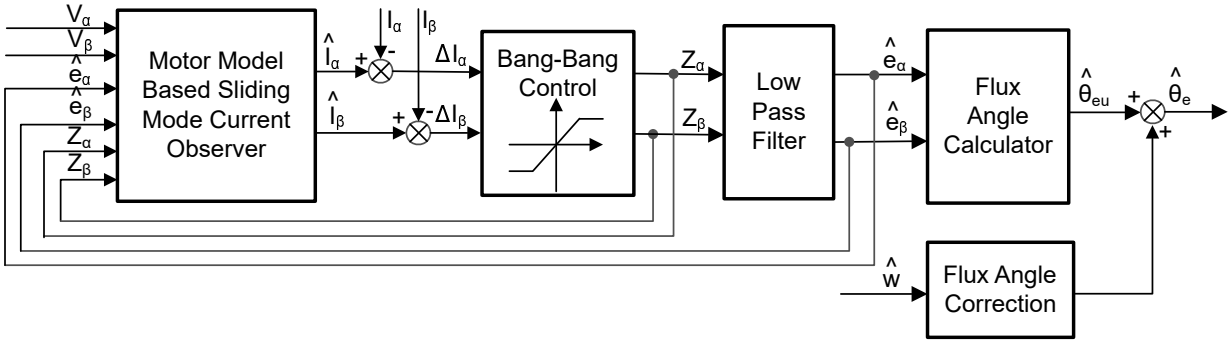


图 3-9. 传统滑模观测器的方框图

α - β 轴上的 EEMF 估算值（方程式 18、方程式 19）可通过低通滤波器从不连续开关信号 z_α 和 z_β 中获得：

$$\begin{bmatrix}\hat{e}_\alpha\\ \hat{e}_\beta\end{bmatrix}=\frac{\omega_c}{s+\omega_c}\begin{bmatrix}z_\alpha\\ z_\beta\end{bmatrix} \tag{17}$$

$$\hat{e}_\alpha \tag{18}$$

$$\hat{e}_\beta \tag{19}$$

其中 方程式 20 是 LPF 的截止角频率，通常根据定子电流的基频来选择该截止角频率。

$$\omega_c = 2\pi f_c \tag{20}$$

因此，转子位置可以直接通过反电动势的反正切计算得出，其定义如下

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

低通滤波器消除了滑模函数的高频项，从而导致出现相位延迟。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：

$$\Delta\theta_e = -\tan^{-1}\left(\frac{\omega_e}{\omega_c}\right) \tag{22}$$

这样使用 SMO 方法估算的转子位置就为：

$$\hat{\theta}_e = -\tan^{-1}\left(\frac{\hat{e}_\alpha}{\hat{e}_\beta}\right) + \Delta\theta_e \tag{23}$$

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 13 的时间离散系统矩阵由方程式 24 给出：

$$\begin{bmatrix}\hat{i}_\alpha(n+1)\\ \hat{i}_\beta(n+1)\end{bmatrix} = \begin{bmatrix}F_\alpha\\ F_\beta\end{bmatrix}\begin{bmatrix}\hat{i}_\alpha(n)\\ \hat{i}_\beta(n)\end{bmatrix} + \begin{bmatrix}G_\alpha\\ G_\beta\end{bmatrix}\begin{bmatrix}V^*_\alpha(n) - \hat{e}_\alpha(n) + z_\alpha(n)\\ V^*_\beta(n) - \hat{e}_\beta(n) + z_\beta(n)\end{bmatrix} \tag{24}$$

其中矩阵 *[F]* 和 *[G]* 由方程式 25 和方程式 26 给出：

$$\begin{bmatrix}F_\alpha\\ F_\beta\end{bmatrix} = \begin{bmatrix}e^{-\frac{R_s}{L_d}}\\ e^{-\frac{R_s}{L_q}}\end{bmatrix} \tag{25}$$

$$\begin{bmatrix}G_\alpha\\ G_\beta\end{bmatrix} = \frac{1}{R_s}\begin{bmatrix}1 - e^{-\frac{R_s}{L_d}}\\ 1 - e^{-\frac{R_s}{L_q}}\end{bmatrix} \tag{26}$$

方程式 17 的时间离散形式由方程式 27 给出：

$$\begin{bmatrix}\hat{e}_\alpha(n+1)\\ \hat{e}_\beta(n+1)\end{bmatrix} = \begin{bmatrix}\hat{e}_\alpha(n)\\ \hat{e}_\beta(n)\end{bmatrix} + 2\pi f_c\begin{bmatrix}z_\alpha(n) - \hat{e}_\alpha(n)\\ z_\beta(n) - \hat{e}_\beta(n)\end{bmatrix} \tag{27}$$

3.1.3.1.2 使用 PLL 的转子位置和转速估算

在反正切法中，由于噪声和谐波分量的存在，位置和转速估算的精度会受到影响。为了消除该问题，可使用 PLL 模型对 IPMSM 的无传感器控制结构中的转速和位置进行估算。节 3.1.3.1.1 中说明了与 SMO 配合使用的 PLL 结构。反电动势估算方程式 18 和方程式 19 可与 PLL 模型配合使用来估算电机角速度和位置，如图 3-10 所示。

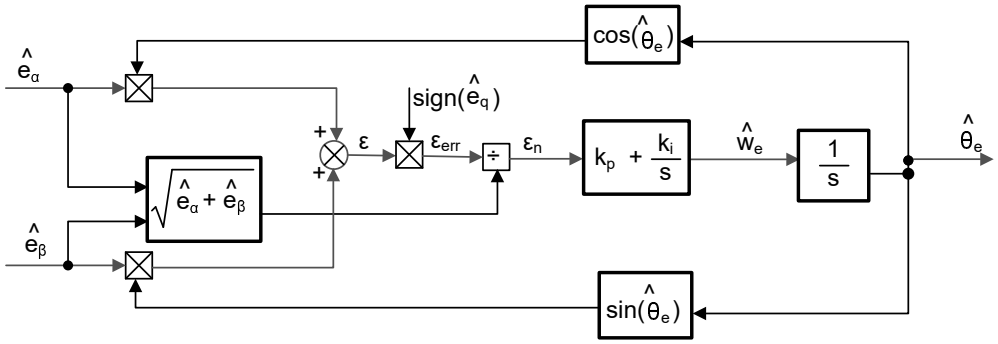


图 3-10. 锁相环位置跟踪器的方框图

由于方程式 28、方程式 29 和方程式 30

$$e_\alpha = E\cos(\theta_e) \tag{28}$$

$$e_\beta = E\sin(\theta_e) \tag{29}$$

$$E = \omega_e \lambda_{pm} \tag{30}$$

位置误差可定义为：

$$\varepsilon = \hat{e}_\beta \cos\left(\hat{\theta}_e\right) - \hat{e}_\alpha \sin\left(\hat{\theta}_e\right) = E\sin(\theta_e)\cos\left(\hat{\theta}_e\right) - E\cos(\theta_e)\sin\left(\hat{\theta}_e\right) = E\sin\left(\theta_e - \hat{\theta}_e\right) \tag{31}$$

其中 E 是 EEMF 的幅度，与电机转速 ω_e 成正比。当方程式 32 时，方程式 31 可简化为

$$\left(\theta_e - \hat{\theta}_e\right) < \frac{\pi}{2} \tag{32}$$

$$\varepsilon = E\left(\theta_e - \hat{\theta}_e\right) \tag{33}$$

可以进一步得到 EEMF 归一化后的位置误差：

$$\varepsilon_n = \theta_e - \hat{\theta}_e \tag{34}$$

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 3-11 所示。PLL 的闭环传递函数可表示为：

$$\frac{\hat{\theta}_e}{\theta_e} = \frac{k_p s + k_i}{s^2 + k_p s + k_i} = \frac{2\xi\omega_n s + \omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{35}$$

其中 k_p 和 k_i 是标准 PI 调节器的比例增益和积分增益，固有频率 ω_n 和阻尼比 ξ 计算公式为

$$k_p = 2\xi\omega_n, \quad k_i = \omega_n^2 \tag{36}$$

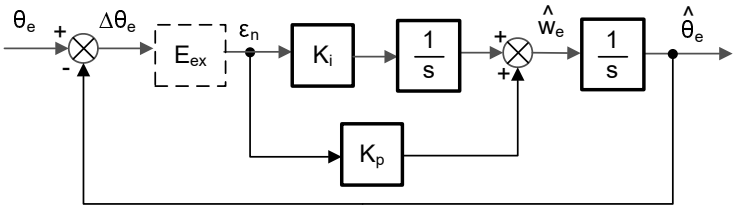


图 3-11. 锁相环位置跟踪器的简化方框图

3.1.4 电机驱动器的硬件必要条件

用于控制电机的算法利用电机条件的采样测量值，包括直流母线电源电压和每个电机相位的电流。需要正确设置一些与硬件相关的参数（例如电流标度值、电压标度值和电压滤波器极点），才能正确识别电机并使用磁场定向控制 (FOC) 有效地运行电机。

3.1.5 额外的控制特性

3.1.5.1 弱磁 (FW) 和每安培最大扭矩 (MTPA) 控制

永磁同步电机 (PMSM) 因其高功率密度、高效率和宽转速范围而广泛应用于家用电器应用。PMSM 包含两种主要类型：表面贴装式 PMSM (SPM) 和内嵌式 PMSM (IPM)。由于 SPM 电机在扭矩和 q 轴电流之间具有线性关系，因此更易于控制。不过，IPMSM 由于凸极比大而具有电磁扭矩和磁阻扭矩。总扭矩相对于转子角度是非线性的。因此，MTPA 技术可用于 IPM 电机，以优化恒定扭矩区域中的扭矩生成。弱磁控制的目的是优化以达到 PMSM 驱动器的最高功率和效率。弱磁控制可以使电机以其基本转速运行，扩大其运行限值以使转速高于额定转速，并允许在整个转速和电压范围内实现出色控制。

IPMSM 数学模型的电压公式可以用 d-q 坐标来描述，如方程式 37 和方程式 38 所示。

$$v_d = L_d\frac{di_d}{dt} + R_s i_d - p\omega_m L_q i_q \tag{37}$$

$$v_q = L_q\frac{di_q}{dt} + R_s i_q + p\omega_m L_d i_d + p\omega_m \psi_m \tag{38}$$

图 3-12 显示了 IPM 同步电机的动态等效电路。

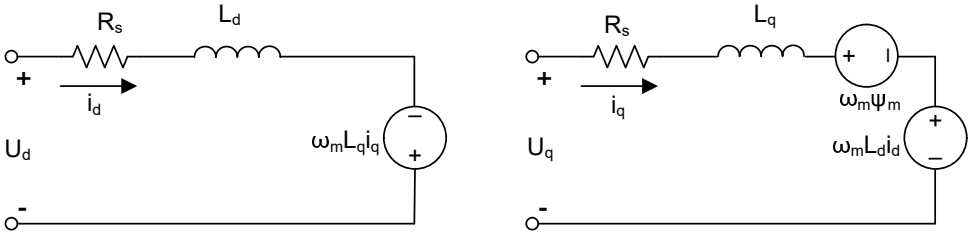


图 3-12. IPM 同步电机的等效电路

IPMSM 产生的总电磁扭矩可以由方程式 40 表示，产生的扭矩包含两个不同的项。第一项对应于扭矩电流 i_q 和永磁体

$$\psi_m \tag{39}$$

之间产生的相互反作用力扭矩，而第二项对应于由于 d 轴和 q 轴上的电感不同而产生的磁阻扭矩。

$$T_e = \frac{3}{2}p\left[\psi_m i_q + \left(L_d - L_q\right)i_d i_q\right] \tag{40}$$

在大多数应用中，IPMSM 驱动器具有转速和扭矩约束，这主要是由于分别存在逆变器或电机额定电流以及可用的直流链路电压限制。这些约束可以用数学公式方程式 41 和方程式 42 进行表示。

$$I_a = \sqrt{i_d^2 + i_q^2} \le I_{max} \tag{41}$$

$$V_a = \sqrt{v_d^2 + v_q^2} \le V_{max} \tag{42}$$

其中 V_{max} 和 I_{max} 是逆变器或电机允许的最大电压和电流。在两级三相电压源逆变器 (VSI) 供电的电机中，可实现的最大相电压受直流链路电压和 PWM 策略的限制。如果采用空间矢量调制 (SVPWM)，则最大电压限制为方程式 43 中所示的值。

$$\sqrt{v_d^2 + v_q^2} \le v_{max} = \frac{v_{dc}}{\sqrt{3}} \tag{43}$$

通常，定子电阻 R_s 在高速运行时可以忽略不计，并且电流的导数在稳态下为零，因此得到方程式 44，如下所示。

$$\sqrt{L_d^2\left(i_d + \frac{\psi_{pm}}{L_d}\right)^2 + L_q^2 i_q^2} \le \frac{V_{max}}{\omega_m} \tag{44}$$

方程式 41 的电流限制在 d-q 平面中产生一个半径为 I_{max} 的圆，而方程式 43 的电压限制产生一个椭圆，其半径 V_{max} 随着转速的增加而减小。必须对得到的 d-q 平面电流矢量进行控制，使其同时遵守电流和电压约束。根据这些约束，可以区分 IPMSM 的三个工作区域，如图 3-13 所示。

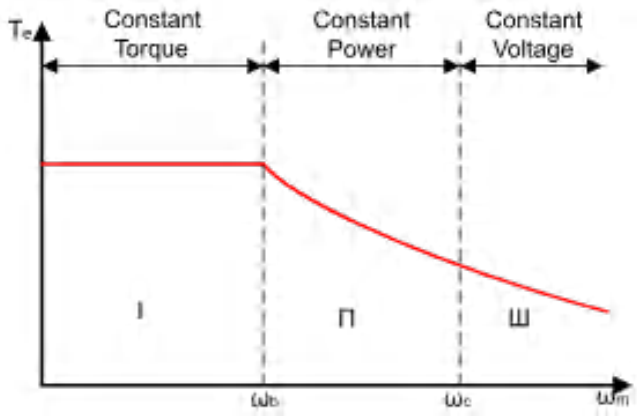


图 3-13. IPMSM 控制工作区域

1. 恒定扭矩区域：可以在该工作区域内实施 MTPA，从而可产生最大扭矩。
2. 恒定功率区域：必须采用弱磁控制，并且在达到电流约束时减小扭矩容量。
3. 恒定电压区域：在这个工作区域，深度弱磁控制使定子电压保持恒定，以尽可能大地产生扭矩。

在恒定扭矩区域，根据方程式 40，IPMSM 的总扭矩包括来自磁链的电磁扭矩和来自 L_d 和 L_q 之间凸极的磁阻扭矩。电磁扭矩与 q 轴电流 i_q 成正比，磁阻扭矩与 d 轴电流 i_d、q 轴电流 i_q 以及 L_d 与 L_q 之差的乘积成正比。

SPM 电机的传统矢量控制系统仅通过将命令的 i_d 设置为零来实现非弱磁模式，从而利用电磁扭矩。但 IPMSM 会利用电机的磁阻扭矩，也必须控制 d 轴电流。MTPA 控制的目的是计算基准电流 i_d 和 i_q，以尽可能增大产生的电磁扭矩与磁阻扭矩之间的比率。以下公式显示了 i_d 和 i_q 之间的关系以及定子电流 I_s 的矢量和。

$$I_s = \sqrt{i_d^2 + i_q^2} \tag{45}$$

$$I_d = I_s \cos\beta \tag{46}$$

$$I_q = I_s \sin\beta \tag{47}$$

其中 β 是同步 (d-q) 坐标系中的定子电流角度。方程式 40 可以表示为方程式 48，其中 I_s 替换了 i_d 和 i_q。

方程式 48 表明电机扭矩取决于定子电流矢量的角度，因此

$$T_e = \frac{3}{2} p I_s \sin\beta\left[\psi_m + (L_d - L_q) I_s \cos\beta\right] \tag{48}$$

当电机扭矩微分等于零时，可以计算出最大效率点。当该微分

$$\frac{dT_e}{d\beta} \tag{49}$$

为零（如方程式 50 所示）时，可以找到 MTPA 点。

$$\frac{dT_e}{d\beta} = \frac{3}{2}p\left[\psi_m I_s \cos\beta + (L_d - L_q) I_s^2 \cos 2\beta\right] = 0 \quad (50)$$

接下来，可以通过方程式 51 得出 MTPA 控制的电流角度。

$$\beta_{mtpa} = \cos^{-1}\frac{-\psi_m + \sqrt{\psi_m^2 + 8 \times (L_d - L_q)^2 \times I_s^2}}{4 \times (L_d - L_q) \times I_s} \quad (51)$$

因此，可以使用 MTPA 控制的电流角度通过方程式 52 和方程式 53 来表示有效的 d 轴和 q 轴基准电流。

$$I_d = I_s \times \cos\beta_{mtpa} \quad (52)$$

$$I_q = I_s \times \sin\beta_{mtpa} \quad (53)$$

不过，如方程式 51 所示，MTPA 控制的角度 β_{mtpa} 与 d 轴和 q 轴电感有关。这意味着电感的变化会阻碍找到出色的 MTPA 点。为了提高电机驱动器的效率，必须在线估算 d 轴和 q 轴电感，但参数 L_d 和 L_q 不易于在线测量，并且受饱和效应的影响。稳健的查询表 (LUT) 方法可确保电气参数变化下的可控性。通常，为了简化数学模型，可以忽略 d 轴和 q 轴电感之间的耦合效应。因此，假设 L_d 仅随 i_d 变化，L_q 仅随 i_q 变化。因此，d 轴和 q 轴电感可以分别建模为 d-q 电流的函数，如方程式 54 和方程式 55 所示。

$$L_d = f_1(i_d, i_q) = f_1(i_d) \quad (54)$$

$$L_q = f_2(i_q, i_d) = f_2(i_q) \quad (55)$$

为了通过简化方程式 51 来减轻 ISR 计算负担，基于电机参数的常数 K_{mtpa} 改为用方程式 56 表示，其中 K_{mtpa} 在后台循环中使用更新的 L_d 和 L_q 计算。

$$K_{mtpa} = \frac{\psi_m}{4 \times (L_q - L_d)} = 0.25 \times \frac{\psi_m}{(L_q - L_d)} \quad (56)$$

$$\beta_{mtpa} = \cos^{-1}\left(K_{mtpa} \div I_s - \sqrt{(K_{mtpa} \div I_s)^2 + 0.5}\right) \quad (57)$$

对第二个中间变量 G_{mtpa}（如方程式 58 中所述）进行了定义，以便进一步简化计算。使用 G_{mtpa}，MTPA 控制的角度 β_{mtpa} 的计算公式为方程式 59。这两个计算在 ISR 中执行，以获得真实的电流角度 β_{mtpa}。

$$G_{mtpa} = K_{mtpa} \div I_s \quad (58)$$

$$\beta_{mtpa} = \cos^{-1}\left(G_{mtpa} - \sqrt{G_{mtpa}^2 + 0.5}\right) \quad (59)$$

在所有情况下，通过作用于直轴电流 i_d 可以削弱磁通量，从而扩展可实现的速度范围。在进入该恒定功率工作区域后，选择弱磁控制而不是在恒定功率和恒定电压区域中使用的 MTPA 控制。由于最大逆变器电压受到限制，PMSM 电机无法在反电动势（几乎与永磁场和电机转速成正比）高于逆变器最大输出电压的转速区域中运行。在 PM 电机中，无法直接控制磁通量。然而，通过添加负 i_d，由于出现 d 轴电枢反应，空气间隙通量会因为消磁作用而减弱。考虑到电压和电流约束，电枢电流和端子电压会受到限制，如方程式 41 和方程式 42 所示。逆变器输入电压（直流链路电压）的变化限制了电机的最大输出。此外，最大基波电机电压还取决于所使用的 PWM 方法。在方程式 44 中，IPMSM 有两个因素：一个是永磁值，另一个是电感和磁通电流。

图 3-14 显示了用于实现弱磁的典型控制结构。

β_{fw} 是弱磁 (FW) PI 控制器的输出，可生成基准 i_d 和 i_q。在电压幅度达到限制之前，FW 的 PI 控制器的输入始终为正，因此输出始终在 0 处达到饱和。

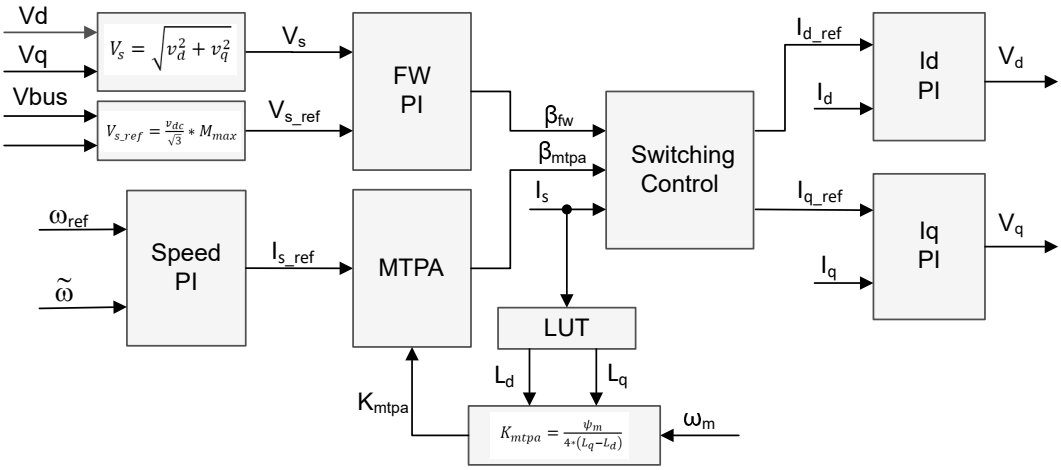


图 3-14. 弱磁和每安培最大扭矩控制的方框图

电机驱动 FOC 系统中有两个控制模块：一个是 MTPA 控制，一个是弱磁控制。这两个模块根据输入参数分别生成电流角度 β_{mtpa} 和 β_{fw}，如图 3-15 所示。

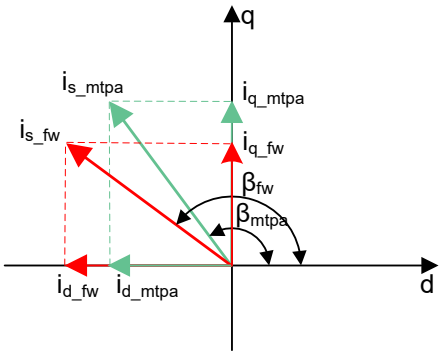


图 3-15. FW 和 MTPA 期间 IPMSM 的电流相量图

切换控制模块用于决定可应用哪个角度，然后计算基准 i_d 和 i_q，如方程式 46 和方程式 47 所示。可以根据下面的方程式 60 和方程式 61 来选择电流角度。

$$\beta = \beta_{fw} \text{ if } \beta_{fw} > \beta_{mtpa} \tag{60}$$

$$\beta = \beta_{mpta} \text{ if } \beta_{fw} < \beta_{mtpa} \tag{61}$$

图 3-16 展示了该参考设计中 PMSM 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。

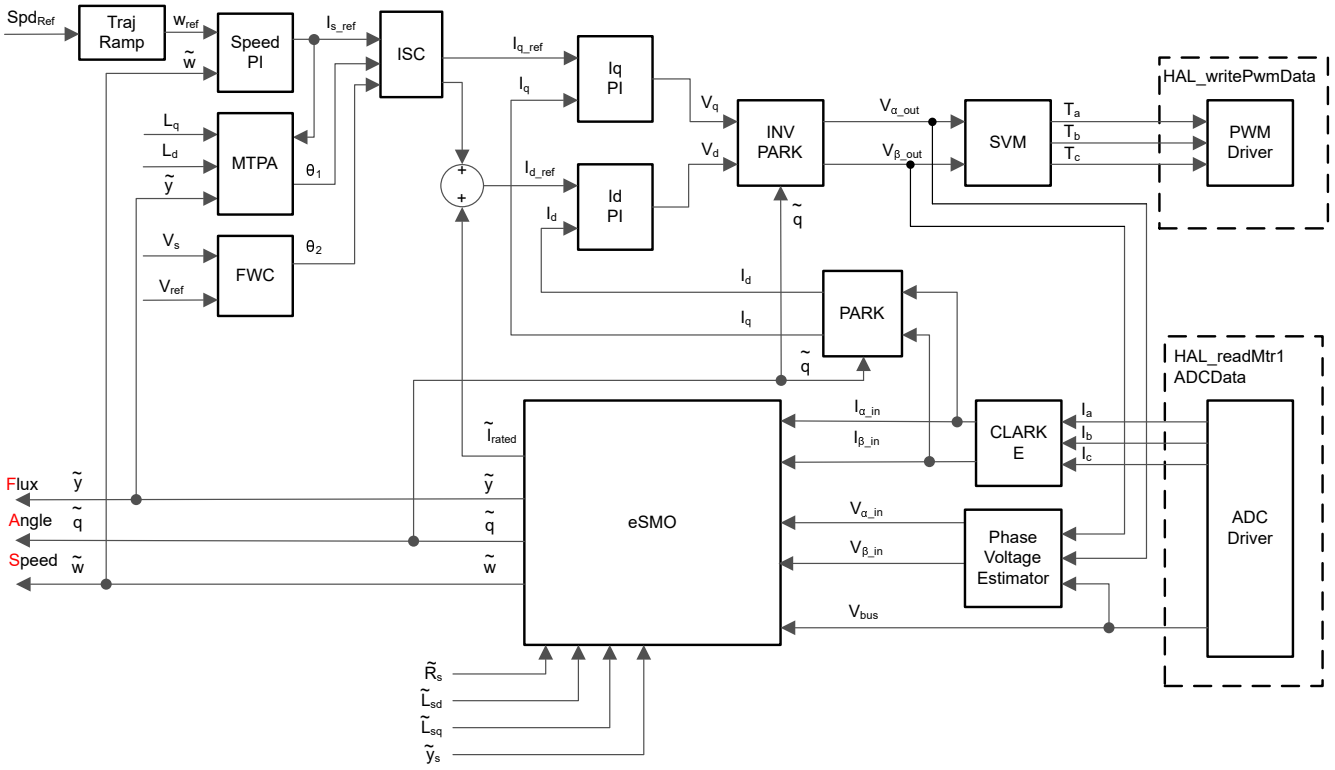


图 3-16. 使用 eSMO 并具有 FWC 和 MTPA 功能的 PMSM 的无传感器 FOC

3.1.5.2 快速启动

快速启动 (FS) 功能使驱动器能够确定旋转电机的转速和方向，并以该转速和方向开始输出电压和频率。如果没有快速启动功能，那么驱动器将以零伏和零转速开始输出，并尝试增加至命令的速度。如果负载的惯性或旋转方向要求电机产生很大的扭矩，则可能会导致过大的电流，并且可能会在驱动器上发生过流跳闸。这些问题可以通过快速启动加以解决。

快速启动是指以除零以外的任何速度启动控制能力，这是空调应用中用于风扇驱动的重要功能。

当电机以正常模式启动时，控制器最初应用 0Hz 的频率并增加至所需的频率。如果驱动器在该模式下启动，并且电机已经以非零频率旋转，则会产生大电流。如果电流限制器反应不够快，则可能会导致过流跳闸。即使电流限制器足够快，可以防止过流跳闸，发生同步和电机达到所需频率也可能需要不可接受的时长。此外，会在应用上施加更大的机械应力。

在快速启动模式下，驱动器对启动命令的响应是与电机的转速（频率和相位）和电压同步。然后电机加速至命令的频率。该过程可防止过流跳闸并显著减少电机达到命令的频率所需的时间。由于驱动器以旋转速度与电机同步并增加至适当的转速，因此几乎不存在机械应力或不存在机械应力。

快速启动功能实现了一种搜索转子转速的算法。该算法搜索与施加到电机上的励磁电流相对应的电机电压。

当电机旋转时，可以通过 BEMF 电压估算转速和位置信息。由于测量定子电压，因此通过开关逆变器可以轻松获得转速和位置。向电机施加零扭矩电流并测量产生的电流和定子电压，然后 FOC 模块使用这些信号来估算转子位置和转速。

图 3-17 显示了具有快速启动功能的 FOC 的方框图，快速启动模块输出一个标志来启用或禁用转速闭环控制。设置了零基准扭矩电流，在快速启动功能运行时速度 PI 控制器输出被禁用。

