User's Guide

用于 PMSM FOC 评估、带增量编码器的 C29x 伺服驱动器

摘要

用于 PMSM FOC 评估的带增量编码器的伺服驱动器 (简称为 *带 QEP 的伺服驱动器*) 是一个用于开发永磁同步电机 (PMSM) 场定向控制 (FOC) 并进行实验的框架。在此框架中，增量编码器 (QEP) 会检测 PMSM 的位置。该框架在一个基于 F29H85x 的低成本 LaunchPad 和一个三相逆变器评估模块 (EVM) 上运行。可以使用低压 PMSM 执行测试。

内容

插图清单

表格清单

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

带 QEP 的伺服驱动器的主要特性包括：

- 利用了搭载 C29x CPU 的 F29H85x 器件系列
- 演示：
  - 永磁同步电机 (PMSM) 的有传感器场定向控制 (FOC) 评估
  - 通过增量编码器 (QEP) 检测位置
- 支持多个三相逆变器评估套件
- 通过增量实施促进分步学习
- 支持进行低成本试验
- 为您自己的设计实现提供参考
- 提供了完整的源代码

备注

此 F29H85x 工程是 C2000 电机控制 SDK 中提供的 *C28x 通用伺服驱动器实验*的直接端口。*C28x 通用伺服驱动器实验*最初基于 *C2000 通用电机控制工程*，移除了无传感器算法。因此，*C2000 通用电机控制工程和实验指南*可作为实用参考，提供电机控制理论以及增量构建级别的更多详细信息。

1.1 硬件方框图

所需硬件为：

- 搭载基于 C29x 的 MCU 的低成本 LaunchPad
- 三相逆变器 BoosterPack
- 具有增量编码器的低压 PMSM 电机
- 外部直流电源

支持使用可选的数模转换器 (DAC) 评估模块进行调试。

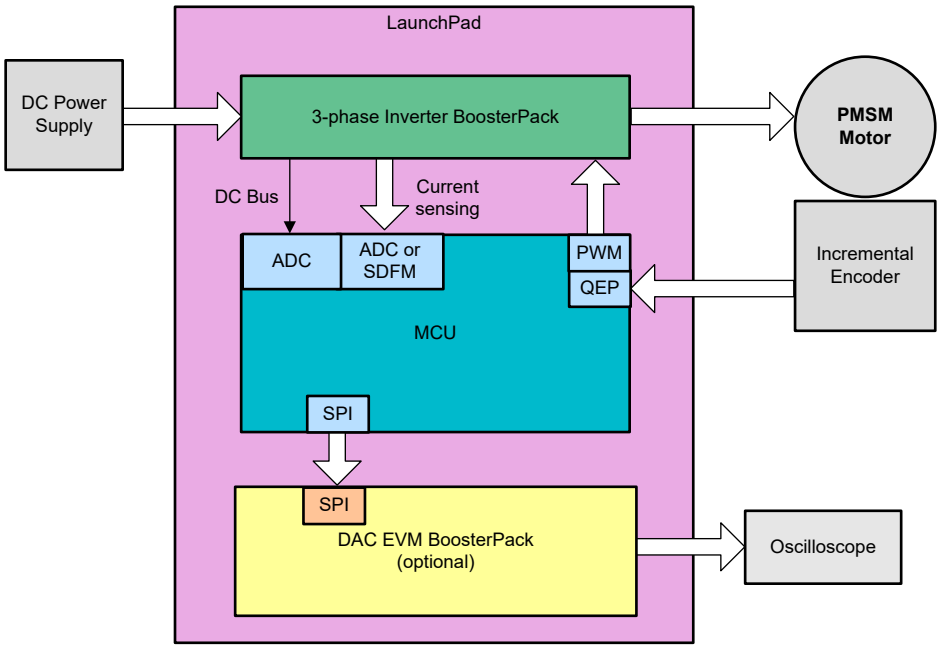

图 1-1. 硬件方框图

参阅：

- 节 2.1：支持的特定器件型号列表。
- 节 2.2：硬件设置。

1.2 软件流程图

带 QEP 的伺服驱动器软件包含：

- 启动例程：配置器件，计算模数转换器 (ADC) 偏移
- 电机控制中断服务例程 (ISR)：由 ADC 转换结束 (EOC) 信号或 SDFM 模块启动。
- 后台循环：定期更新电机控制参数。

