参考文献

Design Guide: TIDA-010948

采用位置反馈和工业通信协议的 6 轴电机控制参考设计

说明

本参考设计展示了使用 2 个 TI Sitara™ MCU-AM243x 器件处理 6 轴控制型电机驱动器（采用简单开放实时千兆以太网 (SORTE_G) 连接）的能力。在该设计中，其中一个 AM243x 器件包含 1 个 800MHz R5F 内核，可针对具有增量编码器的 6 个独立电机以 62.5 µs 周期时间执行闭环磁场定向控制。此 AM243x 的一个可编程实时单元 (PRU) 内核可充当 SORTE_G 控制器，用于发送请求和接收 6 轴电机角度。另一个 AM243x 器件的 PRU 内核则可充当 SORTE_G 器件，用于发送 6 轴电机角度，这些角度将由此 AM243x 上的其他 PRU 内核进行解码，并在每个周期内发送至控制器端。AM243x 还支持多协议绝对编码器和多协议工业以太网。

资源

TIDA-010948、TIDEP-01032	设计文件夹
AM243x 电机控制 SDK、LP-AM243	工具文件夹
BLDC BoosterPack™ 插件模块	工具文件夹
AM2434、TQ-3P-SOM-TQMA243XL	产品文件夹
DRV8316、AMC1035	产品文件夹
DP83869、THVD1450	产品文件夹

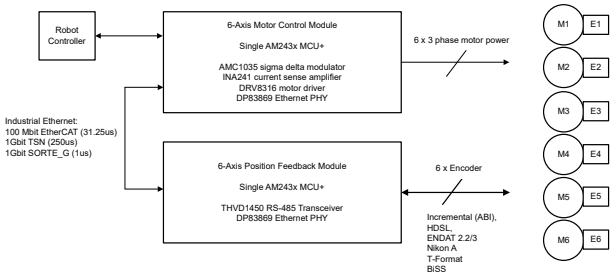
请咨询 TI E2E™ 支持专家

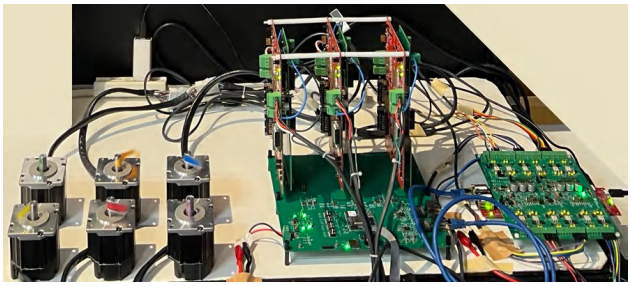
特性

- 18 通道 EPWM 和 18 通道 ICSSG_PRU PWM，可根据六个 FOC 环路的输出生成 PWM 波形
- 12 通道 Σ-Δ 滤波固件支持连续采样，并可针对六个直接连接的电机的相电流反馈，在 PRU 内核之间实现负载共享
- 由 ICSSG 实施的 SORTE_G 固件。包括适用于 EtherCAT®、PROFINET® 或千兆位 TSN 的扩展选项
- 六个电流和速度独立闭环的 FOC，周期时间为 62.5 µs
- 借助 EQEP 模块和 PRU 内核进行解码的六通道增量编码器
- 通过 ICSSG 扩展了多编码器协议选项，例如 EnDAT、HDSL、Biss C、Tamagawa 和 Nikon A

应用

- 机器人伺服驱动器
- 伺服驱动器位置反馈
- 机器人位置反馈聚合器
- 伺服驱动器控制模块





1 系统说明

该参考设计展示了 AM243x 器件处理 6 轴综合实时伺服电机控制和工业通信协议的能力。该实现在两个 TI AM243x SoC、一个 AM243x ALV 封装和一个 AM243x ALX 封装上完成。

AM243x ALV 封装连同 TQ-PWM 一起放置在控制板上，以使用 EPWM 和 ICSSG_PRU SOM 外设生成 36 个互补 PWM 信号，并测量来自 12 通道 Δ-Σ 滤波器模块的相电流。此操作使用 ICSSG 固件实现，还通过 SORTE_G 与位置反馈板进行通信，以便在每个 62.5µs 周期时间内发送和接收 6 轴电机机械角请求。在控制板上，该设计使用一个 800MHz R5F 内核为六个具有增量编码器的独立电机实现 FOC 环路，也可以通过多种协议扩展到绝对编码器。另一个 R5F 内核可用于实现工业以太网堆栈作为选件。一个 ICSSG1 用作 Δ-Σ 滤波器模块，另一个 ICSSG0 用作 SORTE_G 控制器或 EtherCAT 辅助控制器，用于额外的选项。