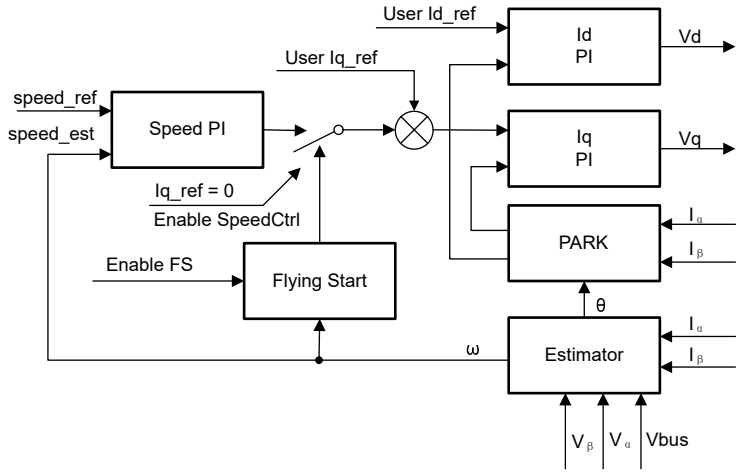


图 3-17. 快速启动控制方框图

图 3-18 展示了该参考设计中 PMSM 的无传感器 FOC（使用 eSMO 并具有快速启动功能）的整体方框图。

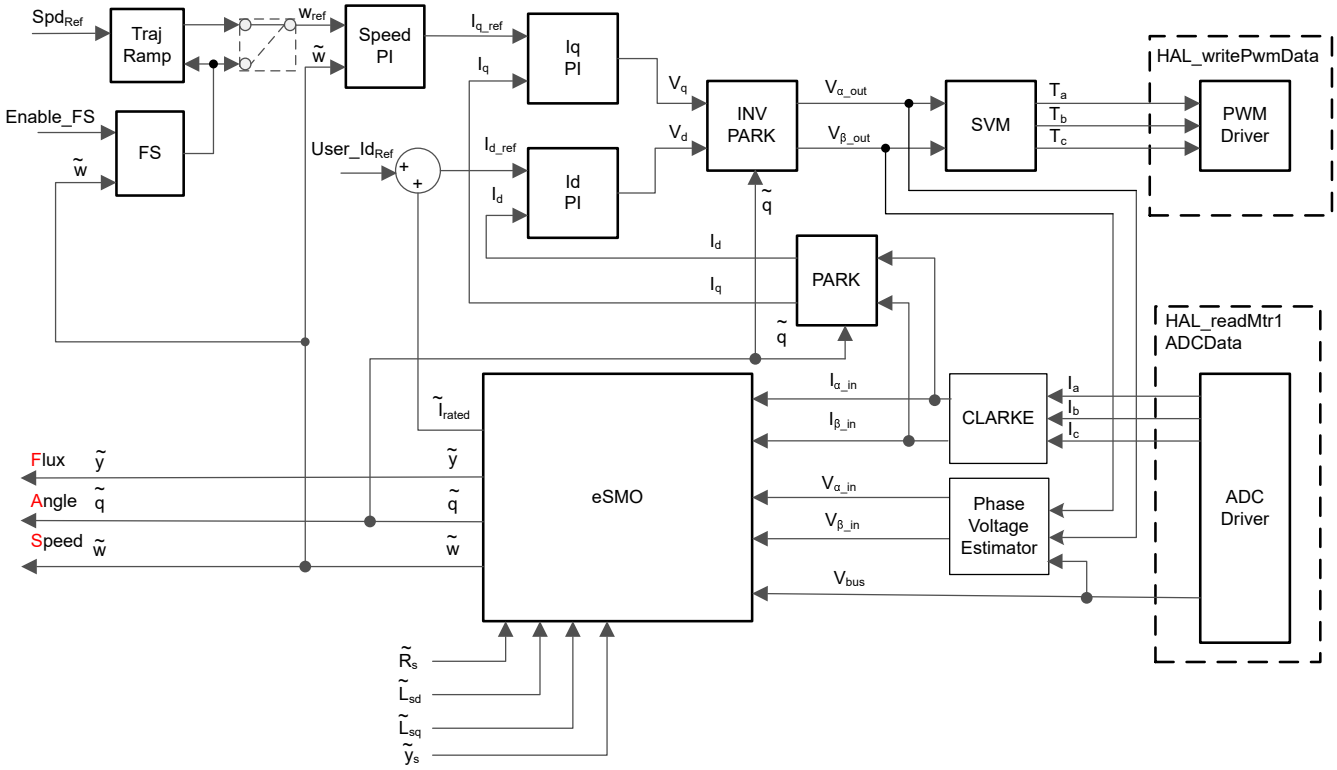


图 3-18. 使用 eSMO 并具有快速启动功能的 PMSM 的无传感器 FOC

如图 3-19 所示，模块例程禁用转速闭环控制，将基准 Iq 设置为零，并在启动期间启用 FOC 模块运行电机。在测量相电流和电压后，该例程运行 FOC 并且可以估算实际的电机转速。程序重新启用速度闭环控制，并在快速启动完成后设置转速基准值。

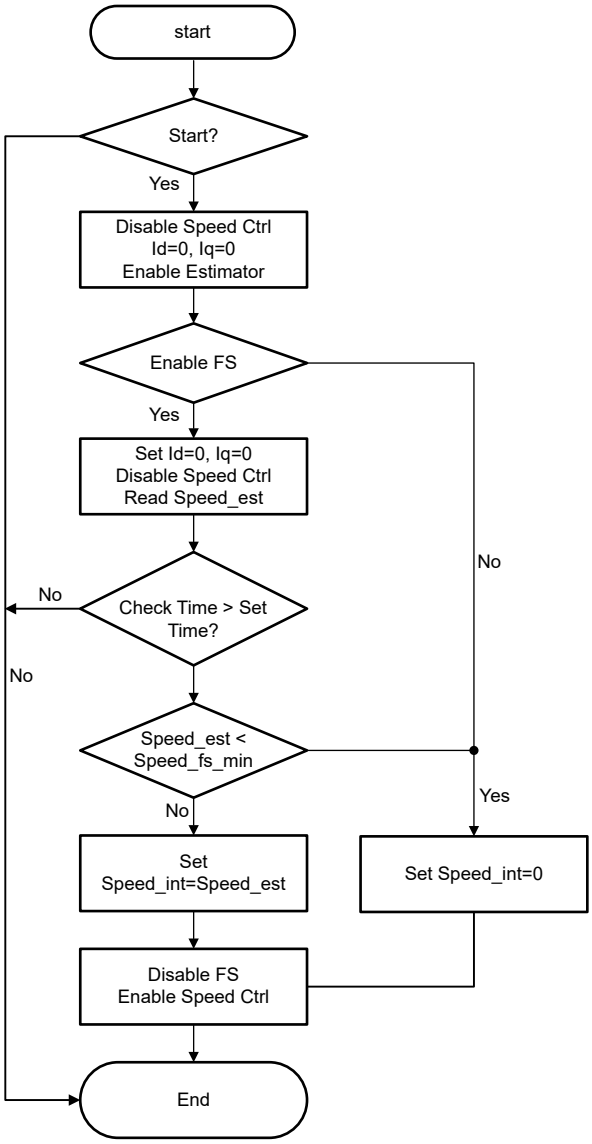


图 3-19. 快速启动模块程序流程图

4 硬件、软件、测试要求和测试结果

4.1 硬件要求

表 4-1 列出了通用电机控制工程支持的当前评估套件。

表 4-1. 受通用电机控制支持的电机驱动评估套件

电机驱动评估板		TI MCU 评估模块	电流检测拓扑	转子位置感应方法	测试电机
器件型号	说明				
BOOSTXL-3PHGANINV	12V 至 60V、3.5A 三相 GaN 逆变器	LP-AM263	三个基于采样电阻的内嵌式电机相电流检测	基于 eSMO 观测器的无传感器 FOC 基于 QEP 编码器的有传感器 FOC 基于霍尔传感器的有传感器 FOC	LVSERVOMTR（编码器和霍尔传感器已嵌入）
TMDSHVMTRINSPIN (1)	400V、10A 三相逆变器	TMDSCNCD263 与 TMDSADAP180TO100	三个低侧分流器	基于 eSMO 观测器的无传感器 FOC 基于 QEP 编码器的有传感器 FOC	HVPMSMMTR（编码器已嵌入）

(1) 如果您要使用高电压套件运行 LVSERVOMTR 等低压电机，则需要在 J1、J2、J3 和 J4 上组装跳线，从而绕过 820k 电阻器来检测相位和直流母线电压。此外，如以下代码所示，在 user_mtr1.h 中设置参数。建议不要在高压套件上运行具有高电流和低电感的低压电机。

```
// Bypass the 820k resistor for low voltage motor on this kit
#define LV_JUMPER_EN            // Bypass the 820k resistor
```

如果工程设置为使用编码器或基于霍尔传感器的 FOC，请确保以正确的顺序连接物理连接。如果电机、编码器或霍尔线的连接顺序错误，则工程将无法正常运行，可能导致电机无法工作。对于电机相线，请验证电机相位是否连接到逆变器板上的正确相位。对于随 TI 电机控制参考套件提供的电机，提供了正确的相位连接，如表 4-2 中所示。

对于编码器，请确认 A 连接到 A，B 连接到 B，I 连接到 I。对于霍尔传感器，请保持 A 连接到 A，B 连接到 B 以及 C 连接到 C。通常还需要 +5V 直流和接地连接。如果使用的霍尔传感器或编码器与表 4-2 中具体列出的传感器或编码器不同，请参阅所用霍尔传感器或编码器的用户手册，以验证是否正确连接电线。

确保为 ENC 模块的设置和配置提供了编码器每旋转一周的时隙数。这使得 ENC 模块能够正确地将编码器信号转换为角度。需要将 *user_mtr1.h* 文件中定义的 USER_MOTOR1_NUM_ENC_SLOTS 常量更新为编码器的正确值。如果此值不正确，则电机会旋转得更快或更慢，具体取决于已设置的值。请注意，该值设置为编码器上的时隙数，而不是了解正交精度后得到的计数值。

表 4-2. 用于参考套件和电机的电机相位、编码器或霍尔传感器连接

		LVSERVOMTR	HVPMSMMTR
电机相线	U	BLACK (16AWG)	红色
	V	RED (16AWG)	蓝色/黑色
	W	WHITE (16AWG)	白色
编码器	GND	BLACK (J4-1)	黑色
	+5V	RED (J4-2)	红色
	I	BROWN (J4-3)	黄色
	B	ORANGE (J4-4)	绿色
	A	BLUE (J4-1)	蓝光
霍尔传感器	GND	BLACK (J10-1)	不支持基于霍尔传感器的有传感器 FOC
	+5V	RED (J10-2)	
	A	灰白色 (J10-3)	
	B	绿色-白色 (J10-4)	
	C	GREEN (J10-5)	

立即开始使用 TI 实时控制微控制器 (MCU) 来实现电机控制。

- 第 1 步：订购所需的电机驱动评估板、TI MCU 评估模块和电机，如表 4-1 所示。
- 第 2 步：下载 MOTOR-CONTROL-SDK-AM263X 的新版本。
- 第 3 步：下载最新版本的 Code Composer Studio IDE。
- 第 4 步：按照技术文档中的说明设置硬件并运行以下各节中所述的工程。
- 第 5 步：有关您提出的任何设计问题的答案，您可以使用 TI C2000 E2E 设计支持论坛搜索现有答案或提出自己的问题。

4.2 软件要求

1. 从 Code Composer Studio (CCS) 集成开发环境 (IDE) 工具文件夹下载 Code Composer Studio 并进行安装。建议使用版本 12.6 或更高版本。有关 CCS 安装和实现的更多详细信息，请参阅 CCS 用户指南。
2. 通过 TI 提供的链接下载并安装 MOTOR-CONTROL-SDK-AM263X 软件包，并在默认文件夹中安装该 Motor Control SDK 软件。可以通过两种方法之一安装 MOTOR-CONTROL-SDK-AM263X：
   a. 通过 MOTOR-CONTROL-SDK-AM263X 下载文件夹下载软件。
   b. 转到 CCS 的“View”→“Resource Explorer”下。在 TI Resource Explorer 下，依次转到“Arm®-based microcontrollers”→“MOTOR CONTROL SDK for AM263x”，然后点击“install”按钮。
3. 安装完成后，关闭 CCS 并创建一个新的工作区以导入工程。按照以下步骤使用不同的增量构建来构建和运行该工程，如以下部分所述。

4.2.1 导入和配置工程

该工程是一个通用电机控制设计，支持 TI EVM 电机驱动器套件，可与 AM263x MCU 器件结合使用。用户可以通过设置工程的编译配置和属性来运行不同的 TI EVM 套件。在以下各节中，将 LP-AM263 与 BOOSTXL-3PHGANINV 实验结合使用，以展示如何导入和运行此套件上的示例实验。

1. 依次点击“Project”➔“Import CCS Projects...”，在 CCS 中导入工程，然后点击“Browse...”按钮选择搜索目录
   a. （位于 `<install_location>\examples\`），来选择“universal_motorcontrol_lab”文件夹。
2. 该工程可配置为在两个电机驱动器套件上运行。通过右键点击导入的工程名称并选择正确的构建配置（例如 **3phGaN_3SC**），可以选择其中一个套件，如图 4-1 所示。
3. 通过右键点击导入的工程名称来配置工程以选择工程中的支持函数，然后点击“Properties”命令为工程设置预定义符号，如图 4-2 所示。
   a. 通过在名称中删除或添加“_N”，可以激活或禁用预定义符号。例如，将“MOTOR1_FWC_N”中的“_N”删除（使其变为“MOTOR1_FWC”）可启用弱磁控制，而将“MOTOR1_FWC”符号名称更改为“MOTOR1_FWC_N”可禁用弱磁控制功能。
   b. 根据电机和硬件板，通过启用上述的相关预定义符号来选择正确的支持电机控制算法。表 4-3 展示了提供支持的算法和相关电机矩阵。
   c. 通过启用预定义符号来选择正确的支持函数，如图 4-2 所示。
4. 选择正确的目标配置文件 (.ccxml)（如图 4-4 所示），方法是右键点击文件名，在弹出菜单中选择“Set as Active Target Configuration”和“Set as Default Target Configuration”。
   a. AM263_LP.ccxml 适用于基于 LP-AM263 的硬件套件。
   b. AM263_CC.ccxml 适用于基于 TMDSCNCD263 的硬件套件。
5. 在 *user_mtr1.h* 和 *user_common.h* 文件中选择或定义正确的电机模型。这些文件位于工程浏览器窗口中的 src_board 文件夹下。取消注释与受测电机相对应的 #define，并确认其余 #define 电机仍被注释掉。确保代码中的电机参数与所连接电机的规格相匹配。
6. 按照节 4.3 中所述设置硬件套件，将电机、编码器和/或霍尔传感器连接到套件。

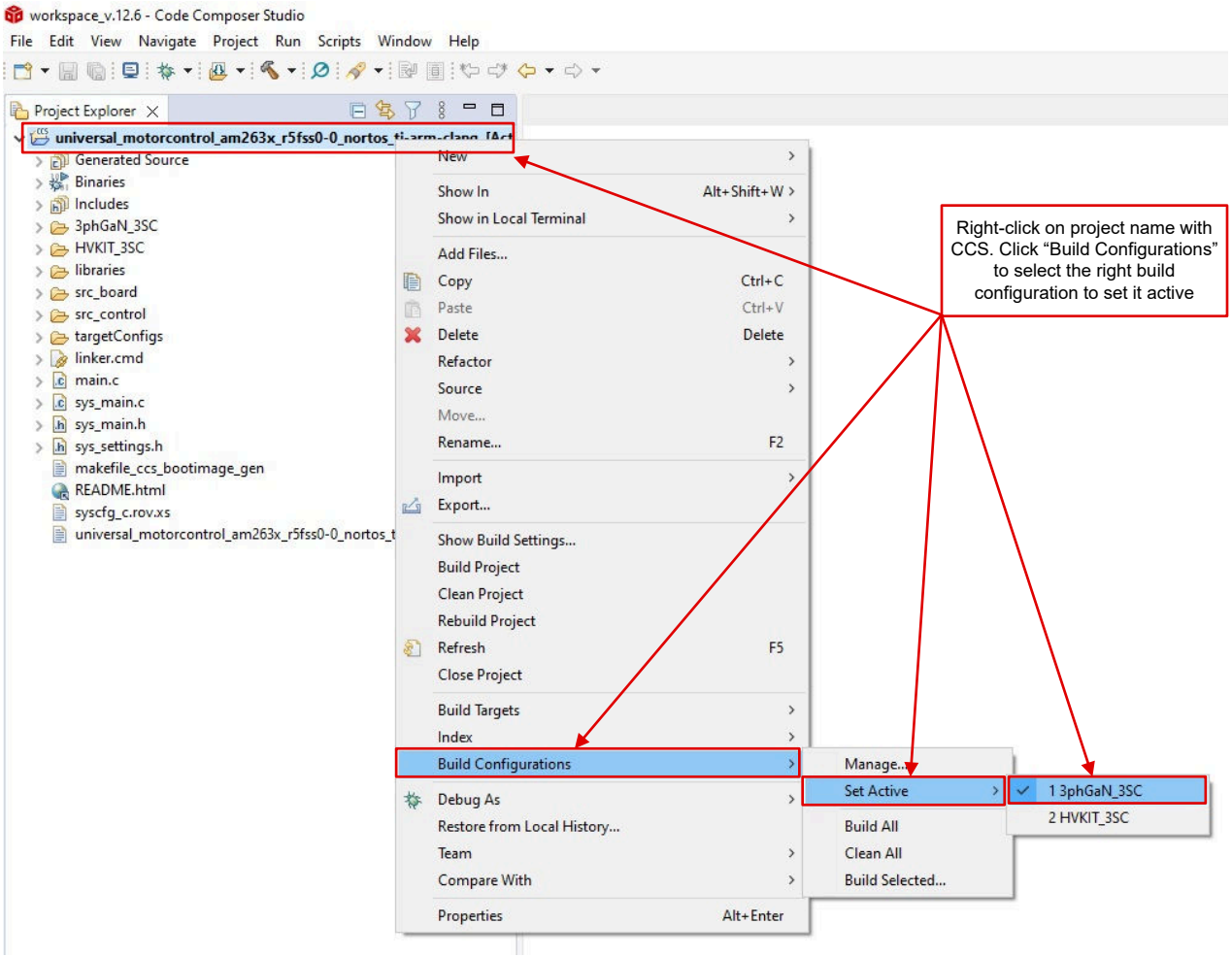


图 4-1. 在 CCS 中选择合适的构建配置

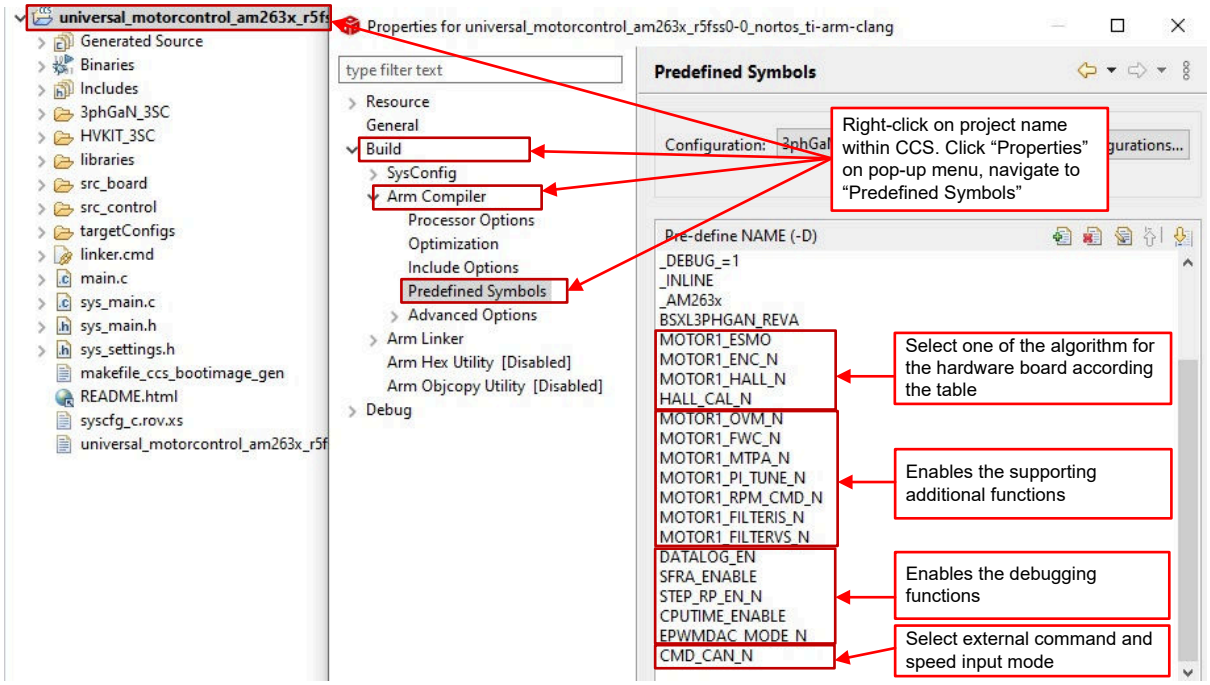


图 4-2. 在工程属性中选择所需的预定义符号

表 4-3. 通用电机控制中的支持算法、功能和电机矩阵

算法或功能	预定义符号	LaunchPad	controlCARD
		BOOSTXL-3PHGANINV	TMDSHVMTRINSPIN
基于 eSMO 的无传感器 FOC	MOTOR1_ESMO	✓，LVSERVOMTR	✓，HVPMSMMTR
基于 QEP 编码器的含传感器 FOC	MOTOR1_ENC	✓，LVSERVOMTR	✓，HVPMSMMTR
基于霍尔传感器的含传感器 FOC	MOTOR1_HALL HALL_CAL	✓，LVSERVOMTR	✖
使用图形工具时的数据日志	DATALOG_EN	✓	✓
PWMDAC	EPWMDAC_MODE	✖	✓
SFRA 工具	SFRA_ENABLE	✓	✓
使用图形工具时的阶跃响应	STEP_RP_EN	✓	✓

4.2.2 工程结构

图 4-3 展示了工程的总体结构。器件外设配置基于 *TI SysConfig*。如果用户需要将参考设计软件迁移到定制板或不同器件上，则只需更改 *hal.c* 和 *hal.h* 中的代码和定义以及 *user_mtr1.h* 中的参数。

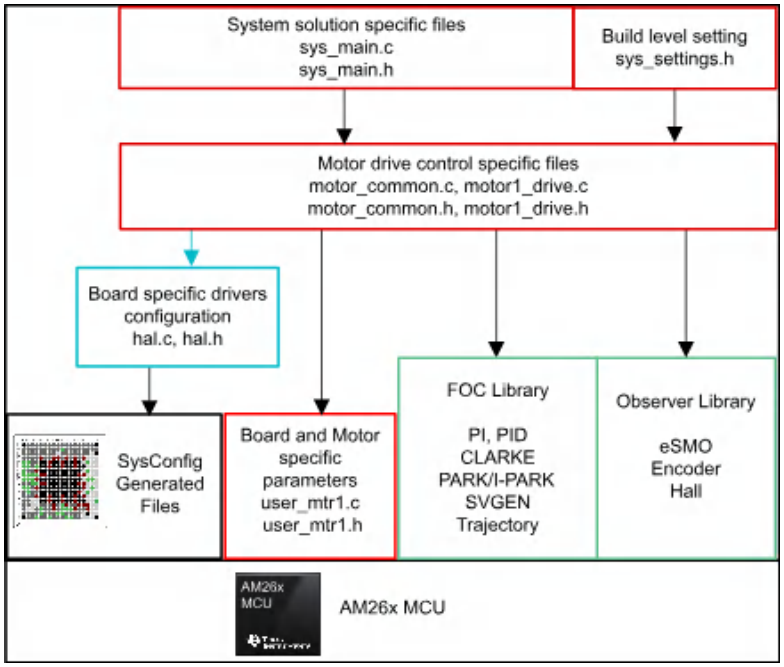


图 4-3. 工程结构概览

将工程导入 CCS 后，CCS 内将显示工程浏览器，如图 4-4 所示。

transforms 文件夹包含典型的 FOC 模块，包括 Park、Clark、逆向 Park 和 SVGEN，它们是电机驱动 ISR 的一部分，独立于特定器件或电路板。

libraries 文件夹包括估算器库和其他非特定于任何特定器件或电路板的库。

src_control 文件夹包含电机驱动控制文件，这些文件在中断服务例程和后台任务中调用电机控制核心算法函数。

文件夹 *src_sys* 包含为系统控制保留的一些文件，这些文件独立于特定的器件或电路板。用户可以添加用于系统控制、通信等功能的代码。

特定于电路板和特定于电机的文件位于 *src_board* 文件夹中。这些文件包含特定于器件的驱动程序，用于运行设计。如果用户希望为自己的电路板迁移工程或迁移到其他器件，则只需根据电路板的器件外设使用情况更改 *hal.c*、*hal.h*、*xxx.syscfg* 和 *user_mtr1.h* 文件。

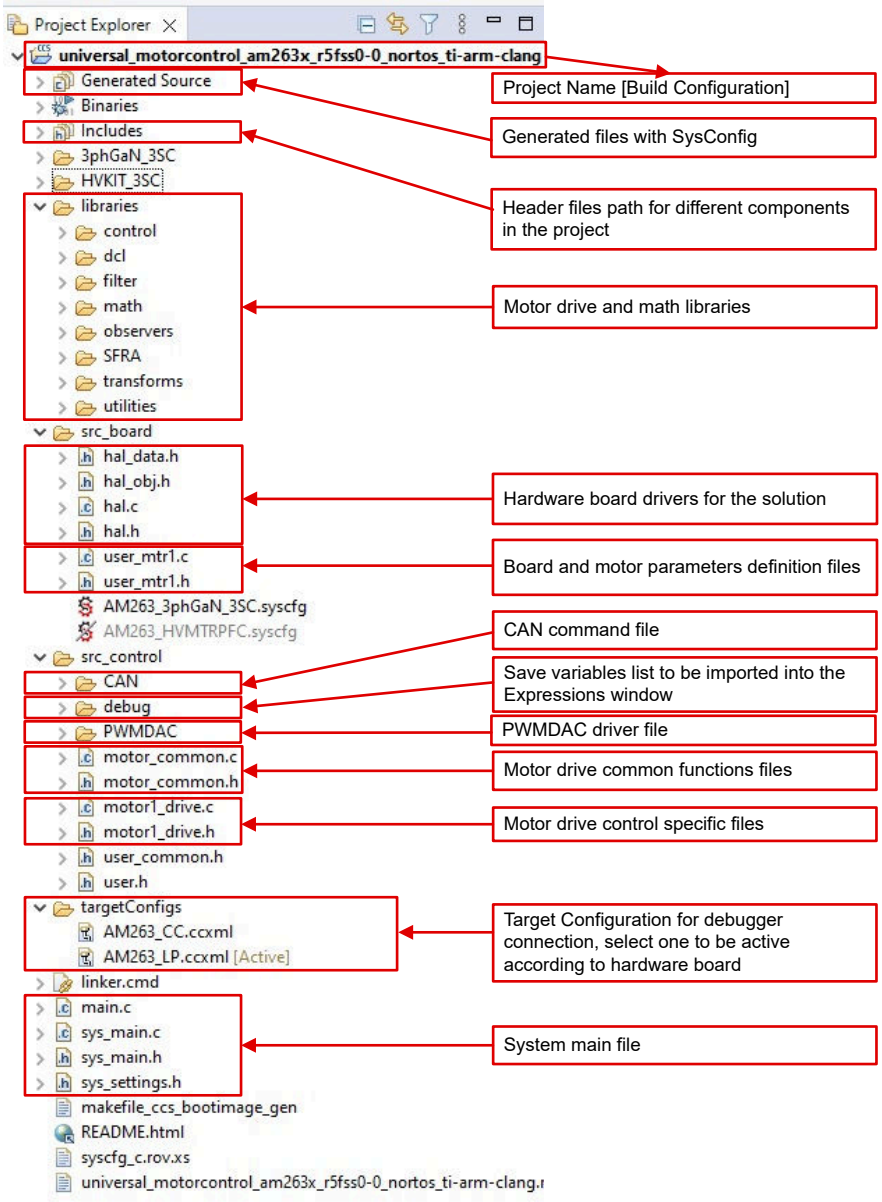


图 4-4. 通用电机控制工程的工程浏览器视图

4.2.3 实验室软件概述

图 4-5 展示了固件的工程软件流程图，其中包括一个用于实时电机控制的 ISR、一个主循环用于在后台循环中更新电机控制参数。ISR 由 ADC 转换结束 (EOC) 触发。