运行应用程序所需的工具：

- Code Composer Studio™
- F29x 电机控制 SDK

参阅：节 2.3

- 所需工具版本
- 安装说明
- 应用软件配置
- 运行示例。

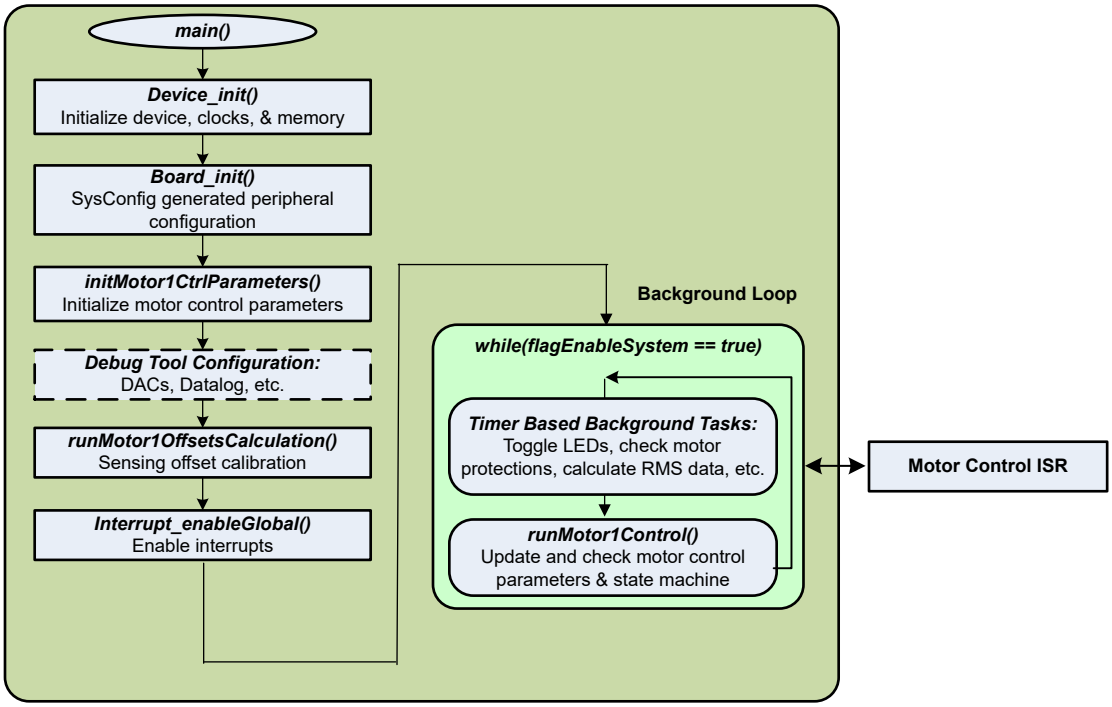

图 1-2. 流程图：启动和后台循环

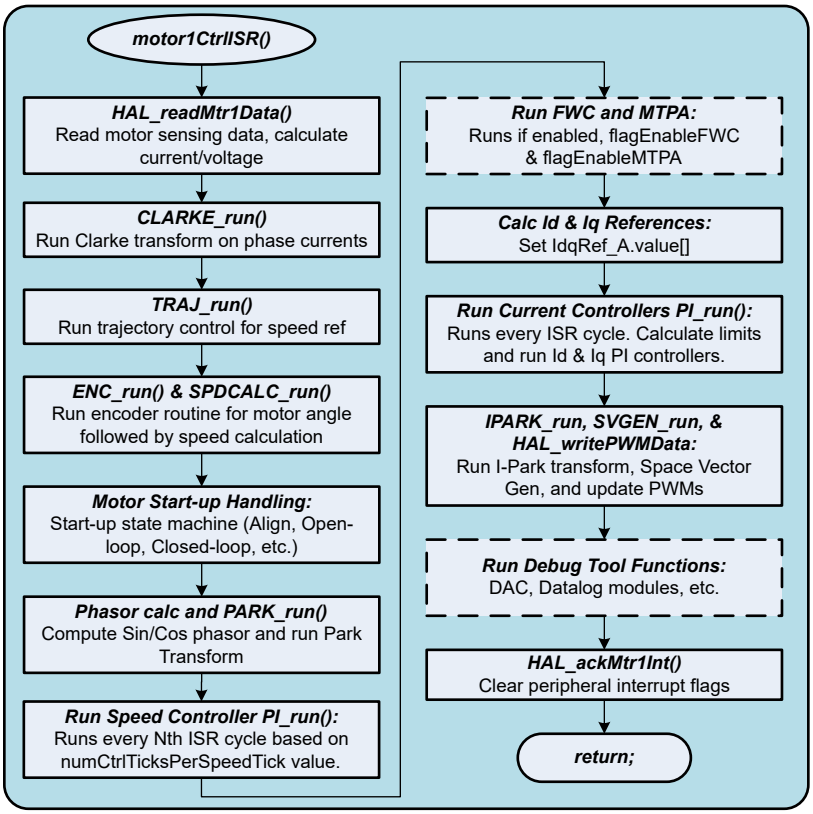

图 1-3. 流程图：电机控制中断服务例程

1.3 使用的 MCU 资源

表 1-1. 使用的资源

资源		
CPU	C29x CPU 1	
PWM 通道	6	
ADC 通道	BOOSTXL-3PHGANINV	4：3 个电流通道，1 个电压通道
	BOOSTXL-LMG2100-MD	4：3 个电流通道，1 个电压通道
	BP-AMC0106-LMG-MD	1：电压
SDFM 通道	BOOSTXL-3PHGANINV	0
	BOOSTXL-LMG2100-MD	0
	BP-AMC0106-LMG-MD	3：电流检测
eQEP 模块	1	
控制方法	场定向控制 (FOC)	
ISR 频率	20kHz	
ISR 性能	在 200MHz CPU 时钟频率且电机运行的情况下耗时 2.2 微秒	
内存消耗	RAM：12.4KB，闪存：60KB，采用编译器优化级别 O2	

2 在 TI 硬件上运行带 QEP 的伺服驱动器

本节提供在 TI 硬件上运行示例伺服器的指南。

2.1 支持的硬件

表 2-1 列出了支持的逆变器评估套件。每个逆变器电路板都已通过以下工具的测试：

- C2000 MCU 评估模块：LAUNCHXL-F29H85X
- 电机套件：LVSERVOMTR (嵌入式编码器)
- DAC EVM (可选) ：DAC128S085EVM：12 位 8 通道 DAC 评估模块

表 2-1. 支持的逆变器评估套件

逆变器电路板		
器件型号	说明	电流检测拓扑
BOOSTXL-3PHGANINV	12~60V、7A RMS、10A 峰值电流、三相 GaN 逆变器	三个基于采样电阻的内嵌式电机相电流检测
BOOSTXL-LMG2100-MD	不带散热器的 12~60V、27A RMS、三相 GaN 逆变器	三个基于采样电阻的内嵌式电机相电流检测
BP-AMC0106-LMG-MD	不带散热器的 12~60V、27A RMS、三相 GaN 逆变器	使用 Δ-Σ 调制器的三种基于采样电阻的内嵌式检测

2.2 硬件设置

本节介绍如何设置硬件以运行伺服驱动器：

- 配置 LaunchPad 开关
- 可选：将 DAC EVM 连接到 LaunchPad
- 将逆变器电路板连接到 LaunchPad
- 将电机连接到 LaunchPad 和逆变器电路板

2.2.1 LAUNCHXL-F29H85X 设置

LAUNCHXL-F29H85X 是一款低成本开发板，支持连接两个 BoosterPack 插件模块。有关用户指南和原理图等文档，请查看工具文件夹。

按如下方式配置 LaunchPad：

- BOOT：设置为闪存 (向上拨动两次)
- ADC 基准：设置为外部基准 (向上拨动)
- QEP：将 Q1 连接到 J12 (右侧开关向上拨动)
- 编码器连接：将 J12 连接到电机线束 J4。J12 接地引脚位于左侧。
- 将 USB 电缆连接到 C2000 LaunchPad 的板载 USB 连接器。该连接为 LaunchPad 供电并实现与 C2000 器件的隔离式 JTAG 连接。测试开始之前，请保持电缆断开连接。

备注

使用并行 I/O 引导可能导致 F29H85x 器件随机复位。除非有主机驱动控制线路，否则 BOOT 模式开关**不得**设置为并行 I/O 引导。

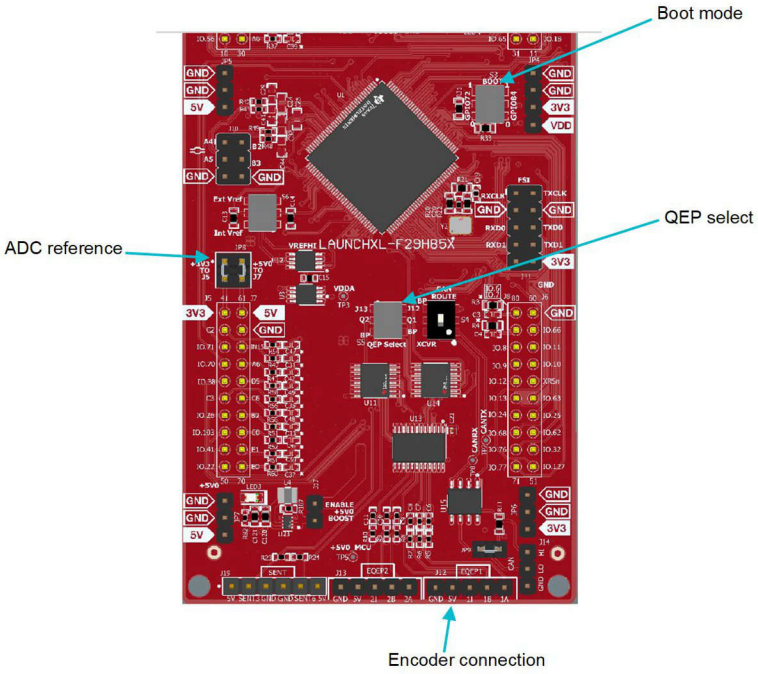

图 2-1. LAUNCHXL-F29H85X 配置

2.2.2 DAC128S085EVM 设置（可选）

DAC128S085EVM 评估模块是一款实用的调试工具。该电路板将通过 SPI 收到的数字数据转换为可在示波器上检查的模拟值。发送到 DAC 的典型变量包括角度、电压和电流测量值。

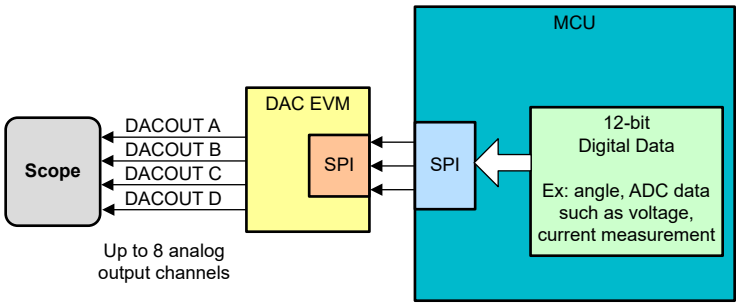

图 2-2. DAC128S 模块方框图

设置 DAC EVM：

1. 连接到 LaunchPad 连接器 J5/J7 和 J6/J8 (BoosterPack site2)。
2. 在 DAC EVM 上，使用跳线将引脚 JA-2 连接到引脚 JB-2。

如需更多信息：

- 要获取 EVM 文档和原理图，请访问 DAC128S085EVM 工具页面。
- 有关带 QEP 的伺服驱动器工程中的用法，请参阅 节 2.3.4.2。

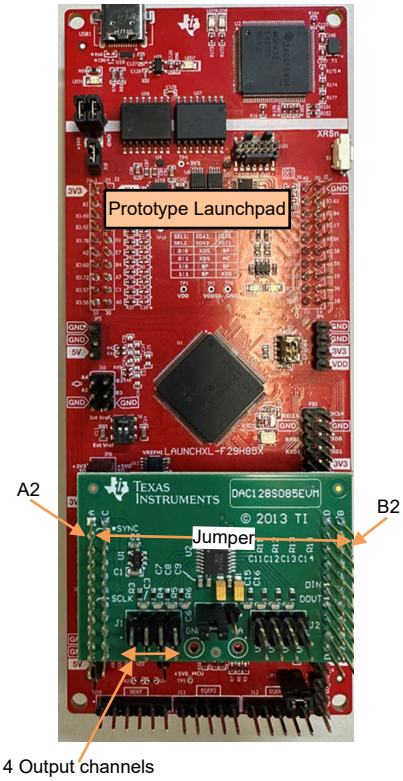

图 2-3. 连接到 F29H85X LaunchPad 的 DAC128S EVM

2.2.3 BOOSTXL-3PHGANINV 设置

备注

如果使用可选的 DAC128S085EVM，请加装垫高柱将 BOOSTXL-3PHGANINV 抬升至 DAC EVM 上方。

1. 按照 节 2.2.1 中所述设置 LaunchPad 跳线。
2. 将 BOOSTXL-3PHGANINV 连接到 LaunchPad 的 J1/J3 和 J4/J2。
3. 按照 节 2.2.6 中所述连接电机和编码器。对于硬件设置的初始测试（ 节 2.4.2 ），请保持电机与线束断开。
4. 将电压范围为 12V 至 60V 的直流电源连接到逆变器电路板的电源引脚。暂时保持电源关闭。

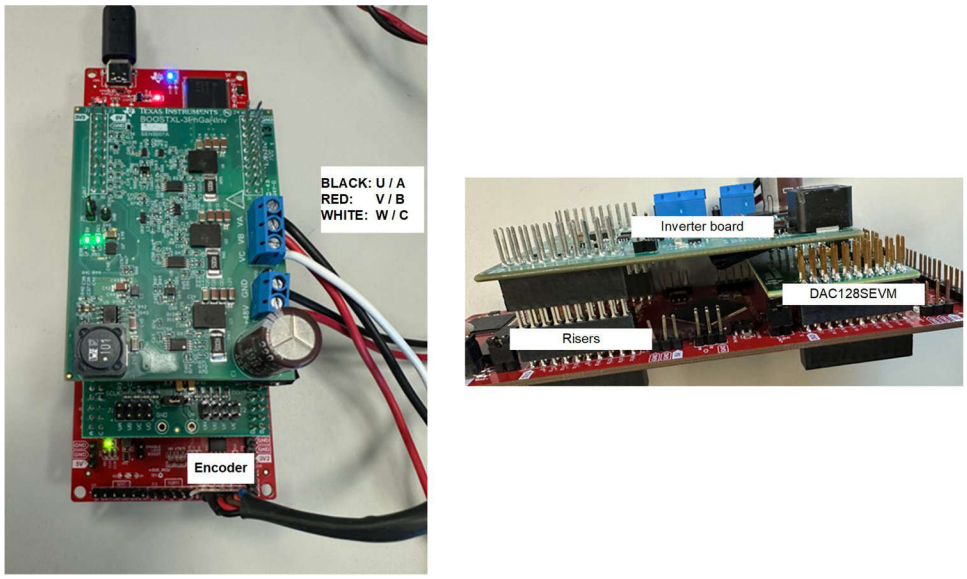

图 2-4. BOOSTXL-3PHGANINV 硬件设置

2.2.4 BOOSTXL-LMG2100-MD 设置

备注

对于该 BoosterPack，建议连接至 LaunchPad 的底部。

1. 按照 节 2.2.1 中所述设置 LaunchPad 跳线。
2. 将 BOOSTXL-LMG2100-MD 连接到 LaunchPad 的 J1/J3 和 J4/J2。使用底部连接器。
3. 按照 节 2.2.6 中所述连接电机和编码器。对于硬件设置的初始测试（ 节 2.4.2 ），请保持电机与线束断开。
4. 将电压范围为 12V 至 60V 的直流电源连接到逆变器电路板的电源引脚。暂时保持电源关闭。