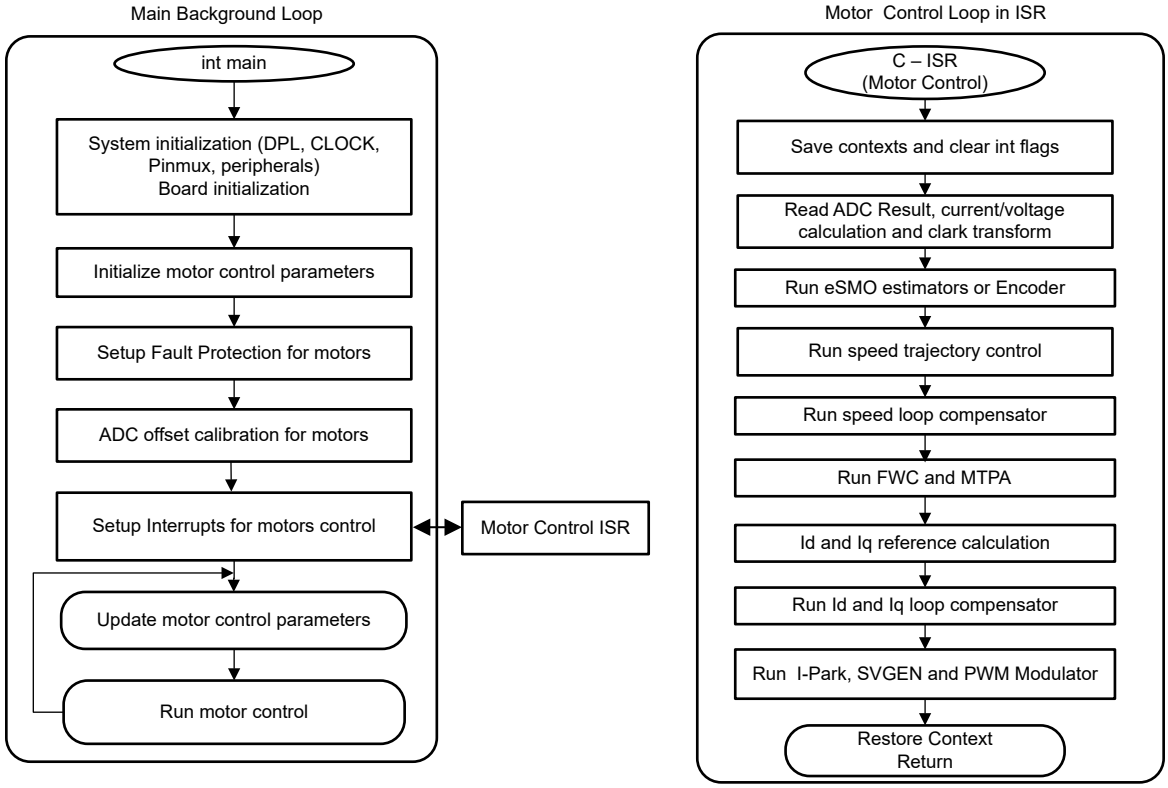


图 4-5. 工程软件流程图

为了简化系统开发和设计，将该软件组织为四个增量构建，这使得学习和熟悉电路板和软件变得更加容易。这个方法对也适用于调试和测试电路板。

表 4-4 列出了要在该工程中使用的框架模块。

表 4-4. 在工程中使用电机控制模块

模块名称	说明	算法
ANGLE_GEN_run	用于开环运行的斜坡角度发生器	eSMO、ENC、霍尔
CLARKE_run	针对电流或电压的 Clarke 变换	eSMO、ENC、霍尔
collectRMSData、calculateRMSData	收集采样值以计算相电流和电压的 RMS 值	eSMO、ENC、霍尔
DATALOG_update	存储实时值，以便使用图形工具显示	所有算法
ENC_run	根据编码器计算转子角度	ENC
ESMO_run	增强滑模观测器 (eSMO) 以实现无传感器 FOC	eSMO
HAL_readMtr1ADCData	以浮点格式返回 ADC 转换值	所有算法
HAL_writePWMDACData	将软件变量转换为 PWM 信号	所有算法
HAL_writePWMData	用于电机的 PWM 驱动	所有算法
HALL_run	根据霍尔传感器计算转子角度和速度	HALL
IPARK_run	Park 逆变换	eSMO、ENC、霍尔
PARK_run	Park 变换	eSMO、ENC、霍尔
PI_run	针对电流和速度的 PI 稳压器	所有算法
PI_run_series	运行串联形式的 PI 控制器	SFRA、MPTA
SPDCALC_run	基于来自编码器信号的角度进行速度测量	ENC
SPDFR_run	基于来自观测器的角度测量速度	eSMO
SVGEN_runMin	具有正交控制的空间矢量 PWM	eSMO、ENC、霍尔
TRAJ_run	用于设置速度基准的轨迹	所有算法
VS_FREQ_run	为 2 级的 Vd 和 Vq 计算生成具有 v/f 曲线的矢量电压。 这可以根据特定电机手动完成。	eSMO、ENC、霍尔

表 4-5 总结了在每个递增系统构建中测试的模块。

表 4-5. 每个增量构建使用的电机控制模块

软件模块	DMC_LEVEL_1	DMC_LEVEL_2	DMC_LEVEL_3	DMC_LEVEL_4
	50% PWM 占空比，验证 ADC 失调电压校准、PWM 输出和相移	开环控制，用于验证电机电流和电压检测信号	闭合电流环路，用于在电路板上验证电流检测和使用 PID 进行电流控制	采用估算器/观测器的闭环运行
HAL_readMtr1ADCData	√ √	√	√	√
HAL_writePWMData	√ √	√	√	√
ANGLE_GEN_run		√ √	√	√ (eSMO、ENC、霍尔)*
VS_FREQ_run		√ √		
CLARKE_run		√	√	√
TRAJ_run		√ √	√	√ √
ESMO_run		√ (eSMO)*	√ (eSMO)*	√ √ (eSMO)*
SPDFR_run		√ (eSMO)*	√ (eSMO)*	√ √ (eSMO)*
ENC_run		√ (ENC)*	√ (ENC)*	√ √ (ENC)*
SPDCALC_run		√ (ENC)*	√ (ENC)*	√ √ (ENC)*
HALL_run		√ (HALL)*	√ (HALL)*	√ √ (HALL)*
PARK_run		√	√	√
PI_run (Id)			√ √	√
PI_run (Iq)			√ √	√
PI_run (速度)				√ √
IPARK_run		√ √	√	√
SVGEN_runMin		√ √	√	√
HAL_writePWMDACData		√**	√**	√**
DATALOG_update		√	√	√

1. √ 表示使用此模块。√ √ 表示此模块正在测试中。
2. √ (eSMO)* 表示此模块仅供 eSMO 使用。√ (ENC)* 表示此模块仅由 ENC 使用。√ (HALL)* 表示此模块仅由 HALL 使用。
3. √** 表示此模块受某些硬件套件的支持，如表 4-1 所示。

通用工程可以单独使用其中一种 FOC 算法进行电机控制，或同时使用两种 eSMO 和编码器 FOC 算法。如果在工程中实现了两种算法，则可以快速平滑地切换正在使用的估算器。

4.3 测试设置

本节介绍了在将电机驱动器评估板与 TI 开发工具结合时如何设置用于电机控制的硬件板。以下各节显示了不同电机驱动器评估板上的详细操作过程。

4.3.1 LP-AM263 设置

LP-AM263 是一款适用于基于 TI Arm® 的实时微控制器的低成本开发板。该 LaunchPad 套件可提供额外引脚用于开发，并支持连接两个 BoosterPack™ 插件模块。

- 有关 LP-AM263 的更多详细信息，请参阅 LP-AM263 LaunchPad 用户指南。
- 确保按图 4-6 中所示设置 LP-AM263 上的引导开关。
  - 对于 **QSPI_D0 (SOP0)**，将开关置于**左侧**（逻辑高电平）。
  - 对于 **QSPI_D1 (SOP1)**，将开关置于**左侧**（逻辑高电平）。
  - 对于 **SPI0_CLK_pad (SOP2)**，将开关置于**右侧**（逻辑低电平）。
  - 对于 **SPI0_D0_pad (SOP3)**，将开关置于**左侧**（逻辑高电平）。
- 确保在 AM263x 片上 LDO 上设置 LP-AM263 上的 DAC VREF 开关 (S1)，以实现 CMPSS 正常运行。

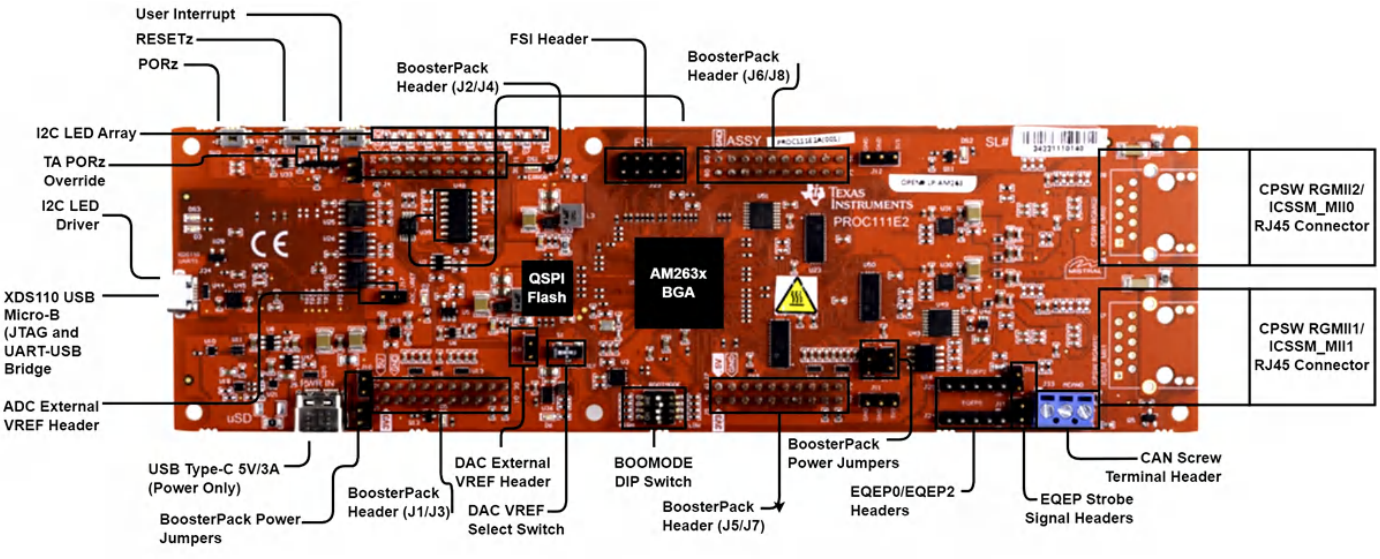


图 4-6. LP-AM263 LaunchPad™ 电路板概述和开关设置

4.3.2 BOOSTXL-3PHGANINV 设置

BOOSTXL-3PHGANINV 评估模块采用 48V/10A 三相 GaN 逆变器，配备基于分流器的精密直列式相电流检测，从而对精密驱动器（例如，伺服驱动器）进行精准控制。该模块还具有独立的直流母线和三相电压检测功能，因此采用 TI LaunchPad™ 的 BLDC/PMSM 控制板专为无传感器 FOC 算法而设计。

- TI.com 内的 BOOSTXL-3PHGANINV 页面上提供了硬件文件和更多详细信息。
- 有关 BOOSTXL-3PHGANINV 的更多详细信息，请参阅相应的用户指南。
- 确保按照所述完成以下各项，然后将 BOOSTXL-3PHGANINV 连接到 LP-AM263 的 J6/J8 和 J5/J7，如图 4-7 所示。
- 将电机、编码器和霍尔传感器连接到表 4-2 中所述的 BOOSTXL-3PHGANINV 和 LP-AM263，如图 4-7 所示。
- 将 24V 电源电压从电池或直流电压源连接到电源引脚。按照节 4.4 中的操作说明打开电源。

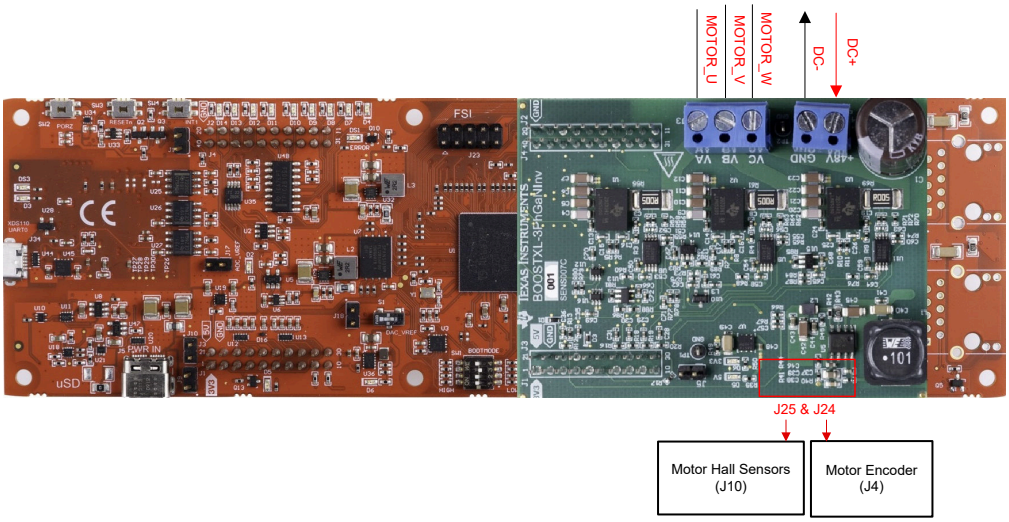


图 4-7. LP-AM263 连接到 BOOSTXL-3PHGANINV

4.3.3 TMDSCNCD263 设置

TMDSCNCD263 是一款适用于基于 TI Arm® 的 MCU 系列 AM26x 器件的评估和开发板。TMDSCNCD263 附带 HSEC180（180 引脚高速边缘连接器），可用于基于 DIMM 的现有 100 引脚 TMDSHVMTRINSPIN（具有 TMDSADAP180TO100 适配器）

- 有关 TMDSCNCD263 的更多详细信息，请参阅 AM263x Sitara 控制卡硬件用户指南。
- 确保按照图 4-8 中所述设置 TMDSCNCD263 上的引导开关。
  - SW3.1、SW3.2 和 SW3.4 位置开关位于**左侧**，SW3.2 位于**右侧**，用于在四路读取 UART 回退模式下使用卡上 XDS110 仿真器
- 确保 TMDSCNCD263 上的 DAC VREF 开关 (SW6) 切换到**向上**，以在 AM263x 片上 LDO 上设置基准电压，从而实现正确的 CMPSS 运行。

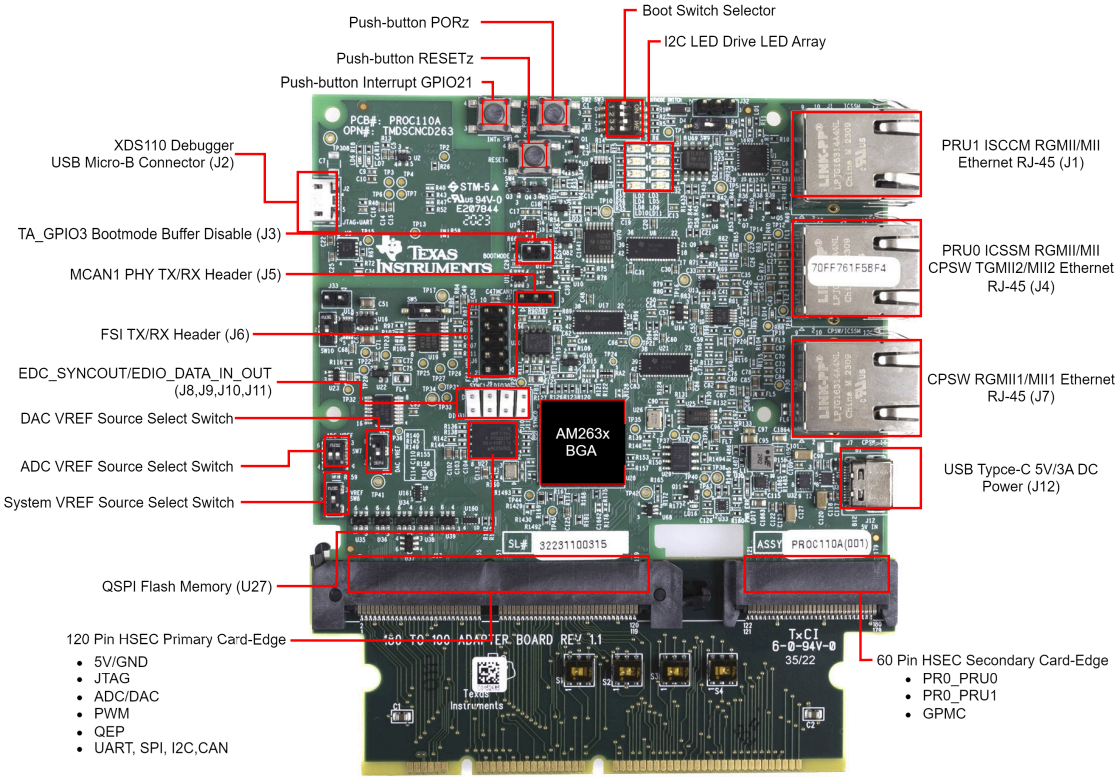


图 4-8. TMDSCNCD263 controlCARD 和开关设置

4.3.4 TMDSADAP180TO100 设置

TMDSADAP180TO100 适配器允许将 180 引脚 TI controlCARD 和现有 100 引脚基于 DIMM 的评估工具结合使用。TMDSCNCD263 controlCARD 需要在 TMDSHVMTRINSPIN 上使用 TMDSADAP180TO100。

- 硬件文件位于 C2000Ware 的 `<install_location>\boards\controlCARDs\TMDSADAP180TO100` 文件夹。
- 确保开关 TMDSADAP180TO100 按照图 4-9 中所述或所示进行设置。
  - S1、S2 和 S3 开关需要放在**右侧**，而 S4 开关需要放在**左侧**。

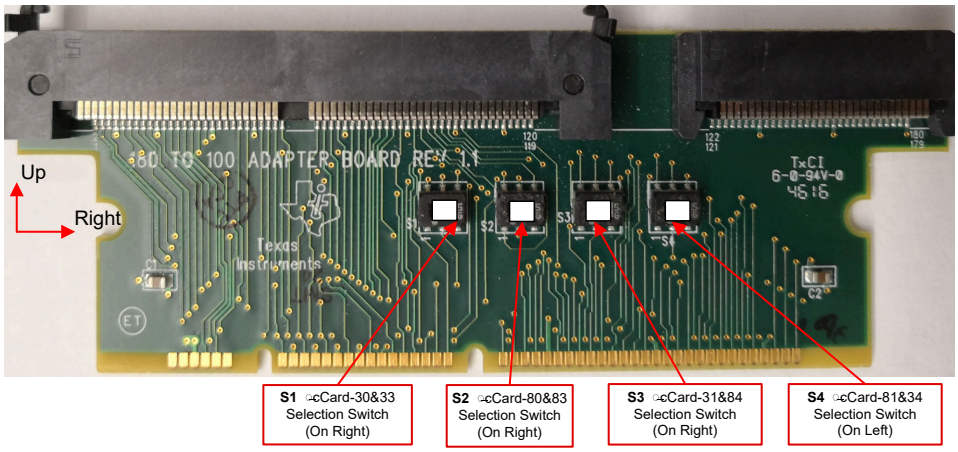


图 4-9. TMDSADAP180TO100 适配器和开关设置

4.3.5 TMDSHVMTRINSPIN 设置

警告

- 此评估模块 (EVM) 仅限在实验室环境中工作，并且 TI 不将其视为适合常规消费品用途的最终产品成品。
- 这个 EVM 只能由具有资质的工程师和技术人员使用，这些专业人员熟悉与处理高压电气和机械部件、系统和子系统相关的风险。
- 此 EVM 中存在电压和电流时，如果处理不当，会引发电击、火灾和/或人身伤害。使用设备时必须小心并采取适当的防护措施以避免人身伤害或财产损失。
- 由于存在高电压，在使用 EVM 电子元件时始终要小心。主电源断开后，直流母线电容器会长时间保持充电状态。
- 该 EVM 可接受来自交流电源/壁式电源的电源，仅使用来自壁式电源的火线和中性线，保护性接地未连接（悬空）。电源接地相对于保护性接地端浮动，所有接地平面都是相同的。因此，在将示波器和其他测试设备连接到电路板之前，必须小心谨慎并满足适当的隔离要求。在将接地设备连接到 EVM 时，必须使用隔离变压器。
- 主板上的功率级具有单独的额定值。用户有责任确保在将这些电源块连接在一起并为主板供电前，已完全理解这些额定值（例如，电压、电流和功率等级）并遵守这些额定值。通电后，不得触碰 EVM 以及与 EVM 相连的元件。

TMDSHVMTRINSPIN 是一款基于 DIMM100 controlCARD 的主板评估模块，展示了最常见类型高电压三相电机控制，这些电机包括交流感应 (ACI) 电机、无刷直流 (BLDC) 电机和永磁同步电机 (PMSM)。高电压电机控制套件具有独立的直流母线和三相电压检测功能，因此采用 TI controlCARD™ 的 BLDC/PMSM 控制板是与无传感器 FOC 算法配合使用的理想选择。

- 硬件文件位于 C2000WARE-MOTORCONTROL-SDK 的 `<install_location>\solutions\tmdshvmtrinspin\hardware` 文件夹中。

本节介绍了使用通过 MotorControl SDK 提供的软件运行 TMDSHVMTRINSPIN 所需的步骤。该套件随附跳线和开关设置、位置正确，可用于连接 controlCARD。确保这些设置在电路板上是有效的，如下所述，然后将带有 TMDSADAP180TO100 适配器的 controlCARD 插入到 TMDSHVMTRINSPIN 电路板中，如图 4-10 所示。

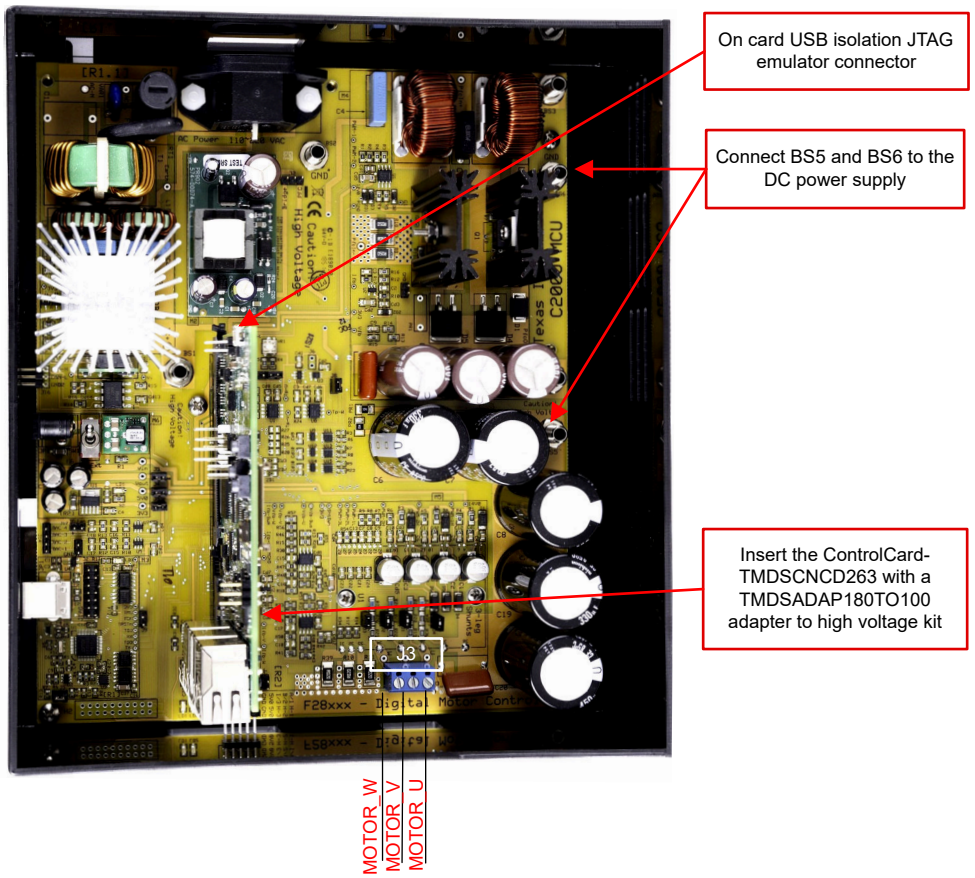


图 4-10. TMDSHVMTRINSPIN 通过 TMDSADAP180TO100 连接到 TMDSCNCD263

- 确保没有任何设备连接至电路板，并且未对电路板施加任何电源。
- 将带 TMDSADAP180TO100 适配器的控制卡插入 [Main]-J1 controlCARD 连接器（如果尚未组装）。
- 确保以下跳线和连接器设置正确实现，如图 4-11 所示。
  - [Main]-J3、J4、J5 和 J8 已组装。
  - 未组装 [Main]-J9 和 [M3]-J5 以便使用具有板载仿真的 controlCARD 来禁用 HVKIT 上的 XDS100。
  - [MAIN]-J7 安装在引脚 2-3（距离 DIMM 100 插槽最远的引脚）之间。
  - 在 > 150W 的负载下操作电机时，确保套件附带的 DC 风扇连接到 DC 风扇跳线 [Main]-J17。
- 获得直流母线电源的两个选项如下，建议使用外部 15V 直流电源。
  - 如果使用来自外部 15VDC 电源的 +15V 电压，则不会组装 [Main]-J2。确认 [M6]-SW1 处于“Off”位置，将 15V 直流电源连接到 [M6]-JP1。
  - 如果使用来自辅助电源模块的 +15V 电源，则 [Main]-J2 在电桥和中间引脚之间组装了一个跳线。
- 打开 [M6]-SW1。现在，[M6]-LD1 接通。请注意，控制卡 LED 也点亮，这表示控制卡正在由电路板供电。
- 将电机、编码器和霍尔传感器连接到表 4-2 中所述的套件，如图 4-11 中所示。
- 将电源电压从交流或直流电压源连接到电源引脚。在节 4.4 中收到指示时接通电源，否则保持断开。

表 4-6 展示了电路板上可用的各种连接。电路板上这些连接的位置如图 4-11 所示。

表 4-6. 关键跳线，连接器说明

[Main]-P1	交流输入连接器 (110V – 220VAC)
[Main]-TB3	用于连接电机的端子块
[Main]-BS1	用于从交流整流器输出的香蕉插孔
[Main]-BS2，BS6	针对接地 (GND) 连接的香蕉插孔
[Main]-BS3	用于为 PFC 级连接输入电压的香蕉插孔，这通常是来自 [Main]-BS1 连接器的交流电压整流。
[Main]-BS4	香蕉插孔用于将负载连接到 PFC 级的输出，当使用 PFC+ 电机工程时，PFC 级的输出连接到逆变器总线的输入，例如 [Main]-BS5

表 4-6. 关键跳线，连接器说明（续）

[Main]-BS5	用于为逆变器提供直流母线电压输入的香蕉插孔
[Main]-J2	辅助电源模块输入电压选择跳线， • 当跳线连接到桥位置时，辅助电源模块从交流整流器电桥输出提供电源。 • 当跳线连接到 PFC 位置时，辅助电源模块从 PFC 级的输出端提供电源。
[Main]-J3，J4，J5	跳线 J3、J4 和 J5 分别用于为电路板从 15V 直流电源提供 15V、5V 和 3.3V 电源。
[Main]-J7	J7 用于选择过流保护阈值源
[Main]-J8	J8 用于启用/禁用 IPM 过流保护
[Main]-J9	JTAG TRSTn 将跳线断开，组装跳线将启用 JTAG 到微控制器的连接。当不需要 JTAG 连接时，例如从闪存引导时，需要拆下跳线。
[Main]-J14	PWMDAC 输出：提供了由一阶低通滤波器所连接的 PWM 引起的电压输出。引脚 1、2、3 和 4 分别被连接至低通滤波的 PWM 输出引脚以在示波器上观察系统变量。
[Main]-J16	隔离式 CAN 总线连接器
[Main]-J17	为连接到 IPM 散热器的直流风扇（随电路板提供）供电的连接器。
[Main]-H1	QEP 连接器：与 0V 至 5V QEP 传感器连接，以收集有关电机速度和位置的信息。 电容/霍尔效应传感器连接器：与 0V 至 5V 传感器连接，以收集有关电机速度和位置的信息。
[M1]-F1	交流输入保险丝
[M3]-JP1	针对板载仿真的 USB 连接
[M3]-J2	外部 JTAG 接口：这个连接器提供到 JTAG 仿真引脚的访问权。如果需要外部仿真，将一个跳线放置在 [M3]-J5 并将仿真器连接到主板。若要为仿真逻辑供电，USB 连接器仍需要连接到 [M3]-JP1。
[M3]-J5	板载仿真禁用跳线：在此处放置一个跳线来禁用板载仿真器并提供到外部接口的访问。