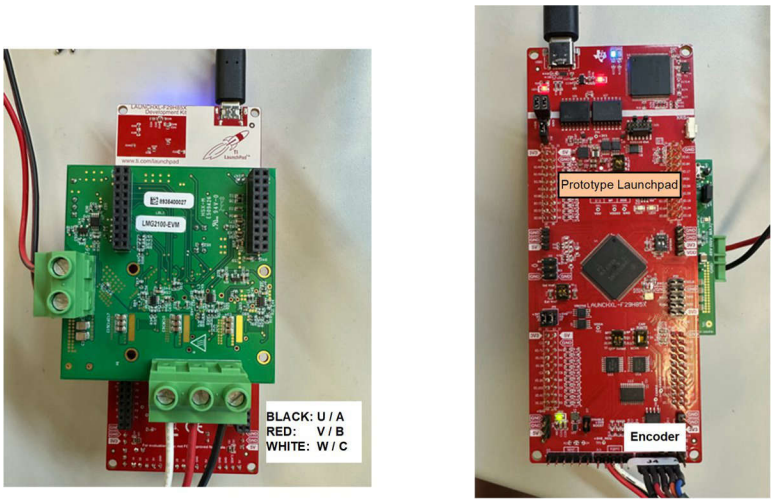

图 2-5. BOOSTXL-LMG2100-MD 硬件设置

2.2.5 BP-AMC0106-LMG-MD

1. 按照 节 2.2.1 中所述设置 LaunchPad 跳线。
2. 将 BP-AMC0106-LMG-MD 连接到 LaunchPad 的 J1/J3 和 J4/J2。
3. 按照 节 2.2.6 中所述连接电机和编码器。对于硬件设置的初始测试（ 节 2.4.2 ），请保持电机与线束断开。
4. 将电压范围为 12V 至 60V 的直流电源连接到逆变器电路板的电源引脚。暂时保持电源关闭。

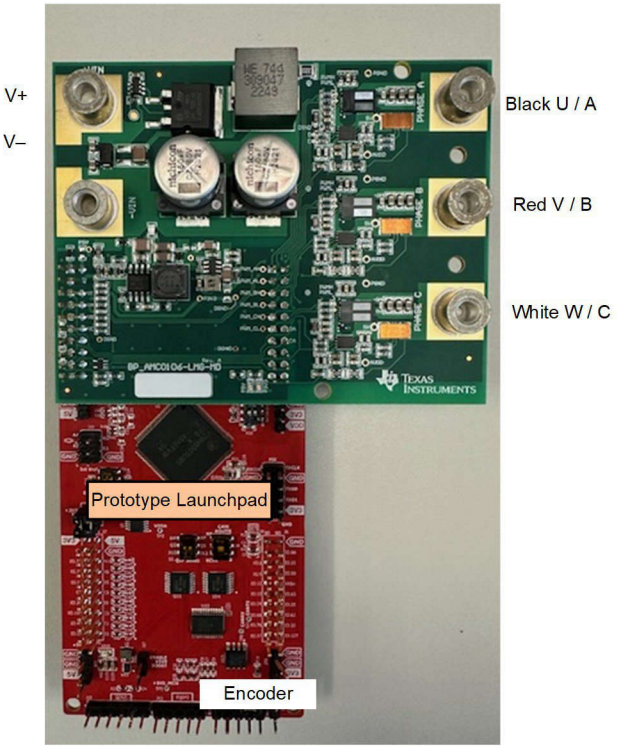

图 2-6. BP-AMC0106-LMG-MD 硬件设置

2.2.6 电机设置

表 2-2. 逆变器电路板和电机的连接

		LVSERVOMTR
电机相线	U/A	黑色 (16AWG)
	V/B	红色 (16AWG)
	W/C	白色 (16AWG)
编码器	**LAUNCHXL-F29H85X 的 J12**	**电机线束的 J4**
	GND (J12-1 左侧)	黑色 (J4-1)
	+5V (J12-2)	红色 (J4-2)
	1A (J12-3 QEP1_A)	棕色 (J4-3)
	1B (J12-4 QEP1_B)	橙色 (J4-4)
	1I (J12-5 QEP1_I 右侧)	蓝色 (J4-5)

2.3 实验软件

备注

根据提供的信息，这些工程使用编译器 V2.0.0.0.**STS**。虽然此版本只是**短期支持**版本，但相较于 V1.0.0.0.LTS，V2.0.0.0.STS 包含性能改进和错误修复。V2.0.0.0.STS 的目的仅是用于早期评估。请在长期支持 (<version>**.LTS**) 版本发布后迁移至该版本。

按照本节中的步骤在硬件上运行 Servo with QEP 应用程序。

- 设置软件开发环境。
- 配置应用软件
- 熟悉主要观察变量
- 利用增量功能分步构建和测试工程

2.3.1 软件开发环境

表 2-3. 开发设置步骤

步骤	条目	详细信息
1	Code Composer Studio	• V20.2 或更高版本 • 下载：Code Composer Studio (CCS)
2	了解 CCS	• Code Composer Studio Academy
3	F29 电机控制 SDK	• V1.0 或更高版本 • 下载：F29X-MOTOR-CONTROL-SDK
4	安装 Python 和 OpenSSL	• F29x SDK 中的入门指南包括“构建、加载和运行”一节中重点介绍的先决条件。 • <install>\c29_sdk\mcu_sdk_f29h85x\docs\html\GETTING_STARTED.html
5	打开 CCS 并导入工程。	• *File → Import Project(s)* • <install>\solutions\servo_drive_qep • 选择与您的逆变器硬件匹配的工程。 – 示例：*servo_drive_qep_**3phGanInv*** 对应 BOOSTXL-**3phGanInv**

备注

继续之前请确保安装了正确的 Python 和 OpenSSL 版本（第 4 步）。

2.3.2 工程组织结构

导入 Code Composer Studio 后，带 QEP 的伺服驱动器工程将具有 表 2-4 中所述的文件夹结构。

表 2-4. 文件夹结构

文件夹	包括
README.html	工程概述。链接到 SDK 文档。
sys_main.c	main() 函数和后台循环
sys_main.h	#包含其他头文件、SYSTEM_Vars_t 的定义
sys_settings.h	#define DMC_BUILDLEVEL DMC_LEVEL_x.此定义确定使用的控制功能：开环、电流闭环、电流 + 速度闭环。
libraries/	典型的 FOC 模块，包括 Park、Clark 以及逆向 Park 和 Clark 变换。支持增量编码器、数据日志和 DAC EVM。
src_board/	用于运行伺服驱动器的特定于器件的驱动器。若要将工程迁移到另一个电路板，主要可以更改：*<board>.syscfg*、*hal.c*、*hal.h* 和 *user_mtr1.h*。
src_control/	这是电机驱动控制文件，会调用中断服务例程和后台任务中的电机控制核心算法函数。motor1_drive.c 包含 *motor1CtrlISR()*。
src_device/	特定于器件的启动文件、驱动程序库、链接器命令文件。