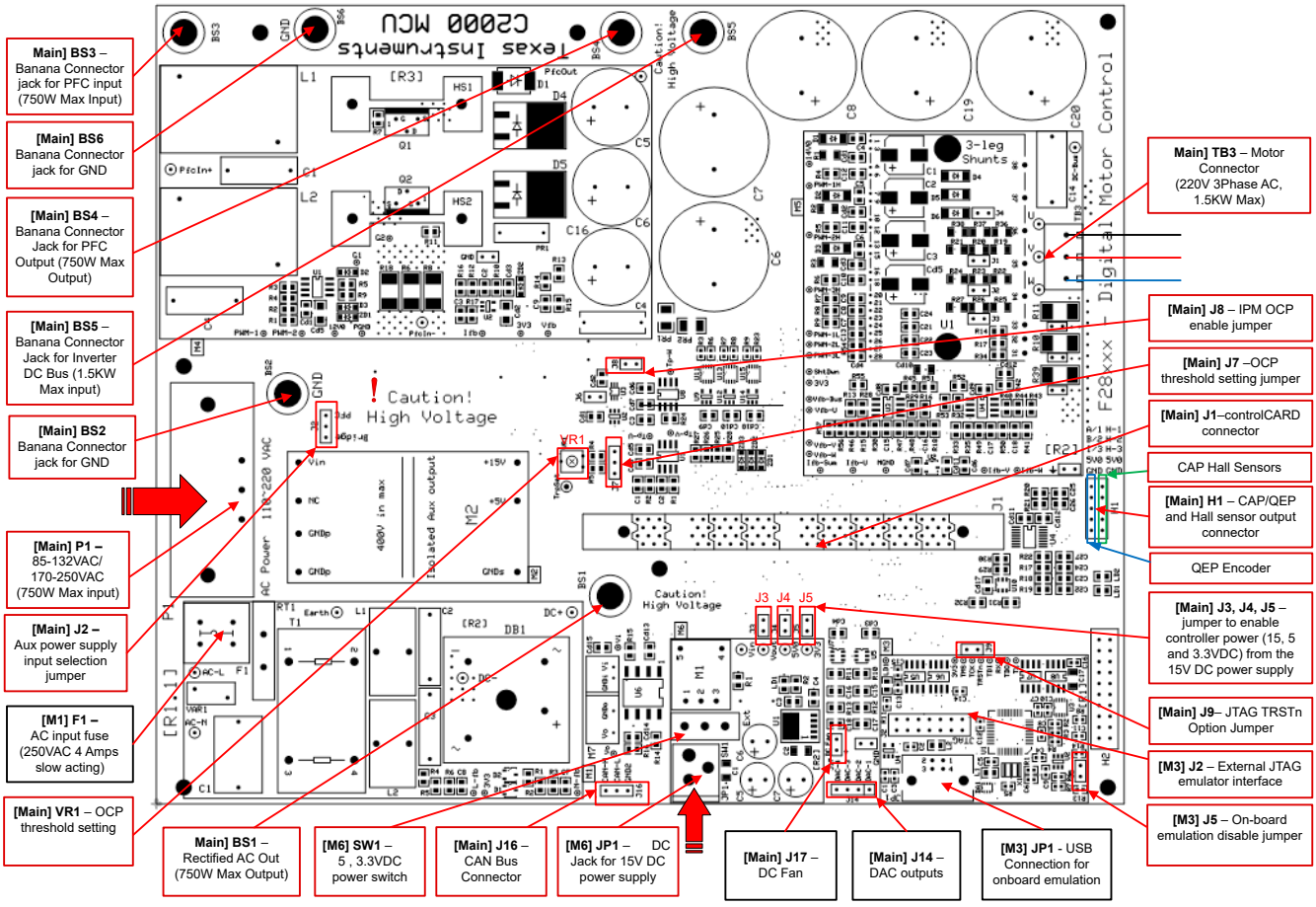


图 4-11. TMDSHVMTRINSPIN 套件跳线和连接器图

4.4 测试结果

此系统在多个阶段逐步测试和验证，这样最终系统才可以安心运转。要选择特定的构建选项，请在 *sys_settings.h* 文件中将 DMC_BUILDLEVEL 定义的值更改为所需的 DMC_LEVEL_X 选项。选中构建选项后，右键点击工程名称并点击 *Rebuild Project* 编译工程。

4.4.1 1 级递增构建

此构建级别的目标：

- 使用 HAL 对象为电机驱动器硬件初始化 MCU 的外设。
- 验证 PWM 和 ADC 驱动器模块
- 确认 ADC 偏移验证
- 熟悉 CCS 的操作。有关 CCS 的更多详细信息，请参阅 CCS 用户指南。

在该构建级别中，电路板以开环模式执行（采用固定 PWM 占空比）。占空比设置为 50%。该构建级别会验证来自功率级的反馈值检测以及 PWM 栅极驱动器的运行，并确保没有硬件问题。此外，可以在该构建级别中执行输入和输出电压检测校准。在此过程中，电机必须保持断开。图 4-12 展示了该构建级别的软件方框图。

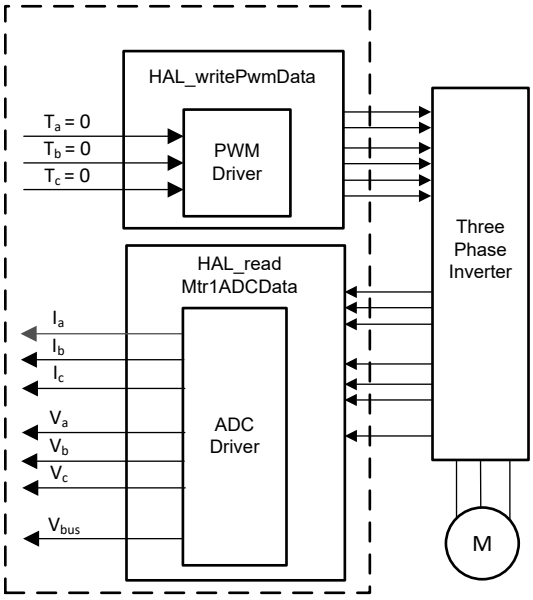


图 4-12. 构建级别 1 软件方框图 - 偏移验证

4.4.1.1 构建和加载工程

1. 按照节 4.3 中所述设置电机驱动器硬件板和 TI LaunchPad 或 controlCARD，但在此构建级别中，电机不需要连接到电机驱动器板。
2. 将 USB 电缆从计算机连接到 TI LaunchPad 或 controlCARD 上的板载 USB 连接器，以启用到 MCU 的隔离 JTAG 仿真。
3. 按照节 4.3 中所述，通过向总线电压输入端子施加适当的电压来为电机驱动器板供电。
4. 将通用电机控制工程导入 CCS 并按照节 4.2.1 中所述选择正确的构建配置。打开 *sys_settings.h* 文件，将 DMC_BUILDLEVEL 设置为 DMC_LEVEL_1。这可以确保工程被配置为运行第一个增量构建。
5. 在 Project Explorer 窗口中，通过右键点击所需的目标配置文件名并选择 *Set as Active Target Configuration*，确保将正确的目标配置文件设置为 Active。建议也通过右键点击文件名并选择 *Set as Default Target Configuration* 来将所需目标配置文件设置为默认值。之所以这样做，一个原因是没有用于显示哪个文件处于活动状态的可见指示符，但如果将文件设置为默认值，则 [default] 指示符会出现在工程浏览器窗口中文件名的旁边。将文件设置为默认值也可能导致默认情况下使用该文件，除非另一个配置文件专门设置为 Active。您也可以通过以下方法将目标配置链接到工作区中的工程：前往 *View > Target Configurations*，右键点击“Target Configurations”视图中的目标配置名称并选择 *Link to Project*。

6. 右键点击工程名称，然后点击 *Rebuild Project*。观察“Console”窗口。“Console”窗口中会显示工程中的所有错误。
7. 成功完成构建后，点击“Debug”按钮或依次点击 *Run* → *Debug*。现在，IDE 会自动连接至目标，将输出文件载入到器件内并换到调试视图。右上角会显示“CCS Debug”图标，该图标表明用户现在处于“Debug Perspective”视图中。程序需要在 main() 的开始处暂停。

4.4.1.2 设置调试环境窗口

在调试代码时观察局部和全局变量是标准调试做法。在 CCS 中有多种不同的方法来实现这一做法，例如存储器视图和监视视图。此外，CCS 能够制作时域（和频域）图。该功能允许用户使用图形工具查看波形。有关如何设置和配置图形工具的信息，请参阅节 4.5.1。有关设置表达式窗口的信息，请参阅以下说明。

1. 设置监视窗口：点击菜单栏上的 *View* → *Expressions*，打开“Expressions”监视窗口。在“Expressions”窗口中点击 *Add new expression*，输入变量的名称，然后按 Enter，即可将变量添加到“Expressions”窗口。显示变量值所用的数字格式基于声明变量值时与变量关联的数字格式。通过右键点击变量，导航至 *Number Format* 并选择所需的格式，可以为特定变量更改所需的数字格式。
2. 或者，可以通过右键点击“Expressions”窗口并点击“Import”将一组变量导入到“Expressions”窗口中，然后浏览至工程目录（`<workspace>\universal_motorcontrol_am263x_r5fss0-0_nortos_ti-arm-clang\src_control\debug\`），选择 *universal_motor_control_level1.txt* 文件，然后点击“OK”以导入图 4-13 中所示的变量。

备注

此时主代码中的某些变量尚未初始化，可能包含一些无用的值。

---

3. 注意：结构变量 *motorVars_M1* 引用了大多数与控制电机驱动相关的变量。
4. 点击“Expressions Window”选项卡右上角的“Continuous Refresh”按钮，启用微控制器的数据定期捕捉功能。通过点击 *View Menu* 按钮（“Expressions”窗口右上角的 3 个点），您可以选择 *Continuous Refresh Interval* 并编辑“Expressions”窗口的刷新率。请注意，选择过短的刷新间隔可能会影响性能。

4.4.1.3 运行代码

1. 通过取消选中 *Tools > Advanced Features* 中的 *Data Cache Enabled* 来禁用数据缓存。
2. 通过按“Resume”按钮来运行工程，或点击“Debug”选项卡中的 *Run → Resume*。
3. 现在工程应该运行，而图和监视窗口中的值应该持续更新。
4. 在监视窗口中看到 *systemVars.flagEnableSystem* 自动设置为 1 后，在“Expressions”窗口中，将 *motorVars_M1.flagEnableRunAndIdentify* 变量设置为 1。
5. 工程现在会运行，在使用该工程时图和表达式窗口中的值应不断更新，如图 4-13 所示。您可以根据您自己的偏好来调整窗口大小。
6. 在监视视图中，如果没有故障，变量 *motorVars_M1.flagRunIdentAndOnLine* 会自动设置为 1。*ISRCount* 会继续增加。
7. 检查电机驱动器板的校准偏移。电机相电流检测值的偏移值大约等于 ADC 满量程电流的一半，如图 4-13 中所示。
8. 如果使用图形工具，图中显示的变量是 u 相、v 相和 w 相的 FOC 角度和相电流。
9. 展开和检查 *MotorVars_M1.faultMtrPrev.bit* 结构，以确保未设置故障标志。
10. 使用示波器探测用于电机驱动控制的 PWM 输出。在此构建级别中，三个 PWM 的占空比设置为 50%。预期的 PWM 输出波形如图 4-14 所示。PWM 开关频率与在 *user_mtr1.h* 文件中为 USER_M1_PWM_FREQ_kHz 定义设置的值相同。
11. 将变量 *motorVars_M1.flagEnableRunAndIdentify* 变量设为 0 以停用 PWM。
12. 如果之前的任何步骤产生意外结果，则需要进行额外的调试。需要检查的几个事项：
    a. 确保使用的电机驱动器板与构建配置中选择的板相同。
    b. 确保设置了正确的预定义。
    c. 确保按节 4.3 中所述在 Lunchpad/ControlCARD 上正确配置开关。
13. 完成上述步骤后，现在可以停止控制器并终止调试连接。通过首先点击工具栏上的“Halt”按钮，或依次点击 *Target → Halt* 来完全停止控制器。最后，通过点击该按钮或依次点击 *Run → Reset*→CPU Reset 来复位控制器。
14. 通过点击“Terminate Debug Session”按钮或点击 *Run → Terminate* 来关闭 CCS 调试会话。这将暂停程序并从 MCU 上断开 Code Composer Studio。
15. 无需在用户每次更改或再次运行代码时终止调试会话。但可以遵循以下流程。重新构建工程后，按下该按钮或依次点击 *Run → Reset*→CPU Reset，然后按“Restart”按钮或点击 *Run → Restart*。如果目标器件或配置发生更改，则必须在关闭 CCS 之前终止工程。

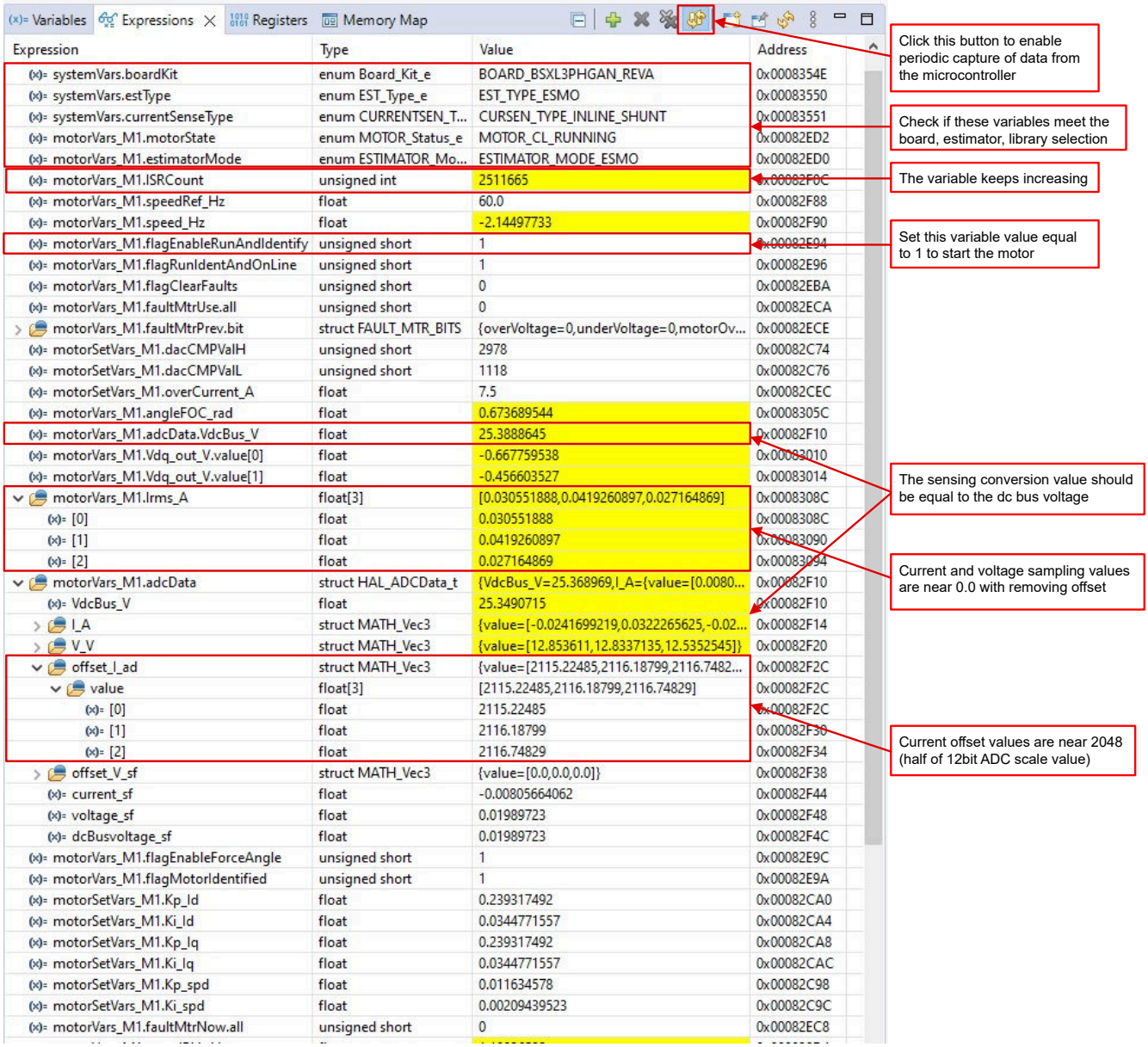


图 4-13. 构建级别 1：表达式窗口中的变量

图 4-14 展示了在栅极驱动输入端具有死区输出的 PWM。

图 4-14. 构建级别 1：PWM 输出波形

4.4.2 2 级递增构建

了解该构建级别中的目标：

- 实现简单的电机标量 v/f 控制以驱动电机，从而验证电流和电压检测电路以及栅极驱动器电路。
- 测试用于电机控制的 eSMO 模块。

在该构建级别中，系统以开环控制方式运行，因此 ADC 值仅用于验证和确认，ADC 值实际上不用在电机的控制环路中。图 4-15 展示了该构建级别的软件流程。

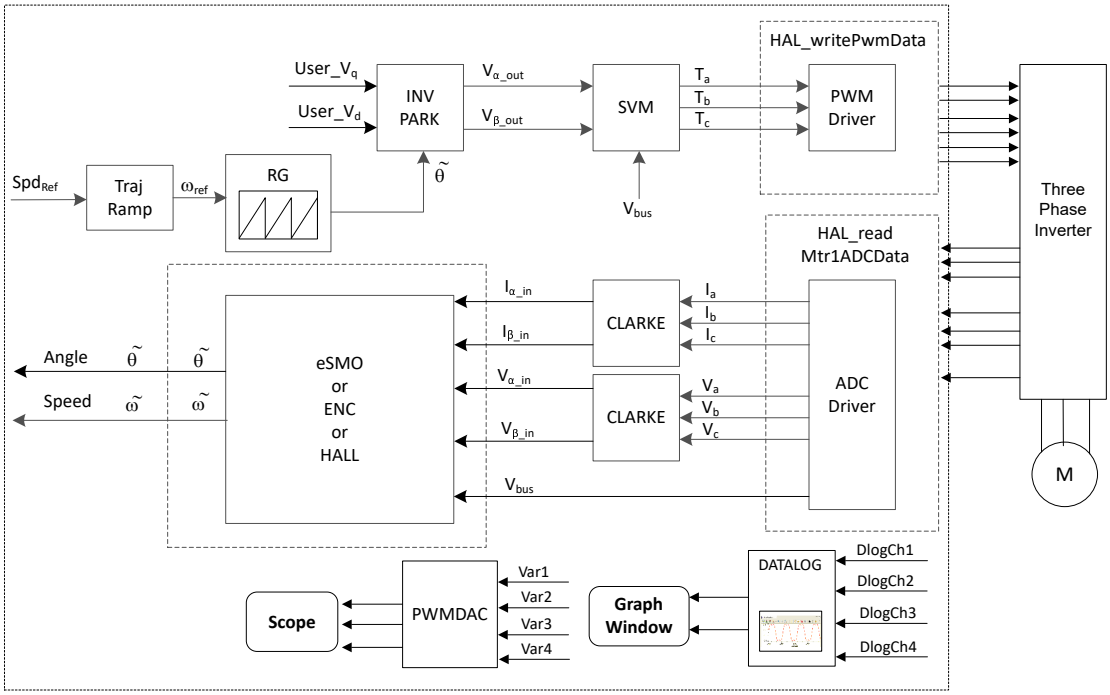


图 4-15. 构建级别 2 软件方框图 – 开环控制

4.4.2.1 构建和加载工程

将电机连接到电机驱动器评估板上的相应端子。按照节 4.4.1.1 中的第 2-7 步构建和加载工程。在第 4 步中，将 DMC_BUILDLEVEL 设置为 DMC_LEVEL_2。

4.4.2.2 设置调试环境窗口

按照节 4.4.1.2 中的步骤将变量导入“Expressions”窗口。对于构建级别 2，选择 *universal_motor_control_level2.txt* 文件。此时将显示“Expressions”窗口，如图 4-16 所示。

4.4.2.3 运行代码

1. 为适当的电源加电，并逐渐增加电源的输出电压以获得适当的直流母线电压。

2. 如果使用图形工具，则级别 2 使用与实验 1 相同的图形配置和参数来监控 2 个相电流。
3. 通过取消选中 *Tools* > *Advanced Features* 中的 *Data Cache Enabled* 来禁用数据缓存。
4. 通过点击“Resume”按钮来运行工程，或依次点击“Debug”选项卡中的 *Run* → *Resume*。*systemVars.flagEnableSystem* 在固定时间后设为 *1*，这意味着已完成偏移校准。故障标志 *motorVars_M1.faultMtrUse.all* 等于 *0*。如果情况并非如此，用户必须按照节 4.4.1.3 中所述仔细检查 1 级电流和电压检测电路。此外，如果 *motorVars_M1.faultMtrPrev.bit* 中的 *moduleOverCurrent* 为 1，则需要将 *motorSetVars_M1.overCurrent_A* 设置为更高的值，以避免初始高电流故障。
5. 要验证电机逆变器的电流和电压检测电路，请在“Expressions”窗口中将变量 *motorVars_M1.flagEnableRunAndIdentify* 设置为 *1*，如图 4-16 所示。电机将以电压/频率 (v/f) 开环运行。如果电机旋转不平稳，请根据电机规格调整 *user_mtr1.h* 文件中的 v/f 曲线参数，如以下所示。注意：修改这些参数需要重新编译工程。有关在调试模式下重新编译工程的更多信息，请参阅节 4.4.1.3 的步骤 15。

```
#define USER_MOTOR1_FREQ_LOW_HZ             (5.0)           // Hz
#define USER_MOTOR1_FREQ_HIGH_HZ            (400.0)         // Hz
#define USER_MOTOR1_VOLT_MIN_V              (1.0)           // Volt
#define USER_MOTOR1_VOLT_MAX_V              (24.0)          // Volt
```

6. *motorVars_M1.speedRef_Hz* 变量用于设置电机的速度基准。在“Expressions”窗口中检查 *motorVars_M1.speed_Hz* 变量的值，以保持电机转速 (*motorVars_M1*speed_Hz) 接近于基准速度 (*motorVars_M1.speedRef_Hz)*，如图 4-16 所示。
7. 在此构建级别中，需要验证电流检测、电压检测、转子角度估算器和发生器。这可以使用 HV 电机套件中的 PWMDAC 来完成，如节 4.5.2 中所述。此外，DATALOG 模块可用于查看这些感应波形。有关使用 DATALOG 查看电流、电压和角度信号的更多信息，请参阅步骤 8。
8. 如果将 DATALOG 模块与图形工具配合使用来检查电流检测信号、电压检测信号和角度输出，请按照下述步骤进行操作。关于 datalog 模块的更多信息，请参阅节 4.5.1。注意：在修改代码之后，您必须在下列两个步骤之间重建工程。
   a. 要使用 DATALOG 模块测试相电流，必须在 *sys_main.c* 文件中设置以下代码。注意：默认情况下，此代码已配置为构建级别 2。图形工具上显示的相电流采样信号波形如图 4-18 所示。

```
datalogObj->iptr[0] = (float32_t*) &motorVars_M1.adcData.I_A.value[0];
datalogObj->iptr[1] = (float32_t*) &motorVars_M1.adcData.I_A.value[1];
```

   b. 要使用 DATALOG 模块测试相位电压，必须在 *sys_main.c* 文件中设置以下代码。图形工具上显示的相电压采样信号波形如图 4-19 所示。

```
datalogObj->iptr[2] = (float32_t*) &motorVars_M1.adcData.V_V.value[0];
```

   c. 可以在图形工具上监视力角生成器或估算器的角度，如图 4-21 中所示。请注意，力角发生器的角度与 eSMO 估算器的估算转子角度非常相似。

```
datalogObj->iptr[3] = (float32_t*) &motorVars_M1.angleFOC_rad;
```

9. 通过减小变量 *motorVars_M1.overCurrent_A* 的值来验证过流故障保护，过流保护由 CMPSS 模块实现。如果 *motorVars_M1.overCurrent_A* 设为小于电机相电流实际值的值，PWM 输出会被禁用，*motorVars_M1.flagEnableRunAndIdentify* 将设置为 *0*，如图 4-17 所示。
10. 将变量 *motorVars_M1.flagEnableRunAndIdentify* 设为 0 停止运行电机。
11. 完成后，现在可以停止控制器，并终止调试连接。通过首先点击工具栏上的“Halt”按钮或点击 *Target* → *Halt* 来完全停止控制器。最后，通过点击“CPU Reset”或依次点击 *Run* → *Reset* 来复位控制器。
12. 通过点击“Terminate Debug Session”按钮或依次点击 *Run* → *Terminate* 来关闭 CCS 调试会话。
13. 关闭逆变器套件的电源。

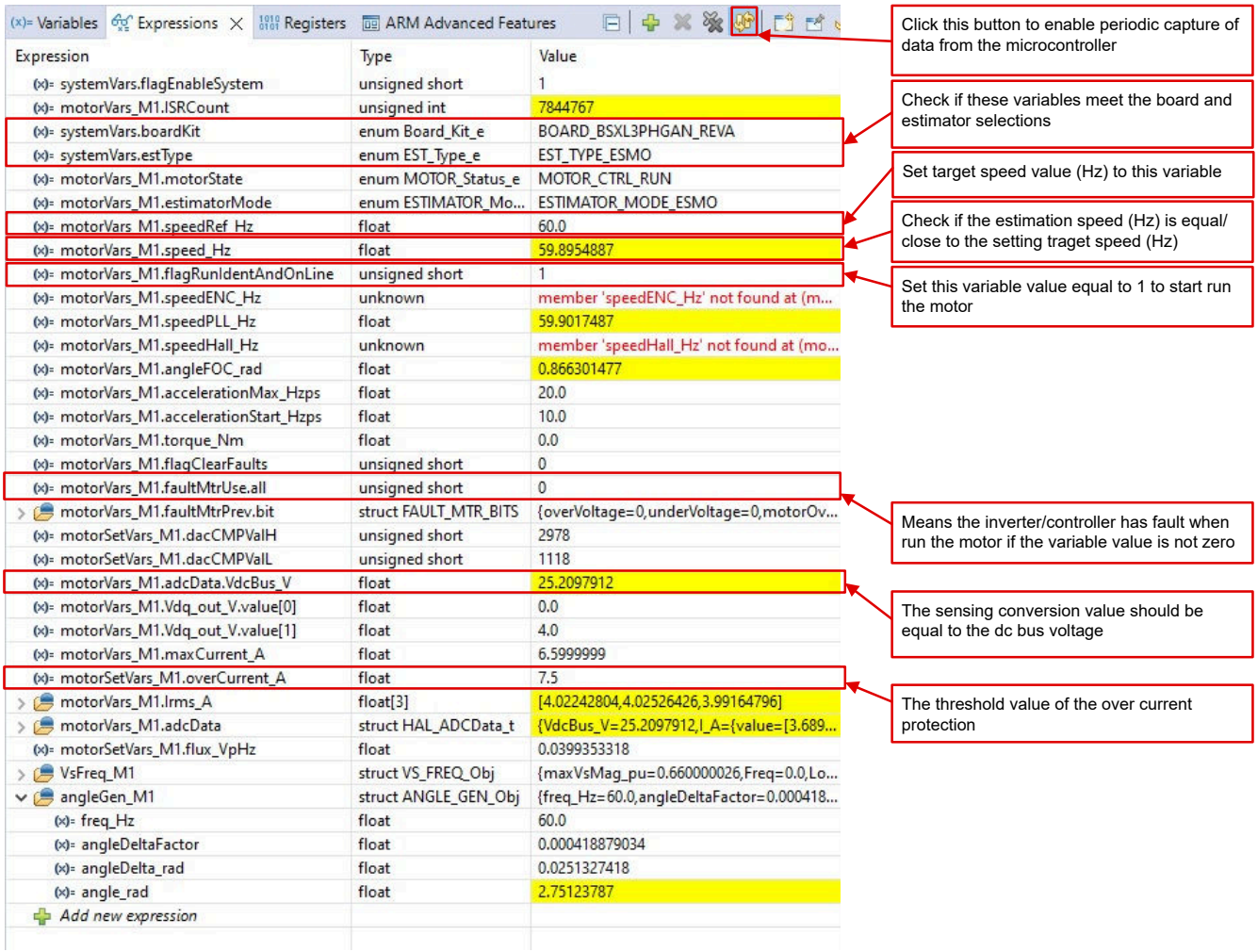


图 4-16. 构建级别 2：表达式窗口中的变量

在“Expression”表达式窗口中调整 *motorVars_M1.overCurrent_A* 的值，以触发过流故障，如图 4-17 所示。

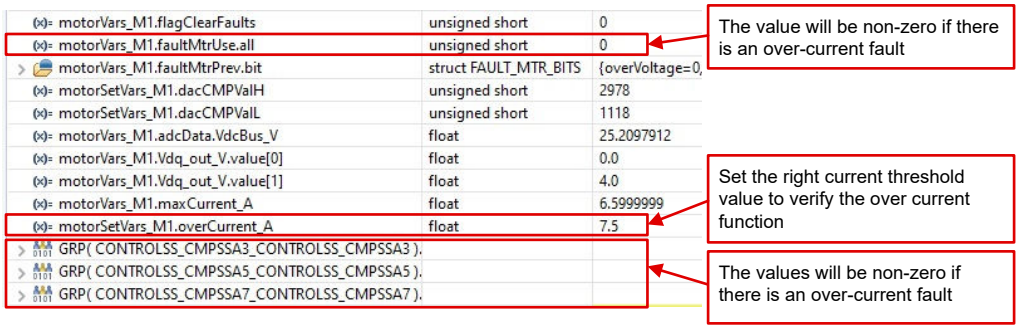


图 4-17. 构建级别 2：电流保护设置

将 DATALOG 与图形工具一起使用，以监测电机的三相检测电流，如图 4-18 所示。

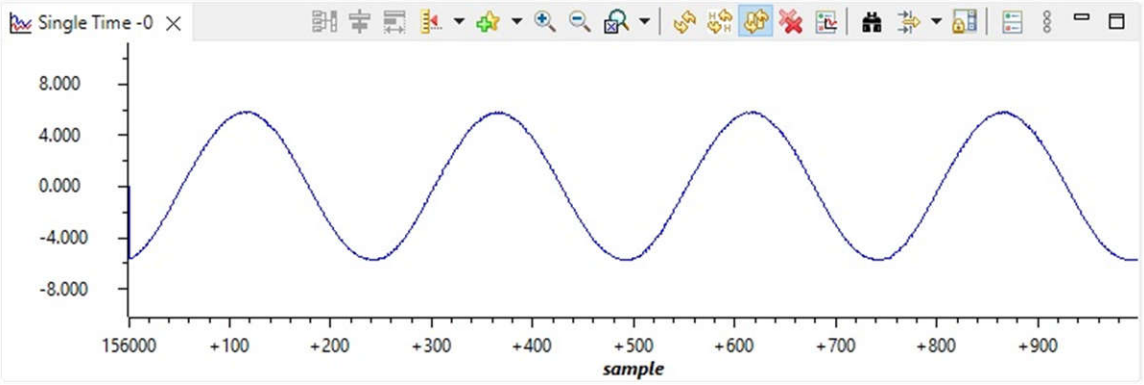


图 4-18. 构建级别 2：使用图形工具时的电机相电流波形

使用数据记录器和图形工具监测电机的三相检测电压，如图 4-19 所示。此处选择的 SVM 模式是具有最小调制的 DPWM。

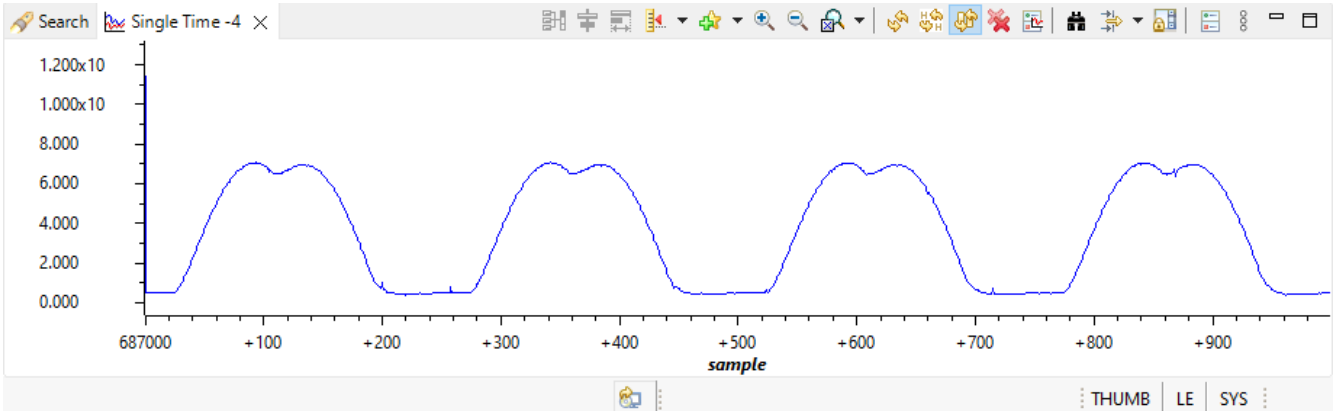


图 4-19. 构建级别 2：使用图形工具时的电机相电压波形 - DPWM 最小值

将 DATALOG 与图形工具配合使用，在 SVPWM 共模调制模式下监测电机的三相检测电压，如图 4-20 所示。可在 *motor1_drive.c* 文件中选择 SVM 类型。

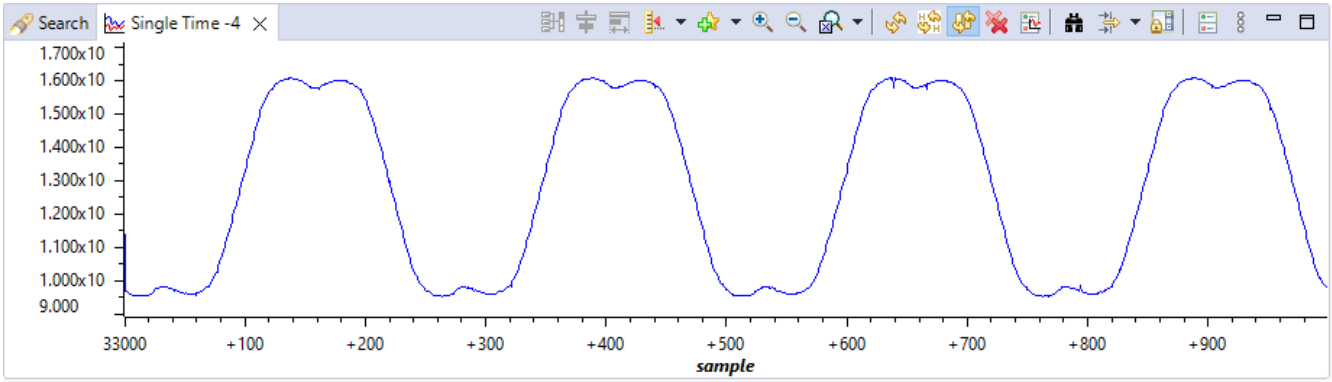


图 4-20. 构建级别 2：使用图形工具时的电机相电压波形 - SVPWM 通用

将 Datalog 与图形工具配合使用，从角度发生器监视电机的转子角度以及 eSMO 估算器的角度，如图 4-21 所示。

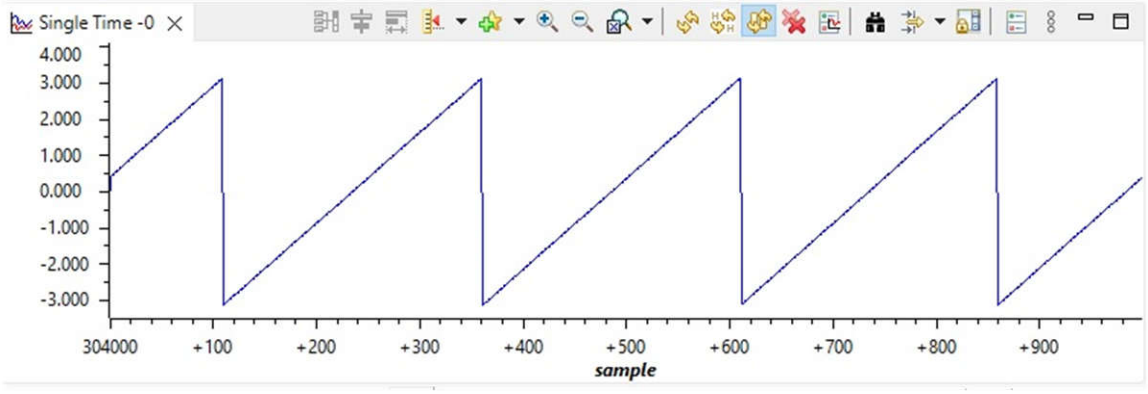


图 4-21. 构建级别 2：使用图形工具时的电机转子角度波形

4.4.3 3 级递增构建

了解该构建级别中的目标：

- 评估电机的闭合电流环路运行。
- 验证电流检测参数设置。

在这个构建级别，电机由 i/f 控制进行控制，转子角度由斜坡发生器模块生成。图 4-22 展示了该构建级别的软件流程。

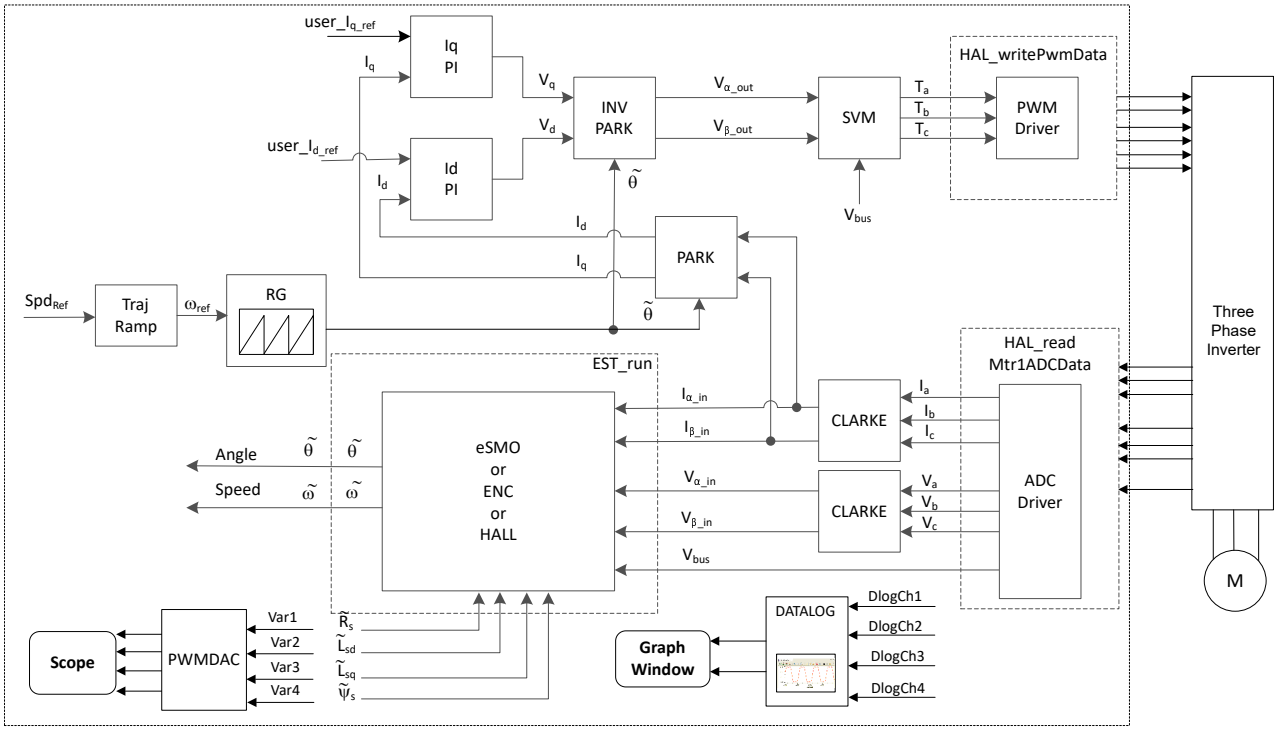


图 4-22. 构建级别 3 软件方框图 – 电流闭环控制

4.4.3.1 构建和加载工程

将电机连接到电源逆变器板上的相关端子。按照节 4.4.1.1 中的操作步骤，通过在 *sys_settings.h* 文件中将 DMC_BUILDLEVEL 设置为 DMC_LEVEL_3 来构建和加载工程。

4.4.3.2 设置调试环境窗口

通过选择 *universal_motor_control_level3.txt*，按照节 4.4.1.2 中的操作步骤将变量导入“Expressions”窗口。此时将显示“Expressions”窗口，如图 4-23 所示。

4.4.3.3 运行代码

1. 打开交流或直流电源，逐渐增加电源上的输出电压以获得适当的直流母线电压。
2. 通过取消选中 *Tools > Advanced Features* 中的 *Data Cache Enabled* 来禁用数据缓存。
3. 通过点击“Resume”按钮来运行工程，或依次点击“Debug”选项卡中的 *Run → Resume*。*systemVars.flagEnableSystem* 必须在固定时间后设为 *1*，这意味着已完成偏移校准。故障标志 *motorVars_M1.faultMtrUse.all* 等于 *0*，否则用户必须检查电流和电压检测电路，如节 4.4.1 中所述。
4. 要验证运行电机时进行电流闭环控制，请在“Expressions”窗口中将变量 *motorVars_M1.flagEnableRunAndIdentify* 设置为 *1*，如图 4-23 中所示。电机在运行时会使用来自角度发生器的角度进行闭环控制，运行速度在 *motorVars_M1.speedRef_Hz* 变量中设置。在“Expressions”窗口中检查 *motorVars_M1.speed_Hz* 的值。这两个变量的值非常接近。
5. 将示波器探头连接到 EPWMDAC（用于 HV 套件）输出和电机相线，以探测角度和电流信号以及电流。示波器上的这些波形如图 4-24 所示。在“Expressions”窗口中更改 *motorVars_M1.Idq_set_A.value[1]* 以设置基准扭矩电流，相应地，电机相电流将以相同的百分比增加。
6. 如果电机无法以电流闭环运行并出现过流故障，则检查是否根据硬件板正确设置了 *motorVars_M1.adcData.current_sf* 的符号和 *userParams_M1.current_sf* 的值。这两个变量的值都与 *user_mtr1.h* 文件中的定义常量 USER_M1_ADC_FULL_SCALE_CURRENT_A 相关。
7. 将变量 *motorVars_M1.flagEnableRunAndIdentify* 设为 0 停止运行电机。
8. 完成后，现在可以停止控制器，并终止调试连接。通过首先点击工具栏上的“Suspend”按钮或点击 *Target → Halt* 来完全停止控制器。最后，通过点击“CPU Reset”按钮或依次点击 *Run → Reset* 来复位控制器。
9. 通过点击“Terminate Debug Session”按钮或依次点击 *Run → Terminate* 来关闭 CCS 调试会话。

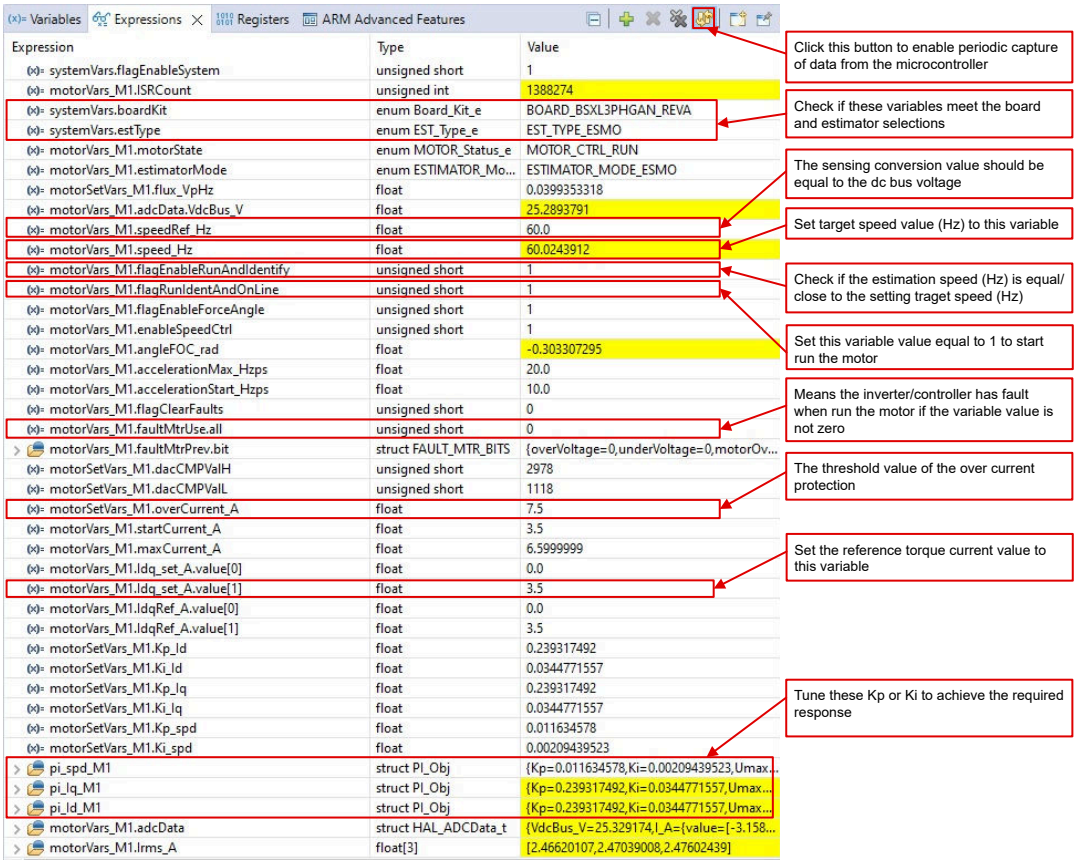


图 4-23. 构建级别 3：表达式窗口中的变量

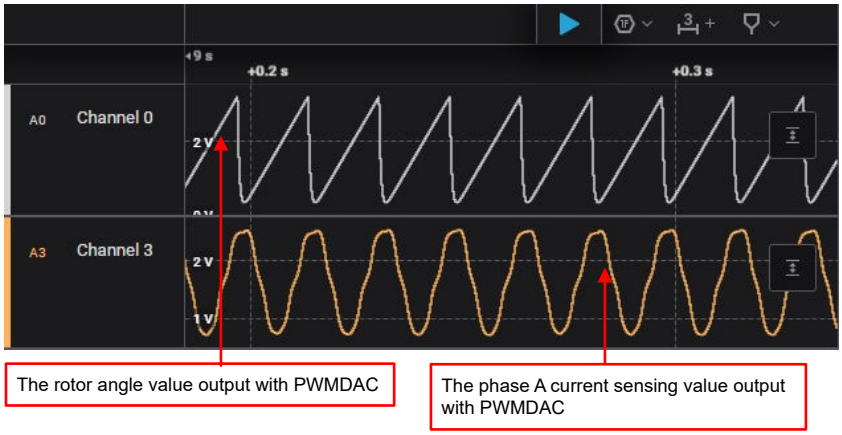


图 4-24. 构建级别 3：EPMWDAC 进行电机转子角度和相电流波形监控

4.4.4 4 级递增构建

了解该构建级别中的目标：

- 使用基于 eSMO 的无传感器 FOC、基于编码器的有传感器 FOC 或基于霍尔的有传感器 FOC 评估整个电机驱动器。
- 评估其他特性，例如弱磁控制、快速启动、MTPA 和制动。

在此构建级别，外部速度环路是闭合的，内部电流环路用于电机，使得转子角度来自 eSMO、编码器或霍尔传感器模块。图 4-25 展示了该构建级别的软件流程。

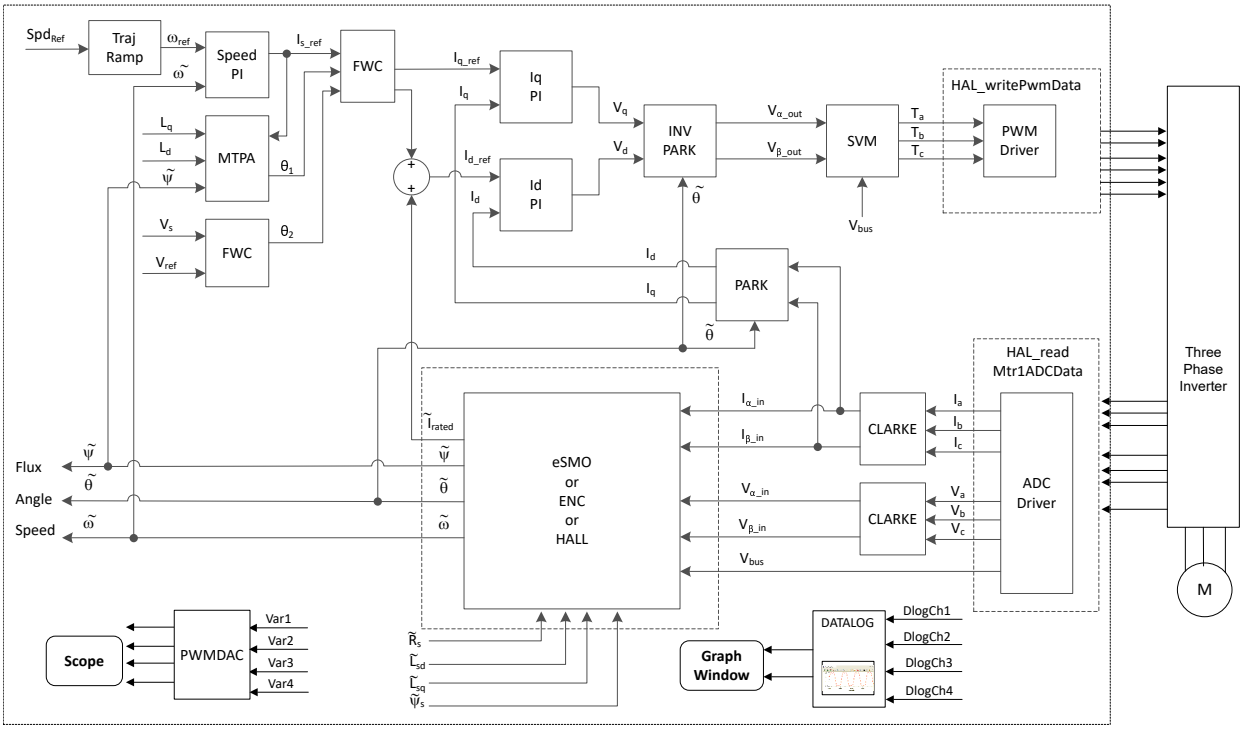


图 4-25. 构建级别 4 软件方框图 – 转速和电流闭环控制

4.4.4.1 构建和加载工程

将电机连接到电源逆变器板上的相关端子。按照节 4.4.1.1 中的操作步骤，通过在 *sys_settings.h* 文件中将 DMC_BUILDLEVEL 设置为 DMC_LEVEL_4 来构建和加载工程。

4.4.4.2 设置调试环境窗口

通过选择 *universal_motor_control_level4.txt*，按照节 4.4.1.2 中的操作步骤将变量导入“Expressions”窗口。此时将显示“Expressions”窗口，如图 4-26 所示。

4.4.4.3 运行代码

1. 打开交流或直流电源，逐渐增加电源上的输出电压以获得适当的直流母线电压。
2. 必须在头文件 *user_mtr1.h* 中定义所需的电机参数，如以下示例代码所示。

```
#define USER_MOTOR1_TYPE MOTOR_TYPE_PM
#define USER_MOTOR1_NUM_POLE_PAIRS (4)
#define USER_MOTOR1_Rr_Ohm (NULL)
#define USER_MOTOR1_Rs_Ohm (0.38157931f)
#define USER_MOTOR1_Ls_d_H (0.000188295482f)
#define USER_MOTOR1_Ls_q_H (0.000188295482f)
#define USER_MOTOR1_RATED_FLUX_VpHz (0.0396642499f)
```

3. 构建工程并将代码加载到控制器中，通过取消选中“Tools”>“ARM Advanced Features”中的“Data Cache Enabled”来禁用数据缓存，通过点击“Resume”按钮来运行工程，或点击“Debug”选项卡中的 *Run → Resume*。经过固定的时长后，*systemVars.flagEnableSystem* 设置为 *1*，这意味着偏移校准已完成并且浪涌电源继电器已开启。电机故障标志 *motorVars_M1.faultMtrUse.all* 等于 *0*，否则用户必须检查电流和电压检测电路，如节 4.4.1 中所述。
4. 在“Expressions”窗口中将变量 *motorVars_M1.flagEnableRunAndIdentify* 设置为 *1*，如图 4-26 所示。
5. 开始运行电机后：
   a. 将目标速度值设为变量 *motorVars_M1.speedRef_Hz*，并观察电机轴速度如何随设定速度改变。
   b. 要改变加速度，请在变量 *motorVars_M1.accelerationMax_Hzps* 中输入不同的加速度值。
   c. 如节 4.5.2 所述，使用 PWMDAC 模块显示监视变量。电机角度和电流波形如图 4-27 所示。
6. FOC 系统电流控制器的默认比例增益 (Kp) 和积分增益 (Ki) 在函数 *setupControllers()* 中计算。调用 *setupControllers*() 后，全局变量 motorSetVars_M1.Kp_Id、motorSetVars_M1.Ki_Id、motorSetVars_M1.Kp_Iq 和 motorSetVars_M1.Ki_Iq 将使用新计算得出的 Kp 和 Ki 增益进行初始化。如图 4-26 中所示，调整“Expressions”窗口中这四个变量的 Kp 和 Ki 值，使电流控制器实现预期的电流控制带宽和响应。Kp 增益会产生一个零点，以抵消电机定子的极点，可以轻松地计算得出。Ki 增益可调整电流控制器-电机系统的带宽。如果需要一个速度控制系统来实现特定阻尼，电流控制器的 Kp 增益将与速度控制系统的时间常数相关。
7. 将变量 *motorVars_M1.flagEnableRunAndIdentify* 设为 *0* 停止运行电机。
8. 完成后，现在可以停止控制器，并终止调试连接。通过首先点击工具栏上的“Suspend”按钮或点击 *Target → Halt* 来完全停止控制器。最后，通过点击“CPU Reset”按钮或依次点击 *Run → Reset* 来复位控制器。
9. 通过点击“Terminate Debug Session”按钮或依次点击 *Run → Terminate* 来关闭 CCS 调试会话。

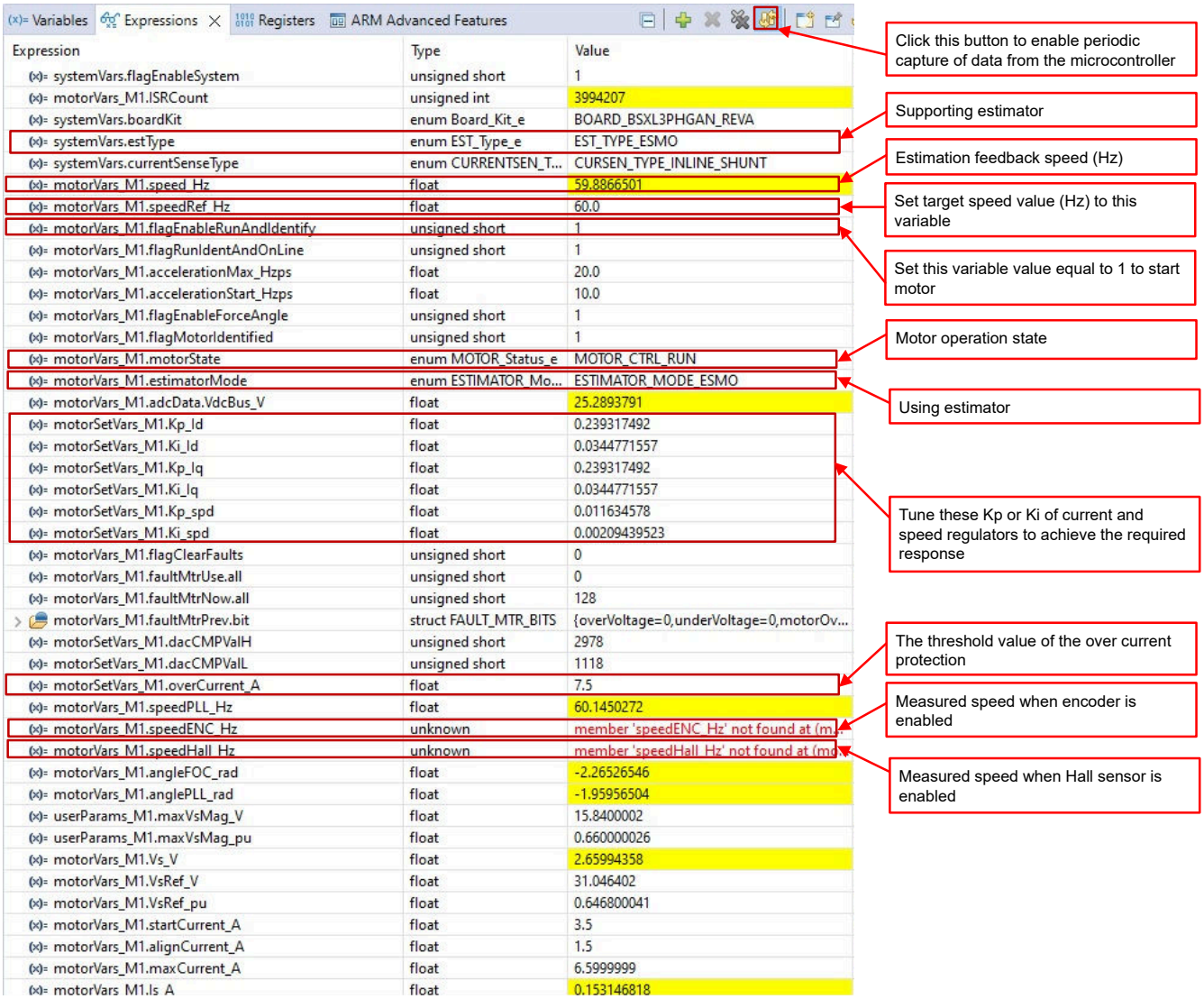


图 4-26. 构建级别 4：表达式窗口中的变量

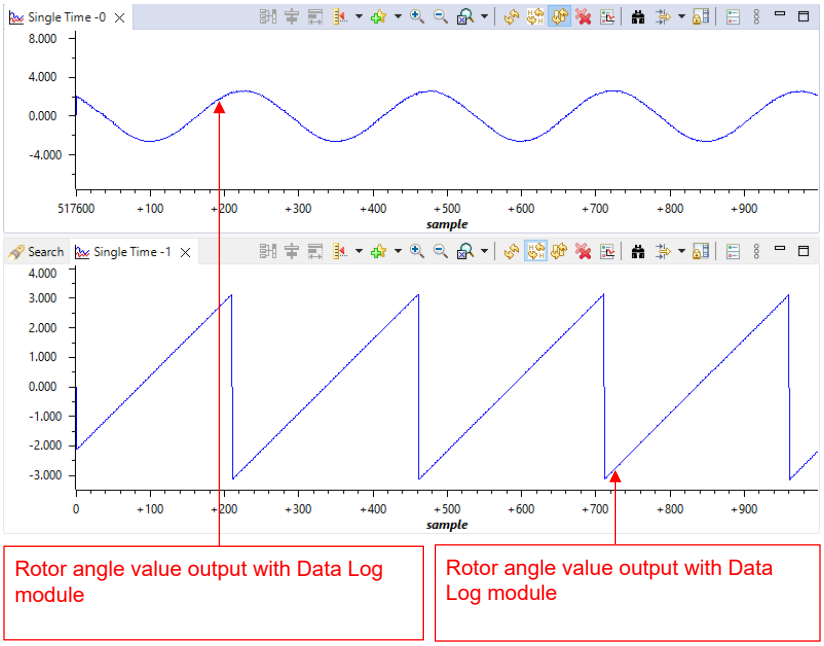


图 4-27. 构建级别 4：正向移动时的相电流和转子角度以及 eSMO 波形

如节 4.2.2 所示，工程中可支持多种 FOC 算法。用户可以使用一种（eSMO、霍尔或编码器）算法或两种算法（eSMO + 编码器）进行工程中的电机控制。

，用户可以通过在工程属性中添加预定义名称 *MOTOR1_ESMO* 和 *MOTOR1_ENC*，在工程中同时实施 eSMO 和编码器估算器，如节 4.2.1 中所述。按照上述操作步骤，重新构建、加载和运行工程。