2.3.3 软件配置

1. **预定义符号：**

   提供了编译器预定义宏来配置软件。表 2-5 列出了相关选项。若要修改、添加或删除宏：

   右键单击工程 → Properties → Build → C2000 Compiler → Predefined Symbols

   备注

   后缀为“_N”的预定义符号已被禁用。例如，可通过删除“MOTOR1_FWC_N”上的“_N”，将预定义符号更改为“MOTOR1_FWC”来启用弱磁控制 (FWC)。

2. **电机模型定义：**

   在 *src_board/user_mtr1.h* 和 *src_board/user_common.h* 文件中定义。

   找到 USER_MOTOR1 的定义。确认该定义与所测试的电机相匹配。此版本仅测试了 Teknic_M2310PLN04K。

```
#define USER_MOTOR1 Teknic_M2310PLN04K
```

表 2-5. 预定义符号定义

预定义符号	说明	必需还是可选	默认值
MOTOR1_ENC	增量编码器	必需	启用
ADC_EXT_REF	ADC 使用外部基准。如果禁用了此预定义符号，则必须修改 SysCfg 文件以使用内部基准。	推荐	启用
MOTOR1_FWC	弱磁控制。通常与 MTPA 一起启用。	可选	禁用
MOTOR1_MTPA	每安培最大扭矩。通常与 FWC 一起启用。	可选	禁用
DATALOG_EN	通过 datalog 缓冲器导出数据	可选	禁用
DAC128S_ENABLE	通过 DAC EVM 导出数据	可选	禁用
DAC_ON_CHIP_ENABLE	通过片上 DAC 导出数据	可选	禁用

2.3.4 调试接口

2.3.4.1 数据日志

数据日志软件模块支持登录到软件缓冲区。然后可使用 Code Composer Studio 的图表功能显示这些缓冲区。

SDK 文档中记录了数据日志库。位于 Libraries 下的 *<install>/docs/html_guide/index.html*。

1. 将符号 DATALOG_EN 添加到工程的预定义符号中。
2. 在 *libraries/utilities/datalog_input.h* 中选择缓冲区数量、缓冲区大小和数据类型。
3. 调用 *src_board/diagnostics.c* 中的 setupDatalog() 函数：

```
// Initialize Datalog
datalogHandle = DATALOG_init(&datalog, sizeof(datalog), manual, 0, 1);
DATALOG_Obj *datalogObj = (DATALOG_Obj *)datalogHandle;
datalogObj->flag_enableLogData = false;
datalogObj->flag_enableLogOneShot = false;
....
datalogObj->iptr[0] = (float32_t*) &motorVars_M1.senseData.I_A.value[0];
datalogObj->iptr[1] = (float32_t*) &motorVars_M1.senseData.I_A.value[1];
datalogObj->iptr[2] = (float32_t*) &motorVars_M1.angleFOC_rad;
```

   可以修改此函数来输出不同的信号。对于此工程，您可以监测角度和检测到的电流值，如此处所示。

4. 在适用时启动数据日志。可以单次记录，也可以连续记录。

```
#if defined(DATALOG_EN)
    // datalog.flag_enableLogOneShot = true;
   datalog.flag_enableLogData = true;
#endif  // DATALOG_ENABLE
```

5. 将数据发送到数据日志：

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

2.3.4.2 数模转换器

DAC128S 软件模块支持 节 2.2.2 中所述的 DAC EVM。该软件模块能够将最多八个软件变量转换为 12 位整数值，并通过 SPI 将该数据传输到 DAC EVM。

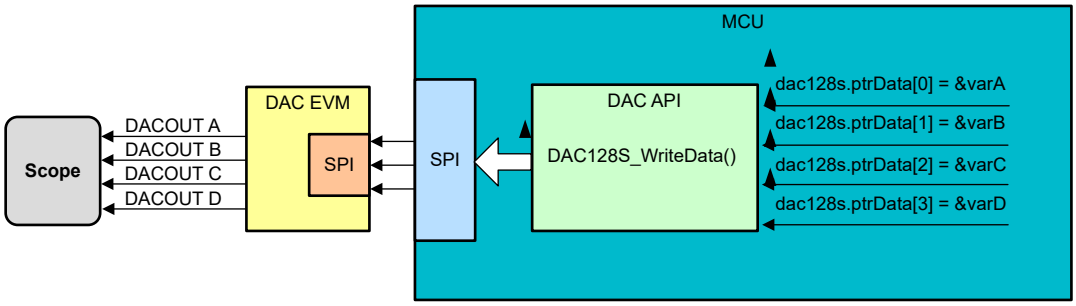

图 2-7. DAC128S 方框图

要使用 EVM 功能，请按如下步骤操作：

1. 按 节 2.2.2 中所述连接硬件。
2. 将符号 DAC128S_ENABLE 添加到工程的预定义符号中。
3. 在 *libraries/dacs/dac128s085.h* 中选择通道数。
4. 调用 *src_board/diagnostics.c* 中的 setupDAC1285S() 函数：

```
dac128s.ptrData[0] = &motorVars_M1.angleENC_rad;                // CH_A
dac128s.ptrData[1] = &motorVars_M1.senseData.I_A.value[0];      // CH_B
dac128s.ptrData[2] = &motorVars_M1.senseData.I_A.value[1];      // CH_C
dac128s.ptrData[3] = &motorVars_M1.senseData.I_A.value[2];      // CH_D
```

   a. 可以修改此函数来输出不同的信号。对于此工程，您可以监测角度和检测到的电流值，如此处所示。

5. 发送数据。

```
// Write the variables data value to DAC128S085 through the SPI
DAC128S_writeData(dac128sHandle);
```

   a. 在执行 *src_control/motor1_drive.c* 中的控制中断 (motor1ctrlISR()) 期间，代码会定期发送数据。
   b. 发送的 DAC 输出路数取决于 *dac128s085.h* 中的配置。

该 EVM 具有一个八通道 12 位数模转换器 (DAC)。根据提供的信息，输出路数设置为 4，因为大多数示波器只有四个探头。更多的输出会消耗转换和传输数据的时间，这可能会影响其他任务可用的时间。不过，用户可以通过更改 *libraries/dacs/dac128s085.h* 文件中的定义，将输出路数设置为 1 到 8 之间。

2.4 通过增量步骤测试工程

伺服驱动系统可分阶段增量式地测试。通过在 *sys_settings.h* 中定义 DMC_BUILDLEVEL 并重新构建工程来启用每个阶段。在所示的代码示例中，“第 4 步：速度闭环和电流闭环”由 DMC_BUILDLEVEL 启用。

```
//=============================================================================
// Incremental Build options for System check-out
//=============================================================================
#define DMC_LEVEL_1     1   //Y 50% duty, offset calibration and verify phase shift
#define DMC_LEVEL_2     2   //Y open loop control to check sensing signals
#define DMC_LEVEL_3     3   //Y closed current loop to check the hardware settings
#define DMC_LEVEL_4     4   //Y run with sensored FOC
```

```
#define DMC_BUILDLEVEL  DMC_LEVEL_4
```