- systemVars.estType 值等于 EST_TYPE_ESMO_ENC，这意味着 eSMO 和编码器估算器在该工程中处于启用状态。
- motorVars_M1.estimatorMode 等于 ESTIMATOR_MODE_ESMO，这意味着 eSMO 估算器正在用于无传感器 FOC；等于 ESTIMATOR_MODE_ENC，这意味着编码器估算器正在用于有传感器 FOC。
- 来自 eSMO 和编码器的估算转子角度如图 4-28 所示。通过将 motorVars_M1.speedRef_Hz 设置为正值，电机在 eSMO 正向旋转的情况下运行。
- 用户可以将值更改为 ESTIMATOR_MODE_ENC，以选择有传感器 FOC 的编码器估算器。用户还可以更改值以动态地使用估算器进行切换。

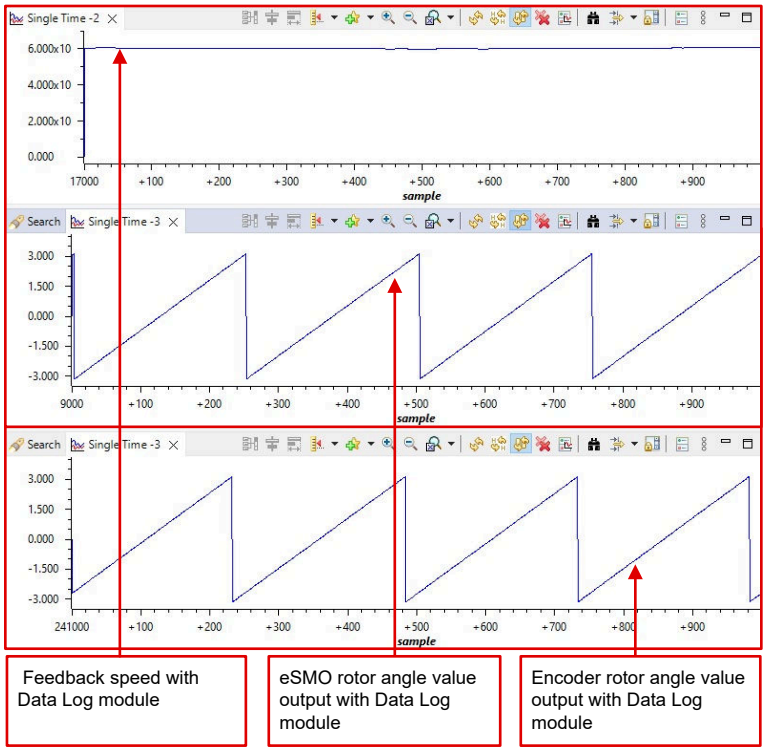


图 4-28. 构建级别 4：采用 eSMO 和编码器的转子角度，正向旋转时的相电流波形

4.5 向电机控制项目中添加附加功能

4.5.1 使用 DATALOG 函数

DATALOG 模块将用户可选软件变量（缺省情况下为四个变量）的实时值存储在 TI MCU 上提供的数据 RAM 中，如图 4-29 所示。通过将模块输入配置为四个变量的地址来选择四个变量。四个 RAM 缓冲区位置的起始地址为 &((datalog).datalogBuff)[0]、&((datalog).datalogBuff)[1]、&((datalog).datalogBuff)[2] 和 &((datalog).datalogBuff)[3]。这些 Datalog 缓冲区是包含值触发数据的大型数组，这些数据随后可以显示在图形中。DATALOG 预分频器是可配置的，这使得 Datalog 函数只能从每个预分频样本中记录一个。可以在 *datalog_input.h* 文件中选择数据日志缓冲区的数量、缓冲区大小和数据类型。

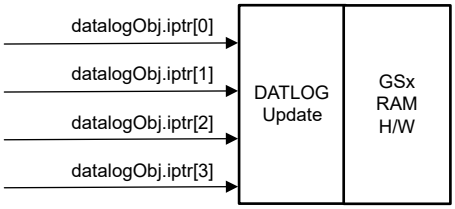


图 4-29. DATALOG 模块方框图

要启用 DATALOG 功能，必须在工程属性中添加预定义符号 DATALOG_EN，如图 4-2 所示。

以下代码演示了一个 DATALOG 对象和句柄的声明。此代码位于 *datalog.c* 文件中。

```
__attribute__ ((section("datalog_data"))) DATALOG_Obj datalog;
DATALOG_Handle datalogHandle;       //!< the handle for the Datalog object
```

这会将数据日志对象置于存储器的 *datalog_data* 段中。该段可以是 TCM 或 OCRAM。通常，我们建议使用 OCRAM，因为 TCM 大小有限，且为软件的时间关键型部分所需。在 CCS12.6 中，禁用数据高速缓存以便能够记录 OCRAM 中的数据。要禁用数据缓存，必须取消选中 *Tools > ARM Advanced Features* 中的 *Data Cache Enabled*，如图 4-30 所示。

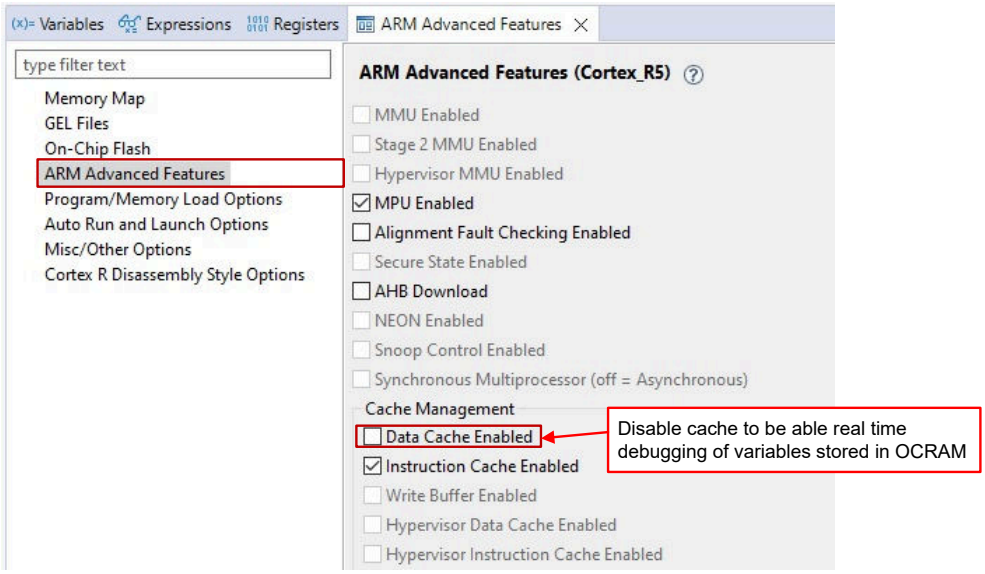


图 4-30. 禁用数据缓存以进行实时调试

以下代码展示了 datalog 对象、句柄和参数的初始化和设置。此代码位于 *sys_main.c* 文件中。

```
// Initialize Datalog
datalogHandle = DATALOG_init(&datalog, sizeof(datalog), manual, 0, 1);
DATALOG_Obj *datalogObj = (DATALOG_Obj *)datalogHandle;
```

以下代码展示了指向变量地址的模块输入的配置。datalog 模块输入指向不同的系统变量，具体取决于构建级别。此代码位于 *sys_main.c* 文件中：

```
datalogObj->iptr[0] = (float32_t*) &motorVars_M1.adcData.V_V.value[0];
datalogObj->iptr[1] = (float32_t*) &motorVars_M1.adcData.I_A.value[0];
datalogObj->iptr[2] = (float32_t*) &motorVars_M1.adcData.I_A.value[1];
datalogObj->iptr[3] = (float32_t*) &motorVars_M1.angleFOC_rad;
```

以下代码演示了在 **motor1ctrlISR()** 中断执行期间使用新数据定期更新 datalog 缓冲区。此代码位于 *motor1_drive.c* 文件中。

```
#if defined(DATALOG_EN)
DATALOG_update(datalogHandle);
#endif  // DATALOG_EN
```

datalog 模块与图形工具一同使用，该工具提供了一种直观检查变量并判断系统性能的方法。CCS 中提供了图形工具，可以各种图形类型显示数据数组。数据数组以各种格式存储在器件的存储器中。

当工程处于调试模式时，打开并设置时间图窗口来绘制数据日志缓冲区，如图 4-31 中所示。或者，用户可以导入位于工程文件夹中的图形配置文件。要导入它们，请点击：Tools -> Graph -> Single Time…，选择“import”并浏览到以下位置 `<workspace>\universal_motorcontrol_am263x_r5fss0-0_nortos_ti-arm-clang\src_control\debug\`，然后选择 *datalog.graphProp* 文件。点击“OK”，这样会将 Graphs 添加到调试视图中。点击图形选项卡左上角的“Continuous Refresh”按钮。

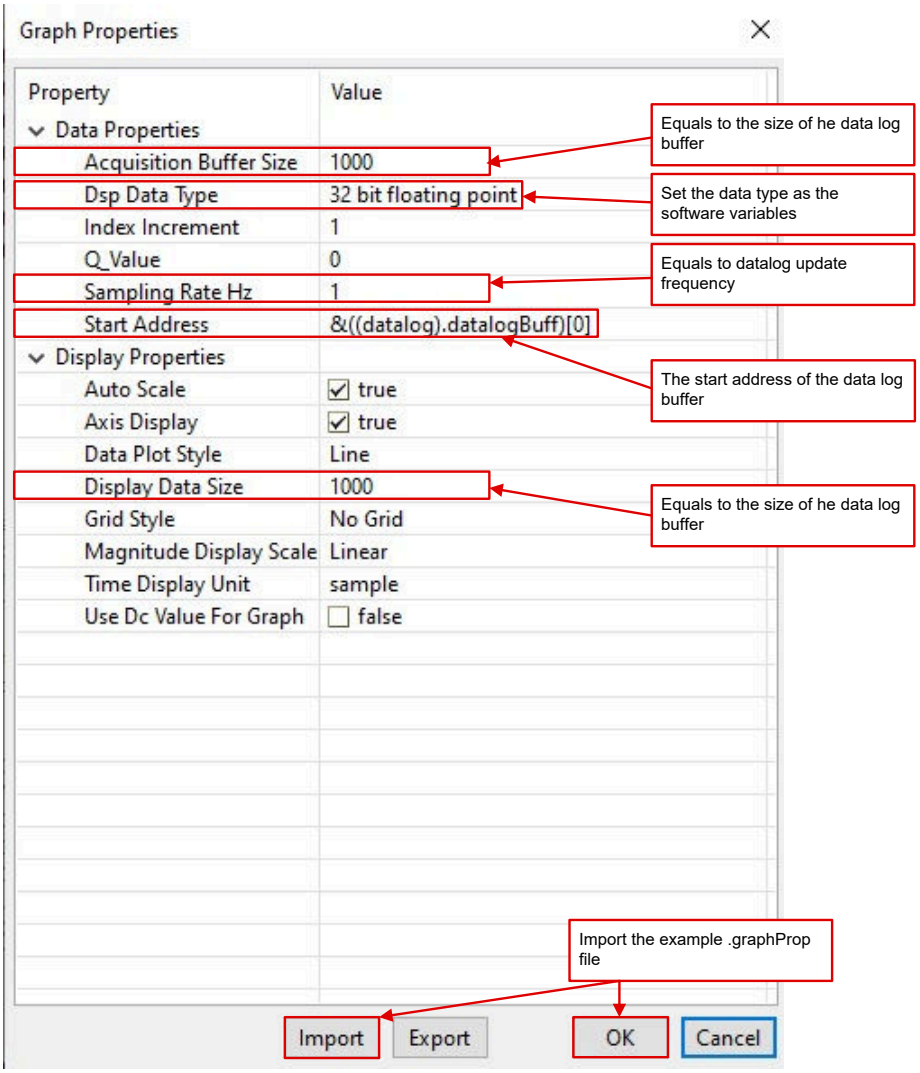


图 4-31. 图形窗口设置

4.5.2 使用 PWMDAC 函数

PWMDAC 模块使用 ePWM 5A、5B、6A 和 6B 将软件变量转换为 PWM 信号，如图 4-32 所示。仅高压套件 (TMDSHVMTRINSPIN) 支持 PWMDAC 模块，因为该模块具有额外的 PWM 输出，电路板上提供 RC 滤波器。如果将 PWMDAC 模块与不支持 PWMDAC 模块的电机驱动器板一起使用，那么 PWM 信号被路由到 TI LaunchPad 上的备用 PWM，并且用户需要在这些引脚上添加 RC 滤波器来利用 PWMDAC 设计。

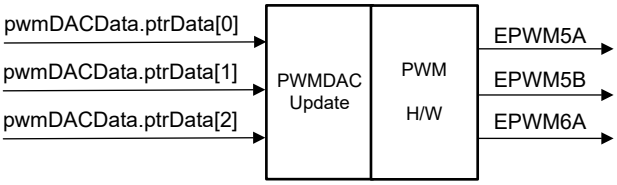


图 4-32. PWMDAC 模块方框图

PWMDAC 模块可被用于通过外部低通滤波器来查看相关引脚输出上的信号，此信号由变量表示。因此，需要使用外部低通滤波器来查看图 4-33 中所示的实际信号波形。（1 阶）RC 低通滤波器用于滤除嵌入在实际低频信号中的高频分量。要选择 R 和 C 值，时间常数可以用截止频率 (f_c) 表示，如以下公式所示。

$$\tau = RC = \frac{1}{2\pi f_c} \tag{62}$$

$$f_c = 2\pi RC \tag{63}$$

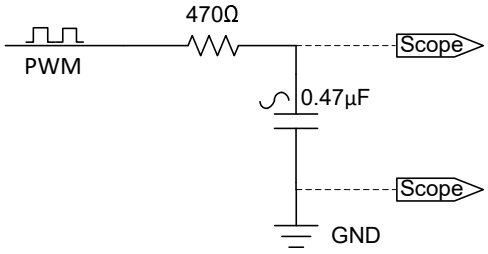


图 4-33. 连接到 PWM 引脚的外部 RC 低频滤波器

要启用 ePWM DAC 功能，必须在工程属性中添加预定义符号 EPWMDAC_MODE，如图 4-2 所示。

以下代码显示了 PWMDAC 对象的声明。此代码位于 *sys_main.c* 文件中。

```
#if defined(EPWMDAC_MODE)
#if defined(HVMTRPFC_REV1P1)
__attribute__ ((section("sys_data"))) HAL_PWMDACData_t pwmDACData;
  // HVMTRPFC_REV1P1
#else
#error EPWMDAC is not supported on this kit!
#endif  // !HVMTRPFC_REV1P1
#endif  // EPWMDAC_MODE
```

以下代码展示了 PWMDAC 对象、句柄和参数的初始化和设置。四个模块输入 ptrData[0]、ptrData[1]、ptrData[2] 和 ptrData[3] 被配置为指向四个变量的地址。PWMDAC 模块输入指向不同的系统变量，这取决于构建级别。此代码位于 *sys_main.c* 文件中。

```
// set DAC parameters
pwmDACData.periodMax =
        PWMDAC_getPeriod(halHandle->pwmDACHandle[PWMDAC_NUMBER_1]);

pwmDACData.ptrData[0] = &motorVars_M1.angleFOC_rad;              // PWMDAC1
pwmDACData.ptrData[1] = &motorVars_M1.speedAbs_Hz;               // PWMDAC2
pwmDACData.ptrData[2] = &motorVars_M1.speedAbs_Hz;               // PWMDAC3
pwmDACData.ptrData[3] = &motorVars_M1.adcData.I_A.value[1];      // PWMDAC4

pwmDACData.offset[0] = 0.5f;     // PWMDAC1
pwmDACData.offset[1] = 0.0f;     // PWMDAC2
```

```
pwmDACData.offset[1] = 0.0f;     // PWMDAC3
pwmDACData.offset[3] = 0.5f;     // PWMDAC4

pwmDACData.gain[0] = 1.0f / MATH_TWO_PI;                              // PWMDAC1
pwmDACData.gain[1] = 1.0f / USER_MOTOR1_FREQ_MAX_Hz;                  // PWMDAC2
pwmDACData.gain[2] = 1.0f / USER_MOTOR1_FREQ_MAX_Hz;                  // PWMDAC3
pwmDACData.gain[3] = 2.0f / USER_M1_ADC_FULL_SCALE_CURRENT_A;         // PWMDAC4
```

以下代码演示了在 **motor1ctrlISR()** 中断执行期间使用新数据更新 PWM 输出。此代码位于 *motor1_drive.c* 文件中。

```
// connect inputs of the PWMDAC module.
HAL_writePWMDACData(halHandle, &pwmDACData);
```

4.5.3 添加 CAN 功能

可以将 CAN 功能添加到实验工程中，为用户提供用于发送启动/停止命令并获取运行状态反馈的通信总线。要利用此功能，请在工程构建属性中启用预定义符号 CMD_CAN，如图 4-2 中所示。PCAN-View 用于简单地监控、发送和记录 CAN 数据流量。可以在 *motor_common.h* 文件中定义不同类型的命令消息。在本工程中，发送器指定了一个启动命令并定义了目标速度值。接收器随后会收到包含目标速度的邮件。

请注意，根据您使用的是套件中的 LaunchPad 还是 EVM，需要进行一些引脚多路复用设置。这些配置是使用 *mcanEnableTransceiver* 和 *tca6416ConfigOutput* 函数实现的，如以下代码所述：

```
#if defined(AM263_CC)
void tca6416ConfigOutput(uint16_t port, uint16_t pin, uint16_t level);
#endif // AM263_CC
```

```
#if defined(CMD_CAN)
#if defined(AM263_LP)

void mcanEnableTransceiver(void)
{
    uint32_t    gpioBaseAddr, pinNum;

    gpioBaseAddr = (uint32_t)AddrTranslateP_getLocalAddr(MCAN_ENABLE_BASE_ADDR);
    pinNum       = MCAN_ENABLE_PIN;

    GPIO_setDirMode(gpioBaseAddr, pinNum, GPIO_DIRECTION_OUTPUT);

    GPIO_pinWriteLow(gpioBaseAddr, pinNum);
}
#endif // AM263_LP

#if defined(AM263_CC)
/* ========================================================================== */
/*                            Macros & Typedefs                               */
/* ========================================================================== */

/* Input status register */
#define TCA6416_REG_INPUT0              ((UInt8) 0x00U)
#define TCA6416_REG_INPUT1              ((UInt8) 0x01U)

/* Output register to change state of output BIT set to 1, output set HIGH */
#define TCA6416_REG_OUTPUT0             ((uint8_t) 0x02U)
#define TCA6416_REG_OUTPUT1             ((uint8_t) 0x03U)

/* Configuration register. BIT = '1' sets port to input, BIT = '0' sets
 * port to output */
#define TCA6416_REG_CONFIG0             ((uint8_t) 0x06U)
#define TCA6416_REG_CONFIG1             ((uint8_t) 0x07U)

/* ========================================================================== */
/*                          Function Declarations                             */
/* ========================================================================== */
static void SetupI2CTransfer(I2C_Handle handle,  uint32_t targetAddr,
                      uint8_t *writeData, uint32_t numWriteBytes,
                      uint8_t *readData,  uint32_t numReadBytes);

void mcanEnableTransceiver(void)
{
    I2C_Handle      i2cHandle;
```

```
    uint8_t         dataToSlave[4];

    i2cHandle = gI2cHandle[CONFIG_I2C0];
    dataToSlave[0] = TCA6416_REG_CONFIG0;
    dataToSlave[1] = 0x0U;
    SetupI2CTransfer(i2cHandle, 0x20, &dataToSlave[0], 1, &dataToSlave[1], 1);
    /* set the P00 to 0 make them output ports. */
    dataToSlave[1] &= ~(0x1U);
    SetupI2CTransfer(i2cHandle, 0x20, &dataToSlave[0], 2, NULL, 0);

    /* Get the port values. */
    dataToSlave[0] = TCA6416_REG_INPUT0;
    dataToSlave[1] = 0x0U;
    SetupI2CTransfer(i2cHandle, 0x20, &dataToSlave[0], 1, &dataToSlave[1], 1);

    /* Set P10 and P11 to 0.
     */
    dataToSlave[0] = TCA6416_REG_OUTPUT0;
    dataToSlave[1] &= ~(0x1);
    SetupI2CTransfer(i2cHandle, 0x20, &dataToSlave[0], 2, NULL, 0);
}

static void SetupI2CTransfer(I2C_Handle handle,  uint32_t targetAddr,
                      uint8_t *writeData, uint32_t numWriteBytes,
                      uint8_t *readData,  uint32_t numReadBytes)
{
    int32_t status;
    I2C_Transaction i2cTransaction;

    /* Enable Transceiver */
    I2C_Transaction_init(&i2cTransaction);
    i2cTransaction.targetAddress = targetAddr;
    i2cTransaction.writeBuf = (uint8_t *)&writeData[0];
    i2cTransaction.writeCount = numWriteBytes;
    i2cTransaction.readBuf = (uint8_t *)&readData[0];
    i2cTransaction.readCount = numReadBytes;
    status = I2C_transfer(handle, &i2cTransaction);
    DebugP_assert(SystemP_SUCCESS == status);
}

#endif // AM263_CC
#endif // CMD_CAN
```

启动“PCAN-View”后，将 CAN 适配器设置为图 4-34：

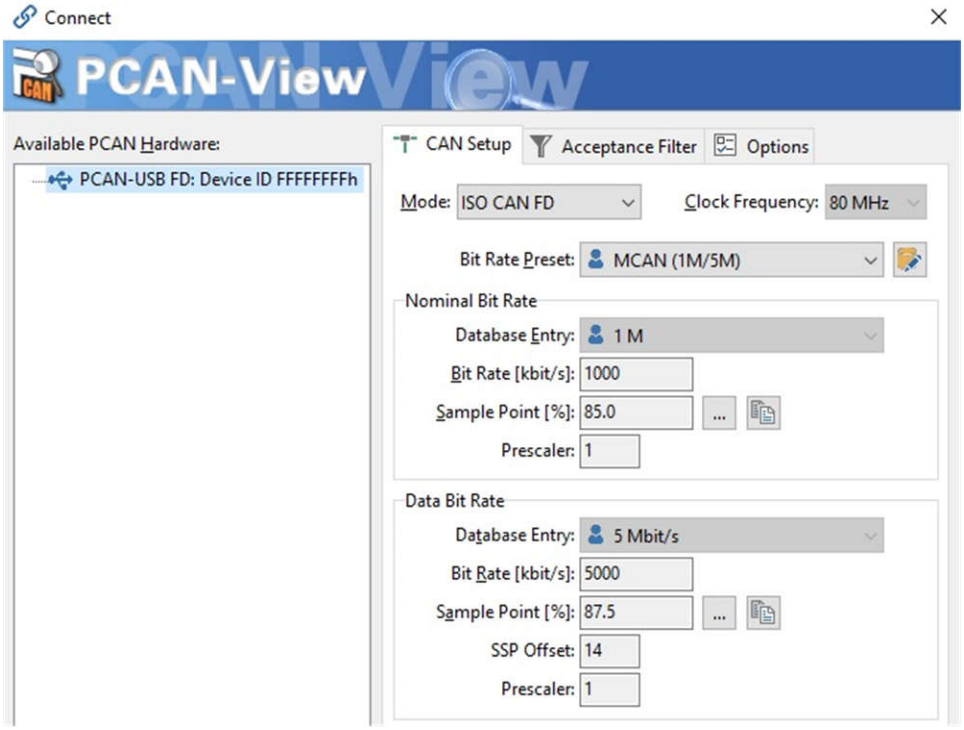


图 4-34. PCAN-View 设置

双击启动 CAN 数据传输，如图 4-35 所示。

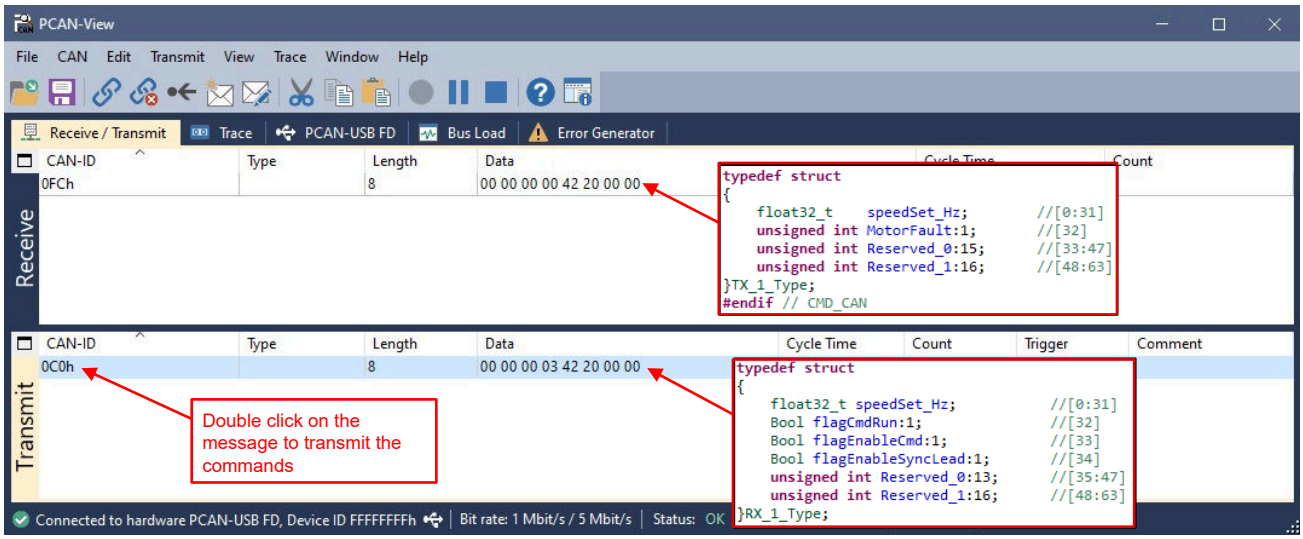


图 4-35. CAN 数据传输开始

如图 4-36 所示，为 *motorVars_M1.flagEnableRunAndIdentify* 配置的值为 1，*motorVars_M1.speedRef_Hz* 设为 40Hz。然后，该速度值通过 CAN 通信传回接收器。

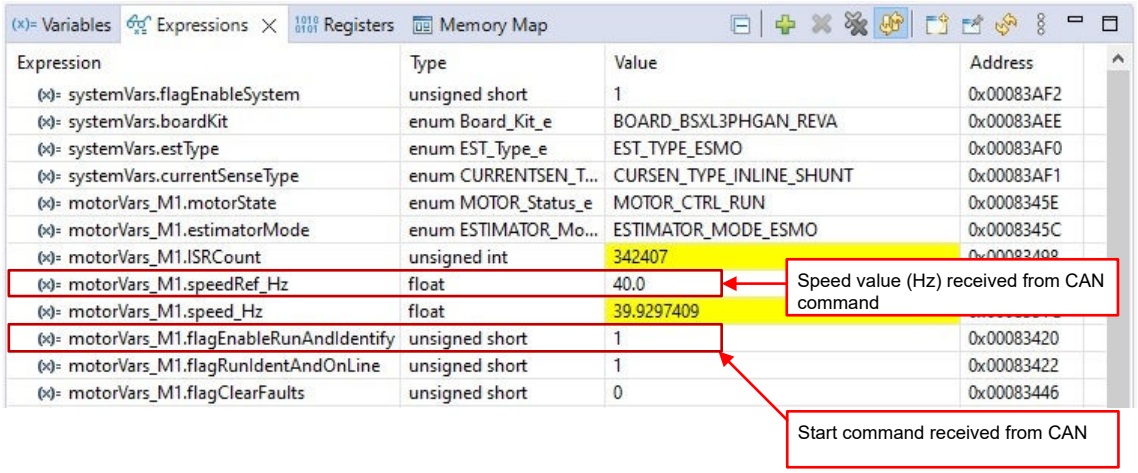


图 4-36. CAN 命令：表达式窗口中的变量

4.5.4 添加 SFRA 功能

德州仪器 (TI) 软件频率响应分析器 (SFRA) 库旨在仅使用软件对电源转换器进行频率响应分析，而无需外部频率响应分析器。优化后的库可用于高频功率转换应用，以识别闭环功率转换器的受控体、闭环和开环增益特性，可用于获取增益裕度、相位裕度和开环增益交叉频率等稳定性信息，来评估控制环路性能。

以数字控制的闭环电源转换器为例，如图 4-37 中所示，其中：

- *H* 是需要控制的受控体的传递函数
- *G* 是数字补偿器
- *GH* 被称为开环传递函数
- *CL* 被称为闭环传递函数，为 GH/(1+GH)
- *r* 是瞬时设定点或转换器的基准
- *Ref* 是直流设定点基准
- *y* 模数转换器 (ADC) 反馈
- *e* 瞬时误差
- *d* 传感器噪声和干扰
- *u* PWM 占空比

闭环系统中补偿器的关键目标可以总结为：

- 确保系统稳定（例如，系统逐渐接近基准）

$$e = \lim_{t \to \infty} e(t) = \lim_{t \to \infty} \frac{r}{(1 + GH)} \to 0 \tag{64}$$

- 系统提供抗扰功能以保持稳健的运行

$$S = \frac{y}{d} = \frac{1}{1 + GH} \to 0 \tag{65}$$

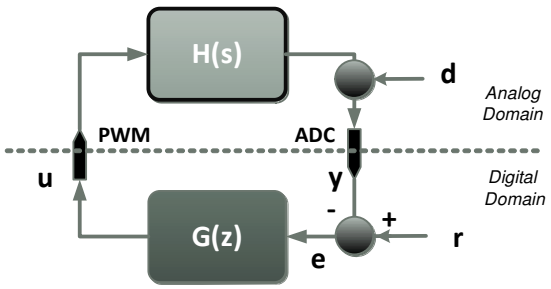


图 4-37. 数控电源转换器

可以通过了解开环传递函数 (GH) 来确定系统是否满足目标，如方程式 64 和方程式 65 所示。

通常出于此目的使用开环传递函数 GH 的波特图，通常使用增益裕度 (gm)、相位裕度 (PM) 和开环增益交叉频率 (Folg_CF) 等量来评价闭环电源转换器的稳定性和稳健性。

闭环传递函数 (GH/(1+GH)) 提供了接近的概念，即系统能够接近命令基准的程度。

SFRA 库可以实现 GH、GH/(1+GH) 和 H 频率响应的软件测量。该数据可用于：

- 验证受控体模型 (H) 或提取受控体模型 (H)
- 为闭环受控体设计一个补偿器(G)
- 通过绘制开环 (GH) 或闭环 [GH÷(1+GH)] 波特图来验证系统的闭环性能

当 GH 和 H 的频率响应携带受控体的信息时，该数据可用于通过定期测量频率响应来评价功率级的运行状况。

SFRA 库基于正弦注入原理，此原理假设注入振幅会使与转换器正常工作点的偏差非常小。SFRA 库可集成到电源转换器的控制代码中，本文档详细介绍了执行此操作的步骤。用于 GH、H 和 CL 计算的所有计算都在 MCU 上完成，GH、H 和 CL 幅度和相位响应的整个数组都存储在控制器上。

一旦集成到代码中，就可以使用 SFRA 库来设计或微调控制器。因此，使用 SFRA 库的典型流程是：

1. 在开环中启动 SFRA 扫描并将数据存储在 Excel 文件中。然后，此信息可用于识别已进行 SFRA 扫描的稳态工作点的受控体模型。
2. 随此工程提供的 MATLAB® 脚本可用于将该数据读取到 MATLAB 中，然后用曲线拟合对传递函数的响应。然后，就可以使用 Sisotool 来设计补偿器。
3. 可以将新的补偿器值从 MATLAB 复制到 Code Composer Studio™ 工程中。
4. 编译带有新系数的代码并将其加载到控制功率级的微控制器中。可以重新运行 SFRA 算法（步骤 1），以通过测量开环增益 GH（在文献中也称为环路增益）来验证闭环系统性能。

总之，TI 的软件频率响应分析器提供了一种以系统方式对功率转换器进行调优的方法，无需外部连接和设备即可对功率转换器进行快速轻松的频率响应分析。由于不使用外部连接，因此可以重复运行 SFRA 以定期评估电源转换器的运行状况并获取诊断信息。

4.5.4.1 操作原理

软件频率响应分析器基于小信号正弦注入原理。如图 4-38 所示，在控制器的基准上注入一个小信号，并计算反馈和控制器输出的频率响应。这提供了闭环系统的受控体频率响应特性和开环频率响应。

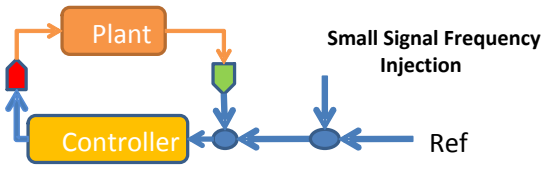


图 4-38. SFRA 工作原理

4.5.4.2 对象定义

SFRA 库定义了基于浮点的 SFRA 结构，如下所述：

```
typedef struct{
  float32_t *h_magVect;     //!< Plant Mag SFRA Vector
  float32_t *h_phaseVect;   //!< Plant Phase SFRA Vector
  float32_t *gh_magVect;    //!< Open Loop Mag SFRA Vector
  float32_t *gh_phaseVect;  //!< Open Loop Phase SFRA Vector
  float32_t *cl_magVect;    //!< Closed Loop Mag SFRA Vector
  float32_t *cl_phaseVect;  //!< Closed Loop Phase SFRA Vector
  float32_t *freqVect;      //!< Frequency Vector
  float32_t amplitude;      //!< Injection Amplitude
  float32_t isrFreq;        //!< SFRA ISR frequency
  float32_t freqStart;      //!< Start frequency of SFRA sweep
  float32_t freqStep;       //!< Log space between frequency points (optional)
  int16_t start;            //!< Command to start SFRA
  int16_t state;            //!< State of SFRA
  int16_t status;           //!< Status of SFRA
```

```
  int16_t vecLength;        //!< No. of Points in the SFRA
  int16_t freqIndex;        //!< Index of the frequency vector
  int16_t storeH;           //!< Flag to indicate if H vector is stored
  int16_t storeGH;          //!< Flag to indicate if GH vector is stored
  int16_t storeCL;          //!< Flag to indicate if CL vector is stored
  int16_t speed;            //!< variable to change the speed of the sweep
}SFRA_F32;
```

4.5.4.3 模块接口定义

表 4-7. 浮点模块接口定义

模块元素名称	类型	说明	可接受的范围
h_magVect、gh_magVect、cl_magVect	输入	指向由 SFRA 存储 H、GH 和 CL 测量值大小的数组的指针。如果您不希望 SFRA 保存该向量，则传递 NULL。	指向 32 位位置的指针，该位置以单精度（32 位）浮点方式存储幅度向量的值
h_phaseVect、gh_phaseVect、cl_phaseVect	输入	指向由 SFRA 存储 H、GH 和 CL 测量值相位的数组的指针。如果您不希望 SFRA 保存该向量，则传递 NULL。	指向 32 位位置的指针，该位置将相位向量的值存储在单精度（32 位）浮点中
freqVect	输入	指向执行 SFRA 的频率值数组的指针。	指向 32 位位置的指针，该位置以单精度（32 位）浮点方式存储频率向量的值
振幅	输入	以 pu 为单位的小信号注入幅值。	单精度（32 位）浮点 (-1,1)
isrFreq	输入	调用 SFRA 例程时的频率。	单精度（32 位）浮点
freqStart	输入	第一个频率扫描数据点的频率。	单精度（32 位）浮点
freqStep	输入	10^(1/(每十倍频的步长数))。	单精度（32 位）浮点
start	输入	此命令用于启动 SFRA。	int16_t
state	输出	SFRA 状态。当 SFRA 注入正在进行时为非零，如果 SFRA 注入未激活/未正在进行，则为“0”。	int16_t
status	输出	SFRA 状态。“1”表示 SFRA 注入正在进行，如果 SFRA 注入未激活/未正在进行，则为“0”。	int16_t
vecLength	输入	执行 SFRA 的点数。	int16_t
freqIndex	输出	正在执行 SFRA 的 freqVect 的频率索引号。	int16_t (0-vecLength)
storeH	输出	反映 SFRA 配置，如果为 1，则存储 H 矢量；如果为零，则不存储 H 矢量。在 SFRA 配置期间为 H mag 或相位矢量传递 NULL 矢量时会发生这种情况。	int16_t（0 或 1）
storeGH	输出	反映 SFRA 配置，如果为 1，则存储 GH 矢量；如果为零，则不存储 GH 矢量。在 SFRA 配置期间为 GH mag 或相位矢量传递 NULL 矢量时会发生这种情况。	int16_t（0 或 1）
storeCL	输出	反映 SFRA 配置，如果为 1，则存储 CL 矢量；如果为零，则不存储 CL 矢量。在 SFRA 配置期间为 CL mag 或相位矢量传递 NULL 矢量时会发生这种情况。	int16_t（0 或 1）
speed	输入	用于更改扫描速度，需要大于 1。如果为 1，STB 示例模板扫描大约需要 58 秒。系统中的实际速度取决于所测量的频率点以及调用 SFRA 模块时采用的 ISR 速率。速度数值越高，扫描越慢。	int16_t（大于 1）

4.5.4.4 使用 SFRA

使用以下步骤将 SFRA 集成到工程中：

1. 要启用 SFRA 功能，必须在工程属性中添加预定义符号 SFRA_ENABLE，如图 4-2 所示。
2. 若要启动 SFRA 扫描，请将 SFRA 对象放入监视窗口中。
3. 当您希望 SFRA 扫描开始时，请将 SFRA_OBJ.start 写入 1，如图 4-39 所示。

sfra1	struct SFRA_F32	{h_magVect=0x00083640 {15.5954409},h_...	0x000838A8
h_magVect	float *	0x00083640 {15.5954409}	0x000838A8
h_phaseVect	float *	0x00083698 {0.306601793}	0x000838AC
gh_magVect	float *	0x000836F0 {7.98486328}	0x000838B0
gh_phaseVect	float *	0x00083748 {-75.6564407}	0x000838B4
cl_magVect	float *	0x000837A0 {-1.32463789}	0x000838B8
cl_phaseVect	float *	0x000837F8 {-19.3731956}	0x000838BC
freqVect	float *	0x00083850 {20.0}	0x000838C0
amplitude	float	0.00499999989	0x000838C4
isrFreq	float	15000.0	0x000838C8
freqStart	float	20.0	0x000838CC
freqStep	float	1.26335502	0x000838D0
start	short	0	0x000838D4
state	short	0	0x000838D6
status	short	0	0x000838D8
vecLength	short	22	0x000838DA
freqIndex	short	22	0x000838DC
storeH	short	1	0x000838DE
storeGH	short	1	0x000838E0
storeCL	short	1	0x000838E2
speed	short	1	0x000838E4

图 4-39. 启动 SFRA 功能

4. 监视 SFRA_OBJ.FreqIndex 变量；该变量会在执行 SFRA 扫描而逐渐增加。
5. SFRA_OBJ.FreqIndex 达到 Vec_Length 后，SFRA 扫描即完成。

freqVect	float[22]	[20.0,25.2671013,31.92132,40.327961,50.94...	0x00083850
olMagVect	float[22]	[7.98486328,6.04782486,4.30153942,2.6732...	0x000836F0
plantMagVect	float[22]	[15.5954409,15.4647503,15.5659208,15.621...	0x00083640
plantPhaseVect	float[22]	[0.306601793,-1.85650432,-3.8041153,-4.7...	0x00083698

图 4-40. SFRA 数据数组

6. 在 SFRA 初始化过程中，开环和受控体幅度及相位存储在被调用的数组中。

```
__attribute__ ((section(".sfradata"))) float32_t plantMagVect[SFRA_FREQ_LENGTH];
__attribute__ ((section(".sfradata"))) float32_t plantPhaseVect[SFRA_FREQ_LENGTH];
__attribute__ ((section(".sfradata"))) float32_t olMagVect[SFRA_FREQ_LENGTH];
__attribute__ ((section(".sfradata"))) float32_t olPhaseVect[SFRA_FREQ_LENGTH];
__attribute__ ((section(".sfradata"))) float32_t freqVect[SFRA_FREQ_LENGTH];
```

7. 将它们放入监视窗口中以检查和研究响应。
8. 扫描完成后，依次点击 CCS 内的“View”->“MemoryBrowser”。
9. 在“Memory Browser”中，输入 &freqVect 来查看频率矢量并选择 32 位浮点，

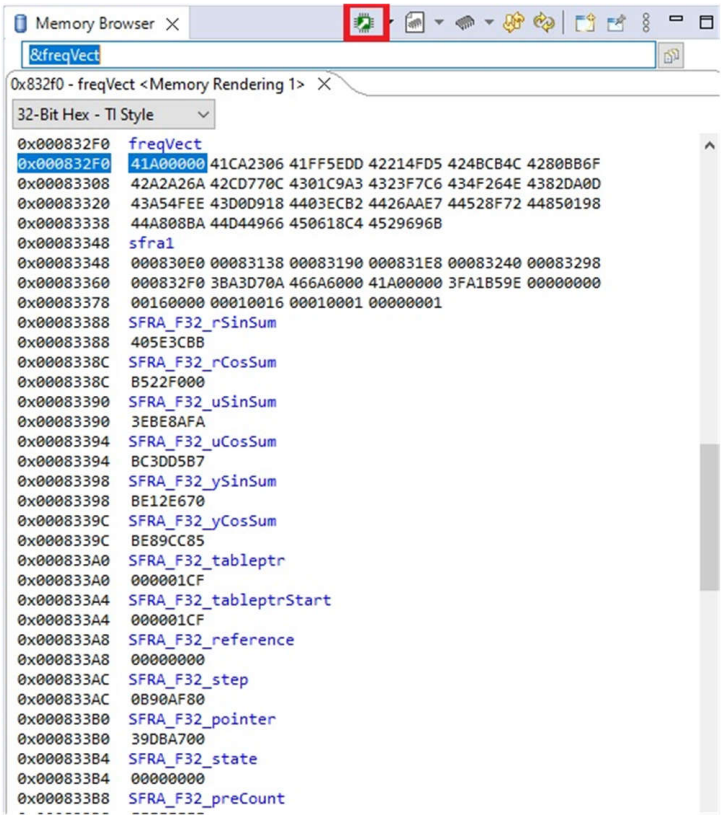


图 4-41. 已存储的 SFRA 矢量的“Memory Browser”视图

10. 点击“Save Memory”，如图 4-41 中所示。

11. 将会出现一个弹出窗口。选择 TI 数据并在您喜欢的位置指定文件名 *.dat。

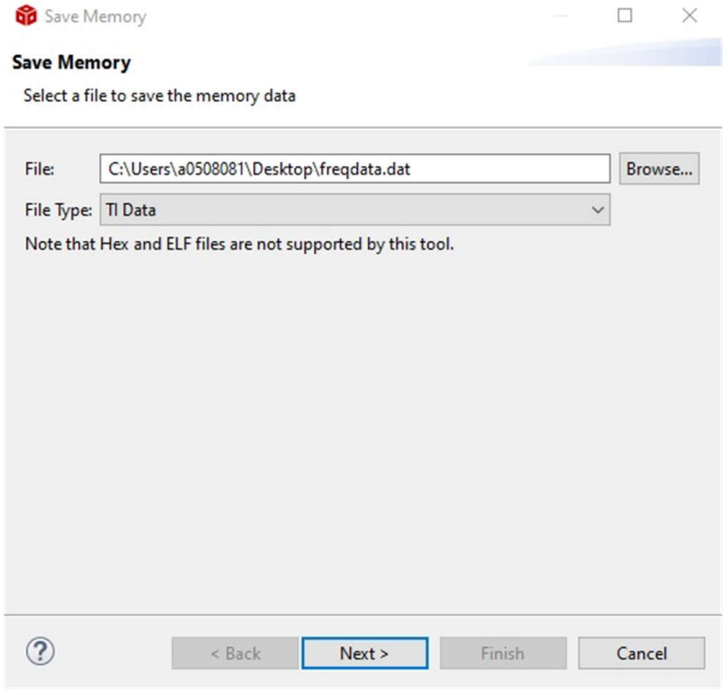


图 4-42. “Save Memory”弹出窗口

12. 点击“Next”，从 Memory Browser 中指定数组起始地址，然后指定长度。

13. 确保选择了 32 位浮点。点击“Finish”。

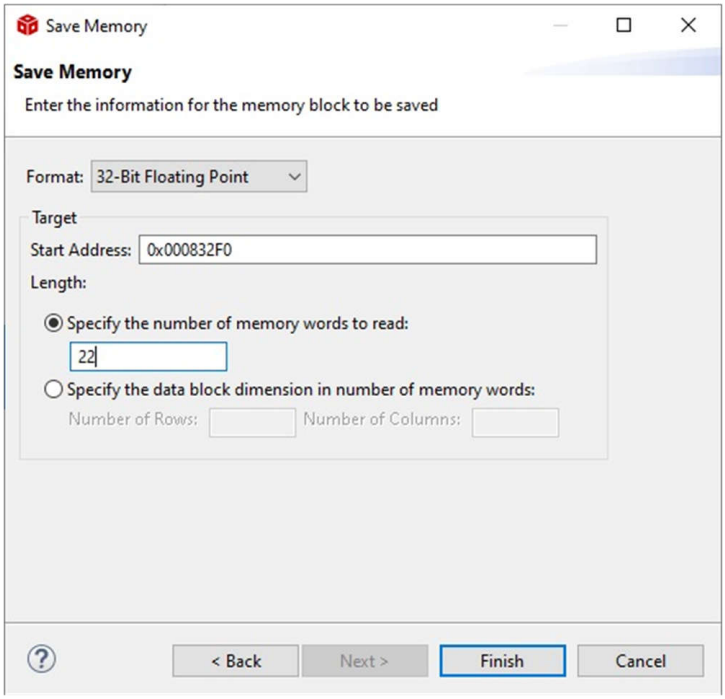


图 4-43. “Save Memory”选项

14. 这会将数据保存在 *.dat 文件中。
15. 对 plantMagVect、plantPhaseVect、olMagVect、olPhaseVect 重复此步骤，这样您就有 5 个 *.dat 文件。
16. 如果您要在 MATLAB 或其他工具中使用此数据，可以将数据填充到 Excel 文件中。
17. 在 Excel 中打开位于 <project directory>\libraries\SFRA\scripts 下的 SFRA.xlsx 文件。
18. 您可以选择重命名并保存文件。
19. 该 Excel 工作表包含五列，第一列是频率数据。
20. 打开保存的 *dat 文件。

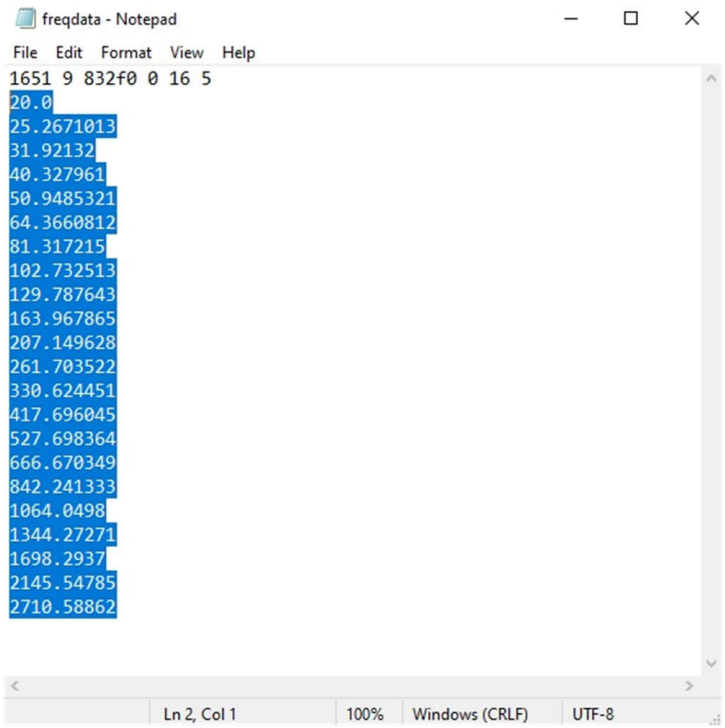
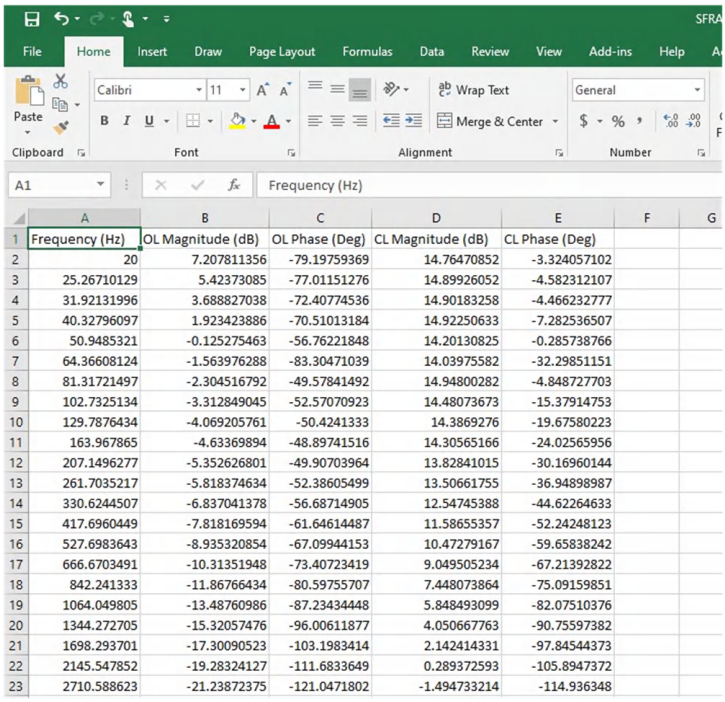


图 4-44. 从 .dat 文件中选择要放入 Excel 的数据

21. 选择从第二行开始到文件末尾的数据，然后按 Ctrl+C 复制数据。
22. 打开 Excel 文件，转到相应向量下的第一个元素，然后按 Ctrl+V 复制数组。



	A	B	C	D	E
1	Frequency (Hz)	OL Magnitude (dB)	OL Phase (Deg)	CL Magnitude (dB)	CL Phase (Deg)
2	20	7.207811356	-79.19759369	14.76470852	-3.324057102
3	25.26710129	5.42373085	-77.01151276	14.89926052	-4.582312107
4	31.92131996	3.688827038	-72.40774536	14.90183258	-4.466232777
5	40.32796097	1.923423886	-70.51013184	14.92250633	-7.282536507
6	50.9485321	-0.125275463	-56.76221848	14.20130825	-0.285738766
7	64.36608124	-1.563976288	-83.30471039	14.03975582	-32.29851151
8	81.31721497	-2.304516792	-49.57841492	14.94800282	-4.848727703
9	102.7325134	-3.312849045	-52.57070923	14.48073673	-15.37914753
10	129.7876434	-4.069205761	-50.4241333	14.3869276	-19.67580223
11	163.967865	-4.63369894	-48.89741516	14.30565166	-24.02565956
12	207.1496277	-5.352626801	-49.90703964	13.82841015	-30.16960144
13	261.7035217	-5.818374634	-52.38605499	13.50661755	-36.94898987
14	330.6244507	-6.837041378	-56.68714905	12.54745388	-44.62264633
15	417.6960449	-7.818169594	-61.64614487	11.58655357	-52.24248123
16	527.6983643	-8.935320854	-67.09944153	10.47279167	-59.65838242
17	666.6703491	-10.31351948	-73.40723419	9.049505234	-67.21392822
18	842.241333	-11.86766434	-80.59755707	7.448073864	-75.09159851
19	1064.049805	-13.48760986	-87.23434448	5.848493099	-82.07510376
20	1344.272705	-15.32057476	-96.00611877	4.050667763	-90.75597382
21	1698.293701	-17.30090523	-103.1983414	2.142414331	-97.84544373
22	2145.547852	-19.28324127	-111.6833649	0.289372593	-105.8947372
23	2710.588623	-21.23872375	-121.0471802	-1.494733214	-114.936348

图 4-45. 在 Excel 文件中复制的 SFRA 数据

23. 对每列重复这些步骤。
24. 为所有五列更新 Excel 文件后，使用 MATLAB 脚本导入 SFRA 数据。然后，使用 sisotool 中的脚本设计补偿器并进行稳定性分析。

4.6 构建定制板

4.6.1 构建新的定制板

本节讨论了用户如何设计用于驱动电机的应用板，以及如何迁移此工程以与自己的板一起使用。

4.6.1.1 硬件设置

如果使用定制板，请确保微控制器和栅极驱动器的电源正确，并且 JTAG 仿真器可以成功连接。按照以下部分中的说明修改参考代码以与定制电路板兼容，然后运行从构建级别 1 开始的代码，并按照节 4.4 中所示的方法构建级别 4。

4.6.1.2 将参考代码迁移到定制板

要将参考代码迁移到新的 TI 电机驱动器套件或定制电路板，用户需要根据电机驱动器电路在 *user_mtr1.h* 文件中配置硬件参数和电机控制参数，在 *AM263_xxx.syscfg*、*hal.h* 和 *hal.c* 文件中配置相关外设，如以下各节所述。

下面的方框图总结了用于配置电机控制设置和 TI MCU 外设的函数调用（图 4-46）。

在这个项目中，有几个仅被调用一次的 HAL 函数，与硬件的配置相关。所有这些函数都涉及外设或电机驱动器 IC 的配置。

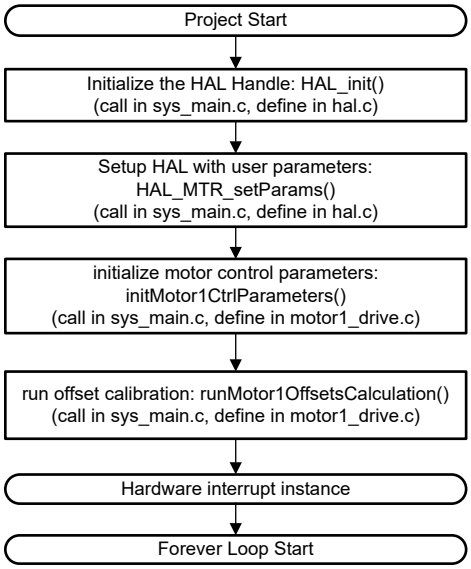


图 4-46. HAL 配置和电机控制设置方框图

4.6.1.2.1 设置硬件板参数

user_mtr1.h 文件用于存储所有用户参数以进行电机控制。模数转换器输入端的最大相电流和相电压值取决于硬件，必须基于电流和电压检测电路以及 ADC 输入端口的调节。相电流传感器和相电压传感器的数量也在 *user_mtr1.h* 文件中定义。这些值取决于硬件。

user_mtr1.h 文件中定义的所有可配置参数都可以使用 `Motor Control Parameters Calculation.xlsx` Excel® 电子表格进行计算。此文件包含在工程文件夹中：`\examples\universal_motorcontrol_lab\doc`。将标记为 ***粗体*** 的参数复制到 *user_mtr1.h* 文件中，如以下代码所示。

```
//! \brief Defines the maximum voltage at the AD converter
#define USER_M1_ADC_FULL_SCALE_VOLTAGE_V         (57.52845691f)

//! \brief Defines the analog voltage filter pole location, Hz
#define USER_M1_VOLTAGE_FILTER_POLE_Hz           (680.4839141f)     // 47nF

//! \brief Defines the maximum current at the AD converter
#define USER_M1_ADC_FULL_SCALE_CURRENT_A         (47.14285714f)     // gain=10
```

4.6.1.2.2 修改电机控制参数

user_mtr1.h 中提供的用于 PMSM 电机的参数如以下代码所示。电机参数可以从电机数据表中识别。

```
#define USER_MOTOR1_TYPE                    MOTOR_TYPE_PM
#define USER_MOTOR1_NUM_POLE_PAIRS          (4)

#define USER_MOTOR1_Rs_Ohm                  (0.38157931f)
#define USER_MOTOR1_Ls_d_H                  (0.000188295482f)
#define USER_MOTOR1_Ls_q_H                  (0.000188295482f)
#define USER_MOTOR1_RATED_FLUX_VpHz         (0.0396642499f)
#define USER_MOTOR1_MAX_CURRENT_A           (6.0f)
```

4.6.1.2.3 更改引脚分配

AM263_xxx.syscfg 文件配置 GPIO 引脚的功能，并根据使用的硬件电机驱动器板/套件设置指定引脚的方向和模式。要修改定制板（当前没有通用实验室代码支持的 TI 电机驱动器 EVM）的代码，或用于不同 TI MCU 的代码，需要更改这些 GPIO 分配以与电机驱动器板正确对应。

4.6.1.2.4 配置 PWM 模块

SysConfig 文件配置 PWM 通道。*hal.h* 文件中定义了用于电机控制器 PWM 输入的 PWM 通道的基地址，并在 *hal.c* 文件中为 PWM 句柄分配了基地址。LP-AM263 与 BOOSTXL-3PHGANINV 之间 PWM 信号的连接图如图 4-47 所示。

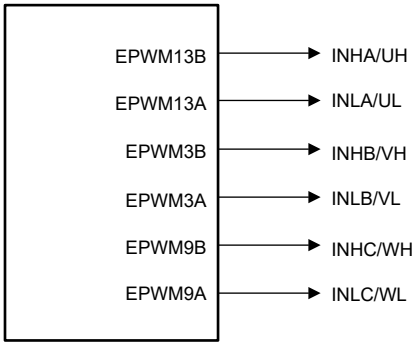


图 4-47. PWM 连接图

用于配置 PWM 信号的代码如以下所示，取自 *.syscfg*、*hal.h* 和 *hal.c* 文件。

1. PWM 模块的基地址在 *hal.h* 文件中定义，如下所示。

```
#define MTR1_PWM_U_BASE         CONFIG_EPWM13_BASE_ADDR
#define MTR1_PWM_V_BASE         CONFIG_EPWM3_BASE_ADDR
#define MTR1_PWM_W_BASE         CONFIG_EPWM9_BASE_ADDR
```