1. **第 1 步：硬件设置校验。**

   #define DMC_BUILDLEVEL DMC_LEVEL_1

   验证 ADC 偏移校准、ePWM 50% 占空比输出、死区和相移。
2. **第 2 步：开环控制。**

   #define DMC_BUILDLEVEL DMC_LEVEL_2

   在开环控制下运行电机，验证电机电流和电压检测信号。
3. **第 3 步：电流闭环。**

   #define DMC_BUILDLEVEL DMC_LEVEL_3

   在电流闭环中运行电机，验证电流检测和电流控制。
4. **第 4 步：速度闭环和电流闭环。**

   #define DMC_BUILDLEVEL DMC_LEVEL_4

   最后一步是在速度环路和电流环路均闭合的情况下运行电机。

2.4.1 观察变量

结构 motorVars_M1 引用了大多数与控制伺服驱动器相关的变量。表 2-6 给出了填充 CCS 监视窗口的建议列表。

备注

Code Composer Studio V20.2 无法导出/导入监视窗口变量。此功能预计将在 V20.3 中实现。

表 2-6. 观察变量概述

变量	说明
motorVars_M1.ISRCount	每次电机 ISR 执行时递增
systemVars.flagEnableSystem	自动从 0 跳变为 1
motorVars_M1.flagEnableRunAndIdentify	• 设置为 1 将在自动设置 flagEnableSystem 变量后启动电机。 • 设置为 0 将禁用 PWM。然后可以暂停 MCU。 注意：对于构建版本 1，这会启动 PWM 进行检查。
motorVars_M1.flagRunIdentAndOnLine	如果没有故障，则变为 1
motorVars_M1.motorState	显示当前的电机控制状态，例如： • MOTOR_STOP_IDLE • MOTOR_FAULT_STOP • MOTOR_ALIGNMENT • MOTOR_OL_START • MOTOR_CL_RUNNING • MOTOR_CTRL_RUN
motorVars_M1.estimatorMode	对于增量编码器，此值为 ESTIMATOR_MODE_ENC
motorVars_M1.faultMtrUse.all	如果存在过流故障，则值为非零
motorVars_M1.faultMtrUse.bit	展开并检查故障标志。在可能导致电机关闭的故障中，这些故障当前处于活动状态。（faultMtrMask 应用于 faultMtrNow）
motorVars_M1.faultMtrNow.bit	在所有可能的故障中，这些故障当前处于活动状态。
motorVars_M1.faultMtrMask.bit	在所有可能的故障中，这些故障会导致电机关闭。
motorVars_M1.senseData.VdcBus_V	直流总线电压近似值

表 2-6. 观察变量概述（续）

变量	说明
motorVars_M1.senseData.offset_I	• ADC 用来检测电流的电流偏移值 • 对于 ADC 检测：2048（12 位 ADC 量程值的一半） • 对于 SDFM 检测：这些数字非常小
motorVars_M1.speedRef_Hz	• 电机的基准速度。 • 更改此变量可以提高或降低速度。 • 负值会反转方向 • 未在构建级别 1 中使用
motorVars_M1.speed_Hz	电机当前的速度。未在构建级别 1 中使用。
motorVars_M1.overCurrent_A	通过减小该值，可验证 CMPSS 模块的故障保护功能。

2.4.2 第 1 步 硬件设置验证

目标：

- 使用 SysConfig 和 HAL 对象针对伺服驱动器硬件来初始化 MCU 外设。
- 验证 PWM 和 ADC 驱动器模块和硬件连接
- 验证来自逆变器的检测反馈值：相电流偏移和直流总线电压

在这一步，PWM 以固定的 50% 占空比运行，并且电机未连接。

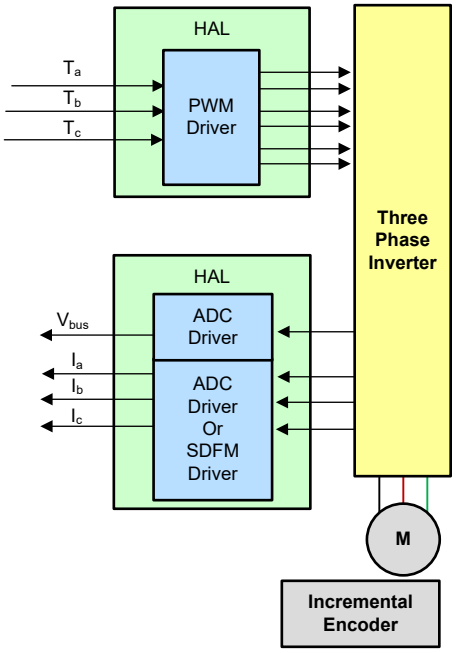

图 2-8. 第 1 步：硬件设置校验方框图

2.4.2.1 构建、加载和运行工程

按照 表 2-7 和 表 2-8 中的步骤验证硬件设置。 表 2-7 中的步骤也适用于开环、电流闭环和电流+速度闭环构建级别。

表 2-7. 设置步骤

步骤	说明	注释
1	硬件设置	请参阅节 2.2
2	通过 USB 将 PC 连接到 LaunchPad	为 LaunchPad 供电并建立 JTAG 连接
3	为逆变器电路板供电	外部 DC 电源
4	导入工程	在 CCS 中：*File → Import Project(s)*
5	设置 DMC_BUILDLEVEL	• *sys_settings.h* • #define DMC_BUILDLEVEL DMC_LEVEL_x 其中 x = 1、2、3、4 • 运行 DMC_LEVEL_1 时，请勿连接电机
5	• 连接到 LaunchPad • 构建工程 • 对闪存编程	在 CCS 中：*Run → Debug Project*（或按 F5）。

表 2-8. 硬件验证程序

步骤	说明	
1	开始在 CCS 中执行：*Run → Continue*（或按 F5）	
2	**确认：**变量递增	*motorVars_M1.ISRCount*
3	**确认：**变量更改为 1	*systemVars.flagEnableSystem*
4	**设置：**设置为 1 以启动 ePWM	*motorVars_M1.flagEnableRunAndIdentify*
5	**确认：**变量更改为 1	*motorVars_M1.flagRunIdentAndOnLine*
6	**确认：**大约为 ADC 量程的一半	*motorVars_M1.senseData.offset_I.value[]*
7	**确认：**匹配硬件直流总线电压	*motorVars_M1.senseData.VdcBus_V*
8	**确认：**使用示波器测量 PWM 占空比和开关频率	
9	**Clear：**（设为 0）以禁用 ePWM	*motorVars_M1.flagEnableRunAndIdentify*
10	现在可以停止运行 CPU 并断开 CCS。	
11	关闭逆变器电路板的电源	
12	将 LaunchPad 与 PC 断开	