2. 在 *.syscfg* 文件中将 GPIO 设置为 PWM 输出。

图 4-48. PWM 模块的 GPIO 配置

3. 以下代码将 PWM 模块的相应基地址分配给 *hal.c* 文件中 HAL_MTR1_init() 函数的 PWM 句柄。在调整代码以适应新电路板或 TI MCU 时，以下代码块演示了如何在代码中初始化 PWM 句柄。

```
// initialize PWM handles for Motor 1
obj->pwmHandle[0] = MTR1_PWM_U_BASE;        //!< the PWM handle
obj->pwmHandle[1] = MTR1_PWM_V_BASE;        //!< the PWM handle
obj->pwmHandle[2] = MTR1_PWM_W_BASE;        //!< the PWM handle
```

4. 图 4-49 显示了 EPWM 时基配置。A 相的同步输出脉冲用作其他 PWM 的同步输入脉冲源。

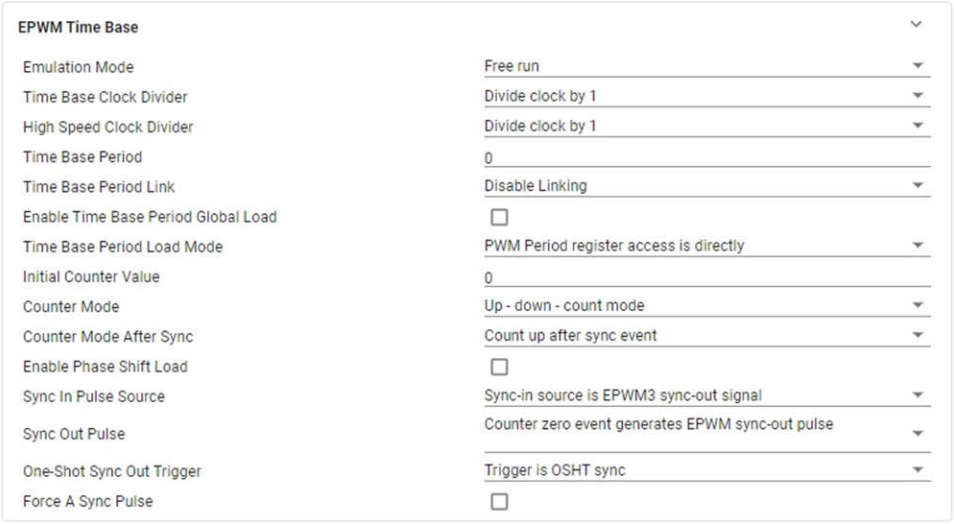


图 4-49. EPWM 时基配置

EPWM 动作限定符配置展示了 LP-AM263 和 BOOSTXL-3PHGANINV 组合的 EPWM 动作限定器输出事件配置。PWM 动作限定符输出需要根据硬件板进行设置。

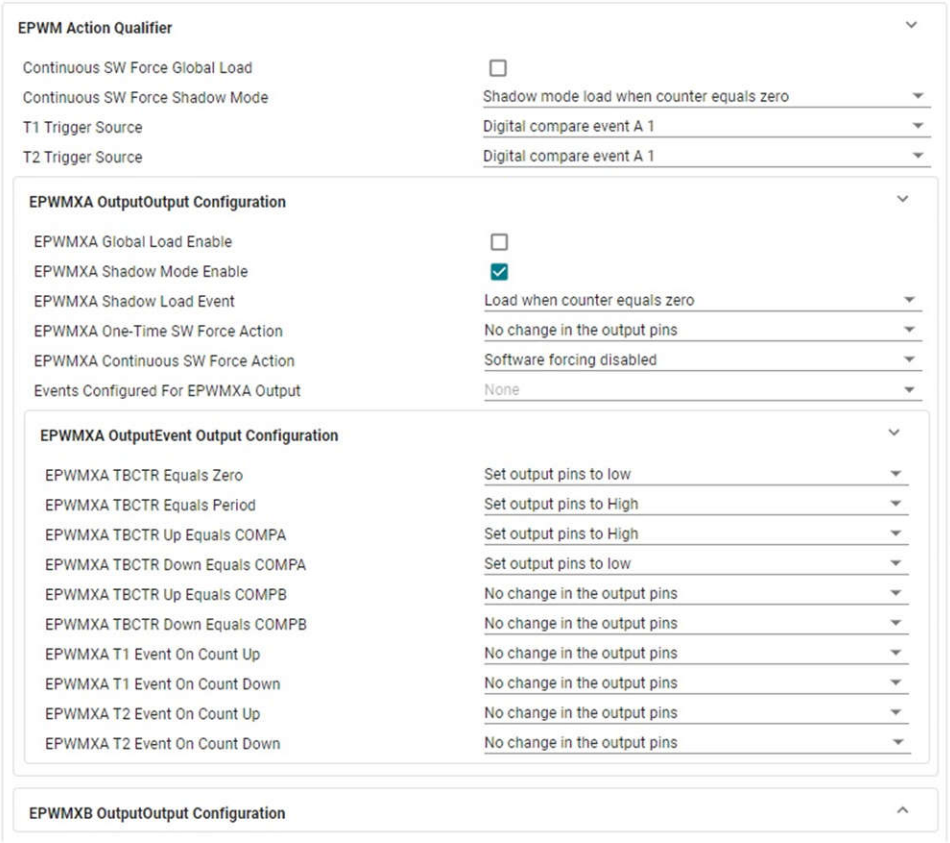


图 4-50. EPWM 动作限定符配置

图 4-51 展示了 LP-AM263 的 EPWM 死区配置。检查 EPWMxA-B 的交换输出以匹配 LaunchPad™ 和 Booster Pack™ 中的高侧和低侧 PWM。

图 4-51. EPWM 死区配置

4.6.1.2.5 配置 ADC 模块

与前面的 PWM 部分类似，对于通用电机控制工程不支持的定制电路板或 TI 电机控制套件，也可以更改 ADC 连接。*.syscfg* 文件配置 ADC 通道，使其与电机驱动器板正确对应。例如，LP-AM263 和 BOOSTXL-3PHGANINV 组合的连接图如图 4-52 所示。下面的步骤描述了 ADC 模块的配置。

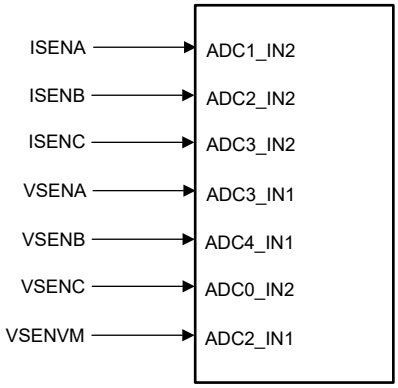


图 4-52. ADC 连接图

1. 以下代码演示了 *hal.h* 文件中 ADC 模块的基地址、分配的通道和 SOC 的定义。请注意，对于 SOC 编号，多个 ADC 可以与同一 SOC 编号相关联，只要它们属于不同的 ADC 模块（在以下情况中，是模块 A 和模块 C）。尝试对所有电流和所有电压进行采样时使其尽可能靠近，因此在配置 SOC 编号时应注意这一点。在调整代码以适应新电路板或 TI MCU 时，不需要更改以下代码，以下代码仅用于演示如何初始化 ADC 以及可以在 *.syscfg* 文件中完成更改。

```
#define MTR1_IU_ADC_BASE         CONFIG_ADC1_BASE_ADDR //J7.67 ADC1_AIN2
#define MTR1_IV_ADC_BASE         CONFIG_ADC2_BASE_ADDR //J7.68 ADC2_AIN2
#define MTR1_IW_ADC_BASE         CONFIG_ADC3_BASE_ADDR //J7.69 ADC3_AIN2
#define MTR1_VU_ADC_BASE         CONFIG_ADC3_BASE_ADDR //J7.64 ADC3_AIN1
#define MTR1_VV_ADC_BASE         CONFIG_ADC4_BASE_ADDR //J7.65 ADC4_AIN1
#define MTR1_VW_ADC_BASE         CONFIG_ADC0_BASE_ADDR //J7.66 ADC0_AIN2
#define MTR1_VDC_ADC_BASE        CONFIG_ADC2_BASE_ADDR //J7.63 ADC2_AIN1

#define MTR1_IU_ADCRES_BASE      CONFIG_ADC1_RESULT_BASE_ADDR
#define MTR1_IV_ADCRES_BASE      CONFIG_ADC2_RESULT_BASE_ADDR
#define MTR1_IW_ADCRES_BASE      CONFIG_ADC3_RESULT_BASE_ADDR
#define MTR1_VU_ADCRES_BASE      CONFIG_ADC3_RESULT_BASE_ADDR
#define MTR1_VV_ADCRES_BASE      CONFIG_ADC4_RESULT_BASE_ADDR
#define MTR1_VW_ADCRES_BASE      CONFIG_ADC0_RESULT_BASE_ADDR
#define MTR1_VDC_ADCRES_BASE     CONFIG_ADC2_RESULT_BASE_ADDR

#define MTR1_IU_ADC_CH_NUM       ADC_CH_ADCIN2
#define MTR1_IV_ADC_CH_NUM       ADC_CH_ADCIN2
#define MTR1_IW_ADC_CH_NUM       ADC_CH_ADCIN2
#define MTR1_VU_ADC_CH_NUM       ADC_CH_ADCIN1
#define MTR1_VV_ADC_CH_NUM       ADC_CH_ADCIN1
#define MTR1_VW_ADC_CH_NUM       ADC_CH_ADCIN2
#define MTR1_VDC_ADC_CH_NUM      ADC_CH_ADCIN1

#define MTR1_IU_ADC_SOC_NUM      ADC_SOC_NUMBER0         // SOC0-PPB1
#define MTR1_IV_ADC_SOC_NUM      ADC_SOC_NUMBER0         // SOC0-PPB1
#define MTR1_IW_ADC_SOC_NUM      ADC_SOC_NUMBER0         // SOC0-PPB2
#define MTR1_VU_ADC_SOC_NUM      ADC_SOC_NUMBER1         // SOC1
#define MTR1_VV_ADC_SOC_NUM      ADC_SOC_NUMBER1         // SOC1
#define MTR1_VW_ADC_SOC_NUM      ADC_SOC_NUMBER1         // SOC1
#define MTR1_VDC_ADC_SOC_NUM     ADC_SOC_NUMBER1         // SOC1

#define MTR1_IU_ADC_PPB_NUM      ADC_PPB_NUMBER1         // SOC0-PPB1
#define MTR1_IV_ADC_PPB_NUM      ADC_PPB_NUMBER1         // SOC0-PPB1
#define MTR1_IW_ADC_PPB_NUM      ADC_PPB_NUMBER1         // SOC0-PPB2
```

2. 图 4-53 展示了 *.syscfg* 文件中 ISR 的中断源定义。

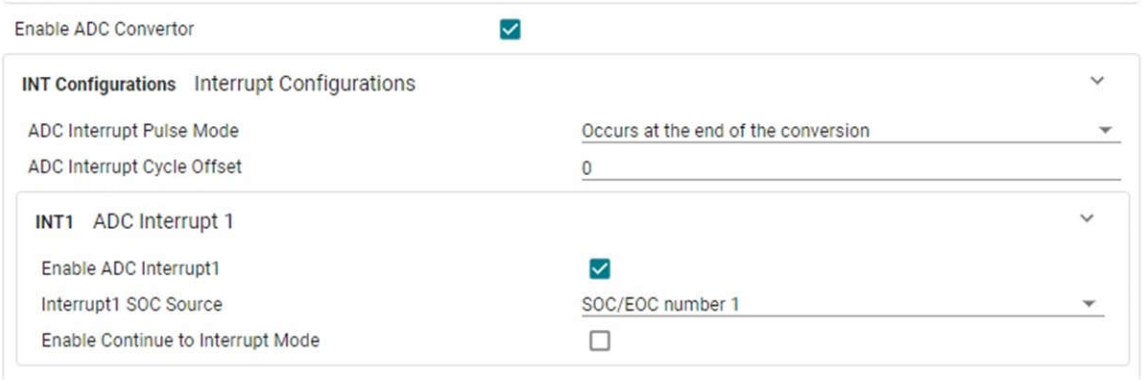


图 4-53. ADC 中断配置

3. 图 4-54 定义 ADC 转换开始触发源。此 ePWM SOC 触发信号必须对应于代码中启用的同一 ePWM SOC，以及与 pwmHandle[0] 相关联的同一 ePWM。在这种情况下，EPWM3 A 用作 ADC 的 SOC。

图 4-54. ADC 转换开始配置

4.6.1.2.6 配置 CMPSS 模块

CMPSS 模块用于对相电流进行过流监测。可使用 CMPSS DAC 设置阈值，如果电流检测放大器的输出超过该阈值，则 CMPSS 输出会发生跳变。

如果使用定制电机驱动器板，或者将代码迁移到当前通用电机控制项目不支持的 TI MCU 或 TI 电机驱动器 EVM，则需要根据电机驱动器和 TI MCU 连接在 *.syscfg* 文件中正确修改 ADC 引脚和 CMPSS 模块之间的连接。有关 CMPSS 模块内部连接的更多详细信息，请参阅 AM263x Sitara™ 微控制器数据表 中的 *ADC 信号说明* 表。

.syscfg 文件根据所使用的电机驱动器板配置 CMPSS 模块。例如，LP-AM263 和 BOOSTXL-3PHGANINV 之间的连接图如图 4-55 所示。图 4-56 展示了 CMPSSA 方框图。CMPSSA 额外支持 INH 和 INL 作为 COMPL 正信号的多路复用输入。

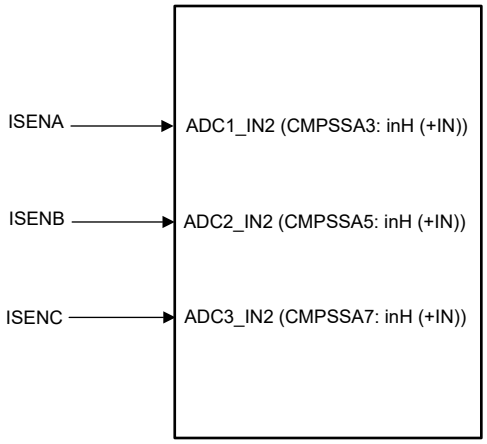


图 4-55. CMPSS 连接图

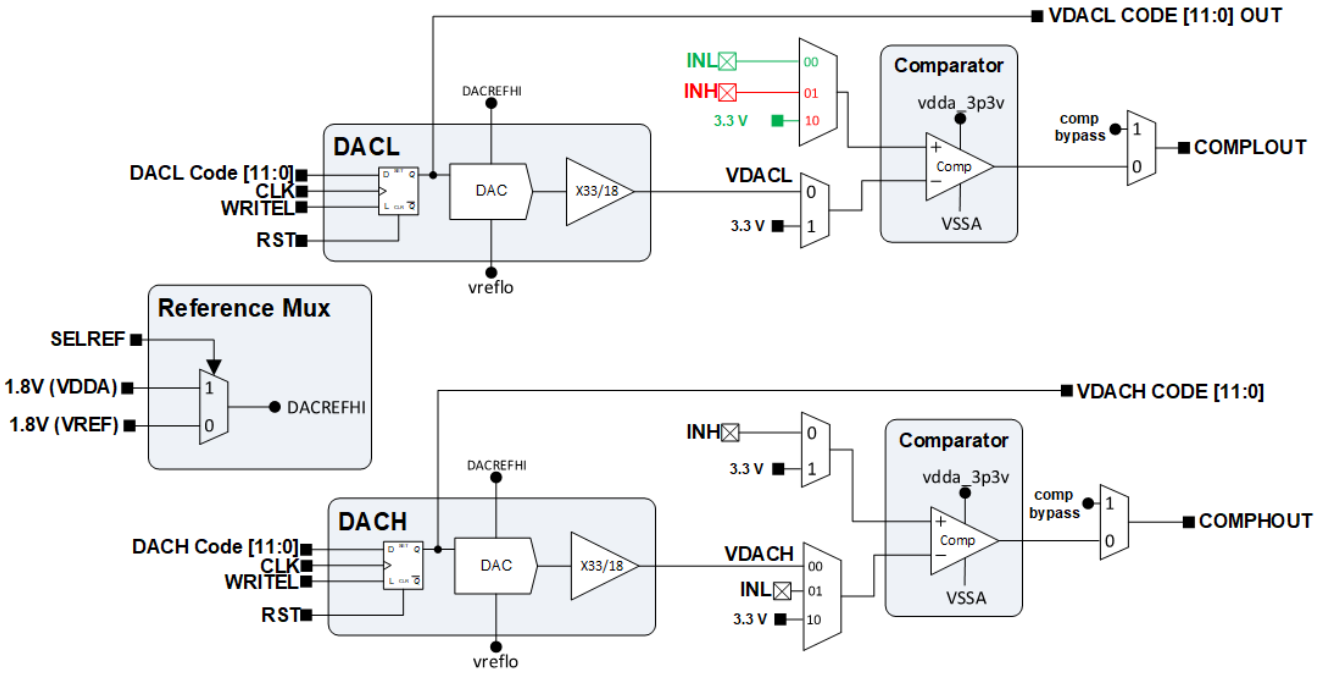


图 4-56. CMPSSA 方框图

每个 CMPSS 比较器都有一个高电平比较器和一个低电平比较器，因此信号必须适当地多路复用到所需比较器的所需输入。有关这些连接的更多信息，请参阅正在使用的微控制器数据表中的“模拟引脚和内部连接”表。图 4-57 展示了 LP-AM263 和 BOOSTXL-3PHGANINV 组合的 CMPSS 比较器配置，用于对相电流进行窗口比较。DAC 值根据定义的最大电流在代码中进行更新。请注意，对于 AM263x 器件，ADCx_AIN1 和 ADCx_AIN3 仅连接到 (INL)，这会限制正负过流跳闸。对于 TMDSHVMTRINSPIN 和 TMDSCNCD263 的组合，用于 U 相电流测量的 ADC 连接到 ADC1_AIN3。

另请注意，要在 LaunchPad 或 EVM 控制卡中选择与 DAC 基准电压匹配的正确电压基准。例如，在 AM263x LaunchPad 中，使用 DAC VREF 开关 (S1) 选择 AM263x 片上 LDO。

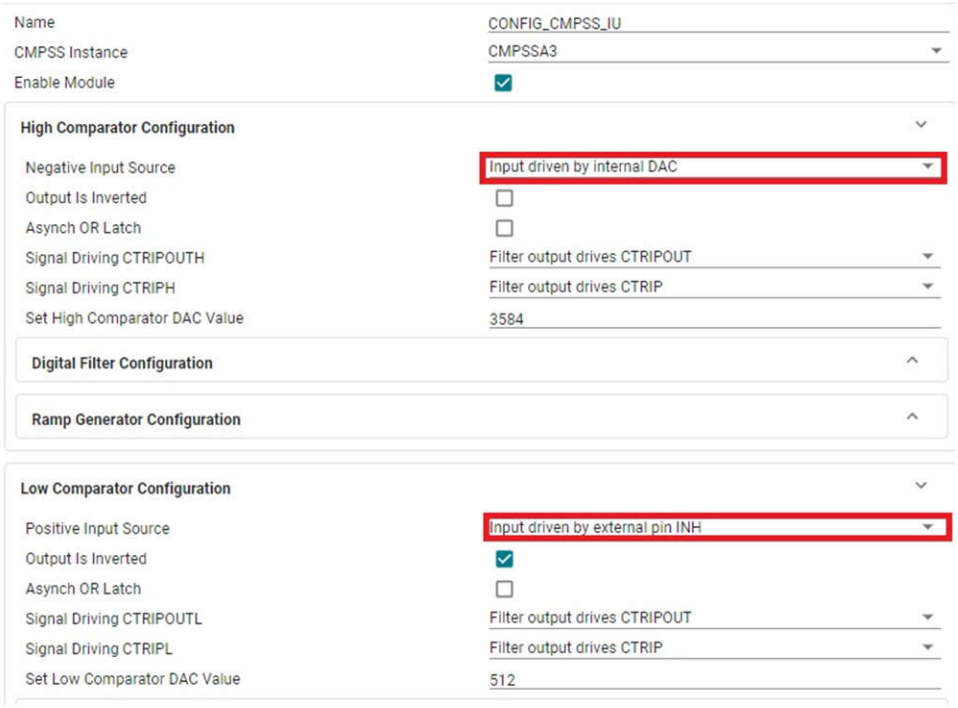


图 4-57. CMPSS 比较器配置

图 4-58 展示了为 EPWM XABR 选择相应的 CMPSS CTRIPL 和 CTRIPH，以在发生过流和欠流时生成跳闸。

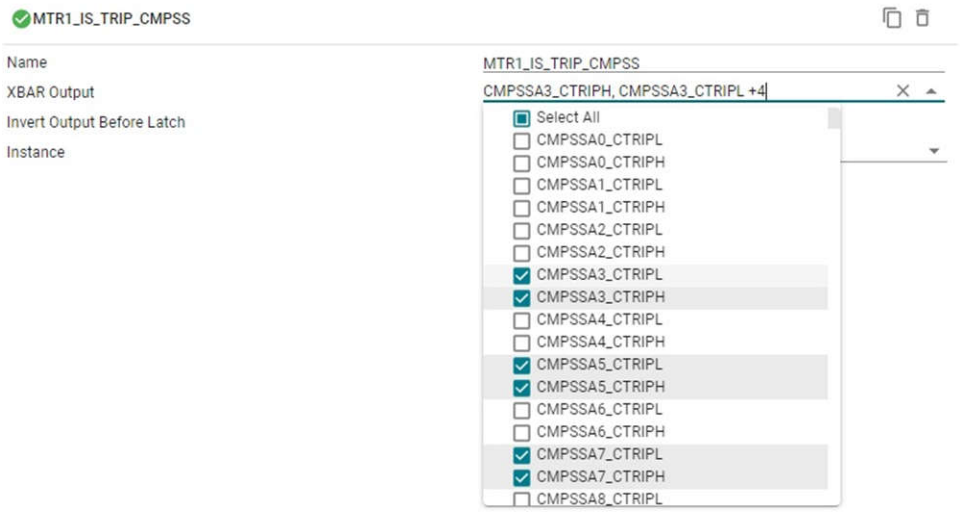


图 4-58. EPWM XBAR 配置

5 通用德州仪器 (TI) 高压评估模块 (TI HV EVM) 用户安全指南

务必遵循 TI 的设置和应用说明，包括在建议的电气额定电压和功率限制范围内使用所有接口元件。务必采取电气安全防护措施，这样有助于确保自身和周围人员的人身安全。有关更多信息，请联系 TI 的产品信息中心。

警告

务必遵循警告和说明，否则可能引发电击和灼伤危险，进而造成财产损失或人员伤亡。

TI HV EVM 一词是指通常以开放式框架、敞开式印刷电路板装配形式提供的电子器件。TI HV EVM *严格用于开发实验室环境，仅供了解开发和应用高压电路相关电气安全风险且接受过专门培训、具有专业知识背景的合格专业用户使用。德州仪器 (TI) 严禁任何其他不合规的使用和/或应用。*如果您不合要求，则必须立即停止进一步使用 HV EVM。

1. 工作区安全
   a. 保持工作区整洁有序。
   b. 每次电路通电时，必须有合格的观察员在场监督。
   c. TI HV EVM 及接口电子元件通电区域必须设有有效的防护栏和标识；指示可能存在高压操作，以避免意外接触。
   d. 开发环境中使用的所有接口电路、电源、评估模块、仪器、仪表、示波器和其他相关装置如果超过 50Vrms/75VDC，则必须置于紧急断电 EPO 保护电源板内。
   e. 使用稳定且不导电的工作台。
   f. 使用充分绝缘的夹钳和导线来连接测量探针和仪器。尽量不要徒手进行测试。
2. 电气安全
   作为一项预防措施，假定整个 EVM 可能存在用户可完全接触到的高电压。
   a. 执行任何电气测量或其他诊断测量之前，需切断 TI HV EVM 及其全部输入、输出和电气负载的电源。再次确认 TI HV EVM 已安全断电。
   b. 确认 EVM 断电后，根据所需的电路配置、接线、测量设备连接和其他应用需求执行进一步操作，同时仍假定 EVM 电路和测量仪器均带电。
   c. EVM 准备就绪后，根据需要将 EVM 通电。

   警告

   EVM 通电后，请勿触摸 EVM 或其电路，因为它们可能存在高压，会造成电击危险。
3. 人身安全
   a. 穿戴人员防护装备（例如乳胶手套或具有侧护板的安全眼镜）或将 EVM 放置于带有联锁装置的透明塑料箱中，避免意外接触。

安全使用限制条件：

勿将 EVM 作为整体或部分生产单元使用。

6 设计和文档支持

6.1 设计文件

6.1.1 原理图

要下载 BOOSTXL-3PHGANINV 原理图，请参阅 BOOSTXL-3PHGANINV 中的设计文件。

要下载 TMDSHVMTRINSPIN 原理图，请参阅位于 C2000WARE-MOTORCONTROL-SDK 的 <install_location>\solutions\tmdshvmtrinspin\hardware> 文件夹中的硬件文件。

6.1.2 BOM

要下载 BOOSTXL-3PHGANINV 物料清单 (BOM)，请参阅 BOOSTXL-3PHGANINV 中的设计文件。

要下载 TMDSHVMTRINSPIN 物料清单 (BOM)，请参见 C2000WARE-MOTORCONTROL-SDK 的 <install_location>\solutions\tmdshvmtrinspin\hardware> 文件夹。

6.1.3 PCB 布局建议

6.1.3.1 布局图

要下载 BOOSTXL-3PHGANINV 布局图，请参阅 BOOSTXL-3PHGANINV 的设计文件。

要下载 TMDSHVMTRINSPIN 布局图，请参阅位于 C2000WARE-MOTORCONTROL-SDK 的 <install_location>\solutions\tmdshvmtrinspin\hardware> 文件夹中的硬件文件。

6.2 工具与软件

工具

TMDSCNCD263	TMDSCNCD263 是一款基于 HSEC180 controlCARD 的评估和开发工具，适用于 AM263x 系列 Sitara™ 高性能微控制器。该板提供易于使用的标准化平台来开发下一个应用，适用于初始评估和原型设计。
Code Composer Studio™	Code Composer Studio™ IDE 是一个完整的集成套件，使开发人员能够创建和调试所有德州仪器 (TI) 嵌入式处理器（Sitara、DSP、汽车、Keystone）、微控制器（SimpleLink™、C2000 数字控制、MSP430、TM4C、Hercules）以及数字电源 (UCD) 和可编程增益放大器 (PGA) 器件。
ARM-CGT-CLANG	tiarmclang 编译器工具提供包括编译器、汇编器和链接器等在内的软件开发工具，可用于使用 C/C++ 源代码开发应用程序，以便在 Arm Cortex-M 和 Cortex-R 系列内核处理器上加载并运行。
SYSCONFIG	SysConfig 是一款配置工具，旨在简化硬件和软件配置挑战，从而加速软件开发。SysConfig 提供直观的图形用户界面，用于配置引脚、外设、无线电、软件栈、RTOS、时钟树和其他元件。SysConfig 将自动检测、发现和解决冲突，来加快软件开发。

软件

MCU-PLUS-SDK-AM263X	AM263x 微控制器 (MCU) 和软件开发套件 (SDK) 共同构成一个面向嵌入式处理器的统一软件平台，此平台设置简单，可快速提供开箱即用的示例、基准测试和演示。
MOTOR-CONTROL-SDK-AM263X	适用于 AM263X 的电机控制 SDK 包含用于开发基于 RTOS 和非 RTOS 的应用程序的示例、库和工具，可实现电机位置感测的实时通信，以及用于 Arm R5F CPU 和相关外设的实时控制库。

6.3 文档支持

1. 德州仪器 (TI)：*电机控制 SDK 通用工程和实验* 用户指南。
2. 德州仪器 (TI)：*AM263x Sitara™ 微控制器* 数据表。
3. 德州仪器 (TI)：*AM263x Sitara™ 微控制器德州仪器 (TI) 产品系列* 技术参考手册。

4. 德州仪器 (TI)：*AM263x 控制卡硬件* 用户指南。

6.4 支持资源

TI E2E™ 中文支持论坛是工程师的重要参考资料，可直接从专家处获得快速、经过验证的解答和设计帮助。搜索现有解答或提出自己的问题，获得所需的快速设计帮助。

链接的内容由各个贡献者“按原样”提供。这些内容并不构成 TI 技术规范，并且不一定反映 TI 的观点；请参阅 TI 的使用条款。

6.5 商标

7 作者简介

Masoud Farhadi 是汽车应用特定 MCU 的系统工程师，为半导体行业的某些迫切需求提供设计帮助。Masoud 毕业于德克萨斯大学达拉斯分校，拥有电气工程电力电子学博士学位。他热衷于设计电力电子系统，并利用最新专利趋势来推动电动汽车技术的发展。

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、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

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

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。 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。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

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

TI 反对并拒绝您可能提出的任何其他或不同的条款。

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