如果任何步骤导致意外结果，请检查以下各项：

1. PWM
   a. PWM 频率在 .syscfg 文件中配置
   b. *usr_mtr1.h* 中的 #define USER_M1_PWM_FREQ_kHz 需要与.syscfg 配置匹配。
2. 电机驱动器板已正确设置并通电
3. 导入的工程与电机驱动器板匹配。电路板可通过名称识别。
4. 检查 LaunchPad 上的开关。

2.4.3 第 2 步 开环控制

目标：

- 在开环控制下运行系统
- 校验电流和电压检测电路
- 校验增量编码器是否正确配置和连接

备注

在开环控制下：

- ADC（或 SDFM）检测到的电流值仅用于验证和校验。这些值不会实际用于控制电机。
- 增量编码器数据用于估算校验速度。该速度不会用于控制电机。

备注

必须提供编码器每旋转一周的时隙数。需要此值才能正确地将编码器信号转换为角度。检查 *user_mtr1.h* 中的 **USER_MOTOR1_NUM_ENC_SLOTS** 定义。

值不正确会导致电机旋转得更快或更慢。请注意，该值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。

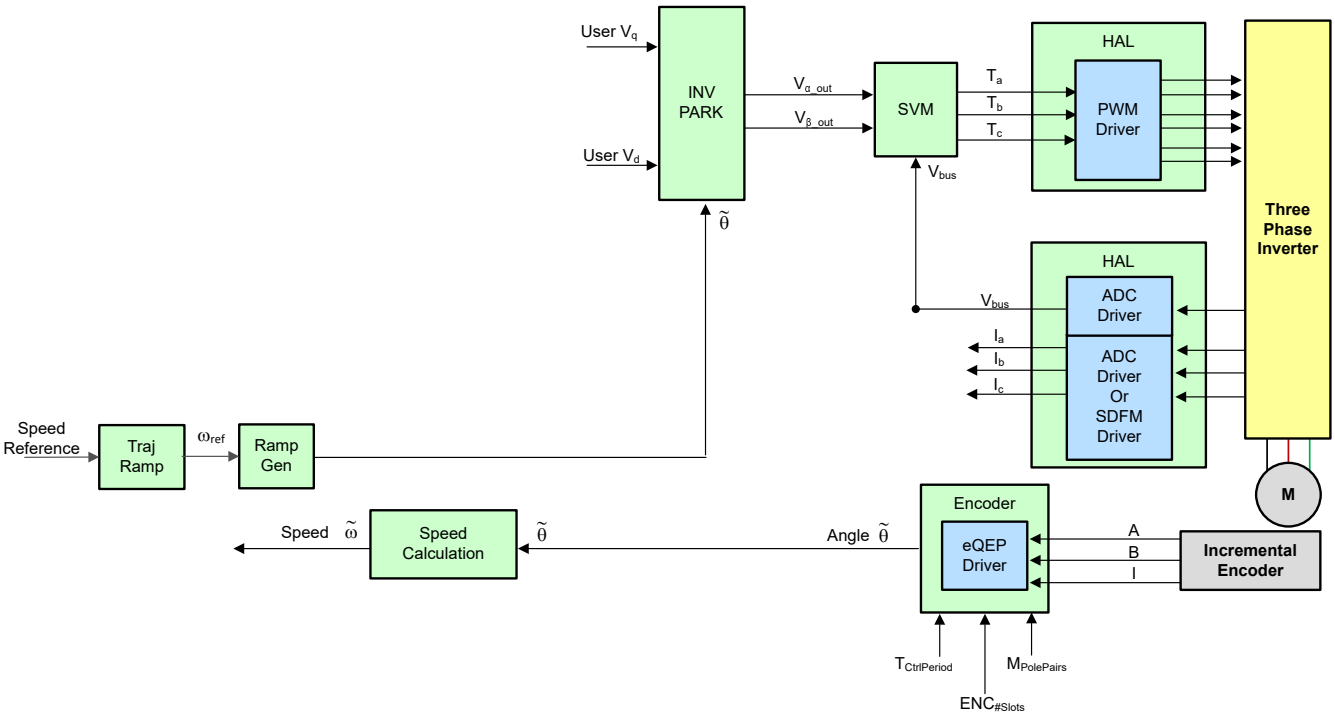

图 2-9. 第 2 步：开环控制方框图

2.4.3.1 构建、加载和运行工程

采用 表 2-7 中的设置：

- 将电机连接到线束。
- #define DMC_BUILDLEVEL DMC_LEVEL_2.

控制测试程序 提供了控制测试程序。该测试程序被重复用于电流闭环测试，以及速度 + 电流闭环测试。

表 2-9. 控制测试程序

步骤	说明	
1	开始在 CCS 中执行：*Run → Continue*（或按 F5）	
2	**确认：**变量递增	*motorVars_M1.ISRCount*
3	**确认：**变量更改为 1	*systemVars.flagEnableSystem*
4	**设置：**设置为 1 以启动 ePWM	*motorVars_M1.flagEnableRunAndIdentify*
5	**确认：** • 变量更改为 1（无故障） • 电机开始旋转	*motorVars_M1.flagRunIdentAndOnLine*
6	**确认：**电机速度上升至基准速度	• *motorVars_M1.speed_Hz* • *motorVars_M1.speedRef_Hz*
7	**修改：** • 基准速度并观察当前速度变化 • 负基准速度会改变方向	
9	**监测：**相电流和角度 • 使用数据日志缓冲区、片上 DAC 或 DAC EVM。 • 将电流与使用电流探头测得的实际相电流进行比较	
10	**Clear：**（设为 0）以禁用 ePWM	*motorVars_M1.flagEnableRunAndIdentify*
10	现在可以停止运行 CPU 并断开 CCS。	
11	关闭逆变器电路板的电源	
12	将 LaunchPad 与 PC 断开	

2.4.4 第 3 步 电流闭环

此构建级别的目标：

- 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，使用 i/f 控制来控制电机。转子角度由斜坡发生器模块生成。

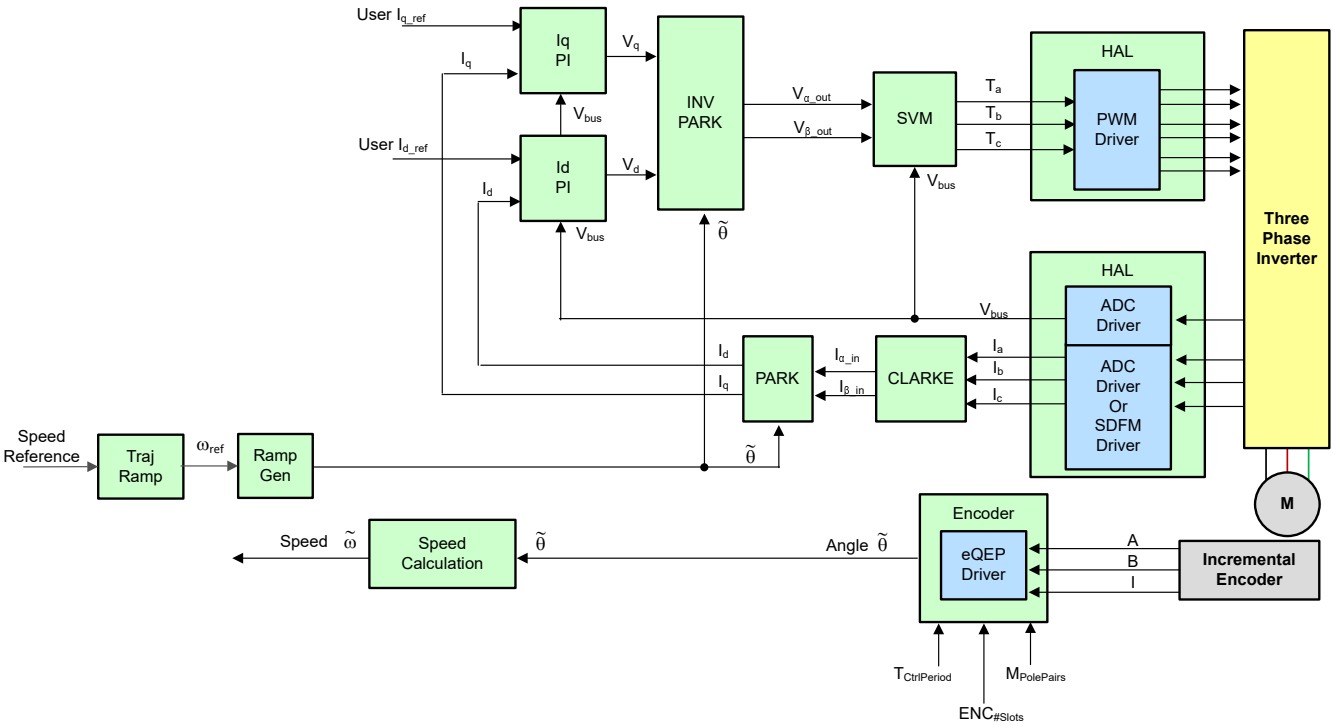

图 2-10. 第 3 步：电流闭环控制方框图

2.4.4.1 构建、加载和运行工程

要验证采用电流闭环控制的电机：

- 采用 表 2-7 中的设置。这一次：
  - 将电机连接到线束
  - 在 *sys_settings.h* 中指定 #define DMC_BUILDLEVEL DMC_LEVEL_3
- 执行表 2-9 中的测试程序

2.4.5 第 4 步 速度闭环和电流闭环

了解该构建级别中的目标：

- 使用基于增量编码器的场定向控制 (FOC) 评估整个电机驱动器。

在该构建级别，外侧速度环路和内侧电流环路是闭合的。转子角度通过连接到 QEP 的增量编码器进行测量。

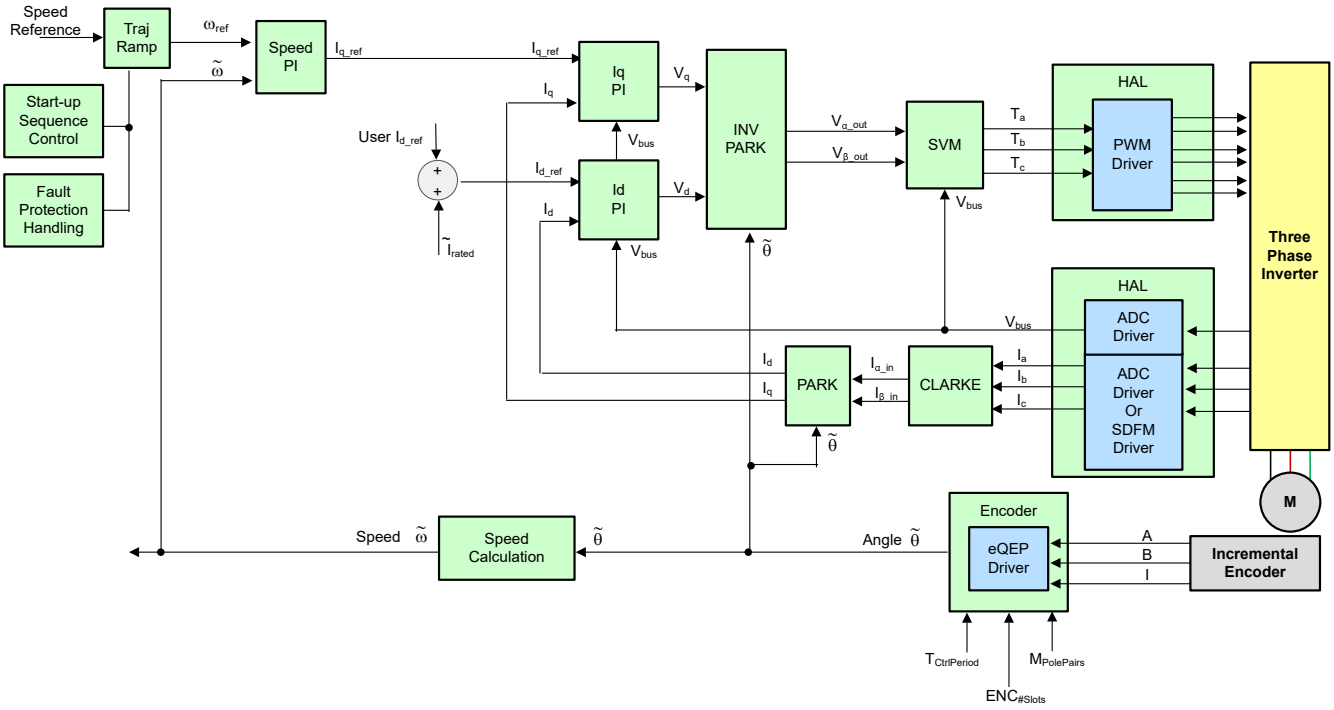

图 2-11. 第 4 步：速度闭环和电流闭环控制方框图

2.4.5.1 构建和加载工程

要验证采用电流闭环控制的电机：

- 采用相同的设置（表 2-7）。这一次：
  - 将电机连接到线束
  - 在 *sys_settings.h* 中指定 #define DMC_BUILDLEVEL DMC_LEVEL_4
- 执行 表 2-9 中的相同测试程序

参考资料

1. C29 Academy
2. Code Composer Studio Academy
3. 适用于 F29 实时 MCU 的基础软件开发套件
4. TI C29x Clang 编译器工具用户指南
5. 德州仪器 (TI)：*C29 CPU - 利用 C2000 MCU 上经过优化的架构提供无可匹敌的实时性能*

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