采用 ALX 封装的 LP_AM243 用于位置反馈板控制器。位置反馈板上的 ICSSG0 通过工业以太网外设 (IEP) 捕获 ABI 信号的上升沿和下降沿，从而对 4 通道增量编码器进行解码。其他两个通道编码器使用 EQEP 模块进行解码。ICSSG1 用作 SORTE_G 器件，以在每个 62.5µs 周期时间内发送 6 轴电机机械角数据。

1.1 术语

SoC	片上系统
FOC	场定向控制
MCU	微编程控制单元
ALV、ALX	AM243x 器件的封装图
ABI	增量编码器，具有两个正交编码输出 A 和 B 以及索引 I
RPM	每分钟转数
LUT	查找表
EnDAT、HDSL、BissC、Tamagawa、Nikon A	适用于绝对编码器的多数字双向接口协议
EtherCAT	用于控制自动化技术的以太网
Profinet	Process Field Network（过程现场网络）的混成词
SORTE	简单的开放实时以太网
ICSSG	千兆位工业通信子系统
PRU	可编程实时单元
RTU	辅助可编程实时单元
SDFM	Σ-Δ 滤波模块
SDDF	Σ-Δ 抽取滤波
SDM	Σ-Δ 调制器
IEP	工业以太网外设
CMP	事件比较器
CAP	事件捕获
ISR	中断服务例程
EPWM	增强型脉宽调制器
EQEP	增强型正交编码器脉冲
GPIO	通用输入输出
FIFO	先入先出
TSR	时间同步路由器（AM243x 中通用中断路由器模块的实例）
TCM	紧耦合存储器
DRAM	动态随机存取存储器

RGMII	简化千兆位媒体独立接口
MII_G_RT	千兆位实时媒体独立接口
MDIO	管理数据输入输出
TQ-SOM	基于 Arm® 的 TI 处理器的模块上系统供应商，提供内部制造和设计服务

1.2 主要系统规格

表 1-1. 主要系统规格

子系统	规格	注释
SoC EPWM	18 通道互补 PWM，具有： • 可配置死区 • 单或双更新功能 • 可调开关频率 • 同步或相移模式功能 • 与 PRU-ICSS PWM 和 SDFM 同步	当前演示仅针对同步模式。
ICSSG_PRU PWM	18 通道互补 PWM，具有： • 可配置死区 • 单或双更新功能 • 可调开关频率 • 同步或相移模式功能 • 与 EPWM 和 SDFM 同步	当前演示仅针对同步模式。
电流反馈 – ICSS_SDFM	用于相电流反馈（每个轴 2 个通道）的 12 通道 SDFM，具有： • 正常电流 OSR：64 • 在 RTU 和 PRU 内核之间分担负载 • 连续采样模式 • 单或双更新功能	
位置反馈 – PRU EQEP	通过 PRU 内核和 IEP_CAP 从增量 ABI 编码器解码 4 通道电机角度	
位置反馈 – SoC EQEP	通过 SoC EQEP 模块从增量 ABI 编码器解码 2 通道电机角度	
控制算法 – FOC 环路	在 62.5 µs 周期时间（16kHz PWM 频率）内针对电流和速度实现 6 个独立的 FOC 环路	当前演示针对 16kHz 双更新
控制算法 – 时间同步	控制环路（位置、速度、电流、PWM 和工业以太网）之间的时序同步	在当前演示中，使用 SORTE_G 和 TSR 实现控制器和器件之间的同步。
工业通信 – SORTE_G	控制器和器件之间的点对点通信。 控制器发送 6 轴电机机械角度请求，并在每个 62.5µs 周期时间中通过千兆位工业以太网协议接收数据	与 EtherCAT、Profinet 相比，使用 SORTE_G 来降低延迟

2 系统概述

图 2-1 展示了 TIDA-010948 整体系统的设置。该系统包含以下特性：

- TIDA-010948 参考设计，这是一款 6 轴控制板，采用 AM2434 ALV SoC 作为主伺服控制 MCU
- 三个 BP-AM2BLDCSERVO 板作为伺服控制的功率级，每个 BP-AM2BLDCSERVO 板可以支持双轴
- 三个适配器板，用于将信号从功率级路由到控制板
- 六轴位置板，用于接收六通道编码器信号
- 一个 LP-AM243x，采用 AM2434 ALX SoC 作为位置反馈 MCU 来解码电机角度和速度

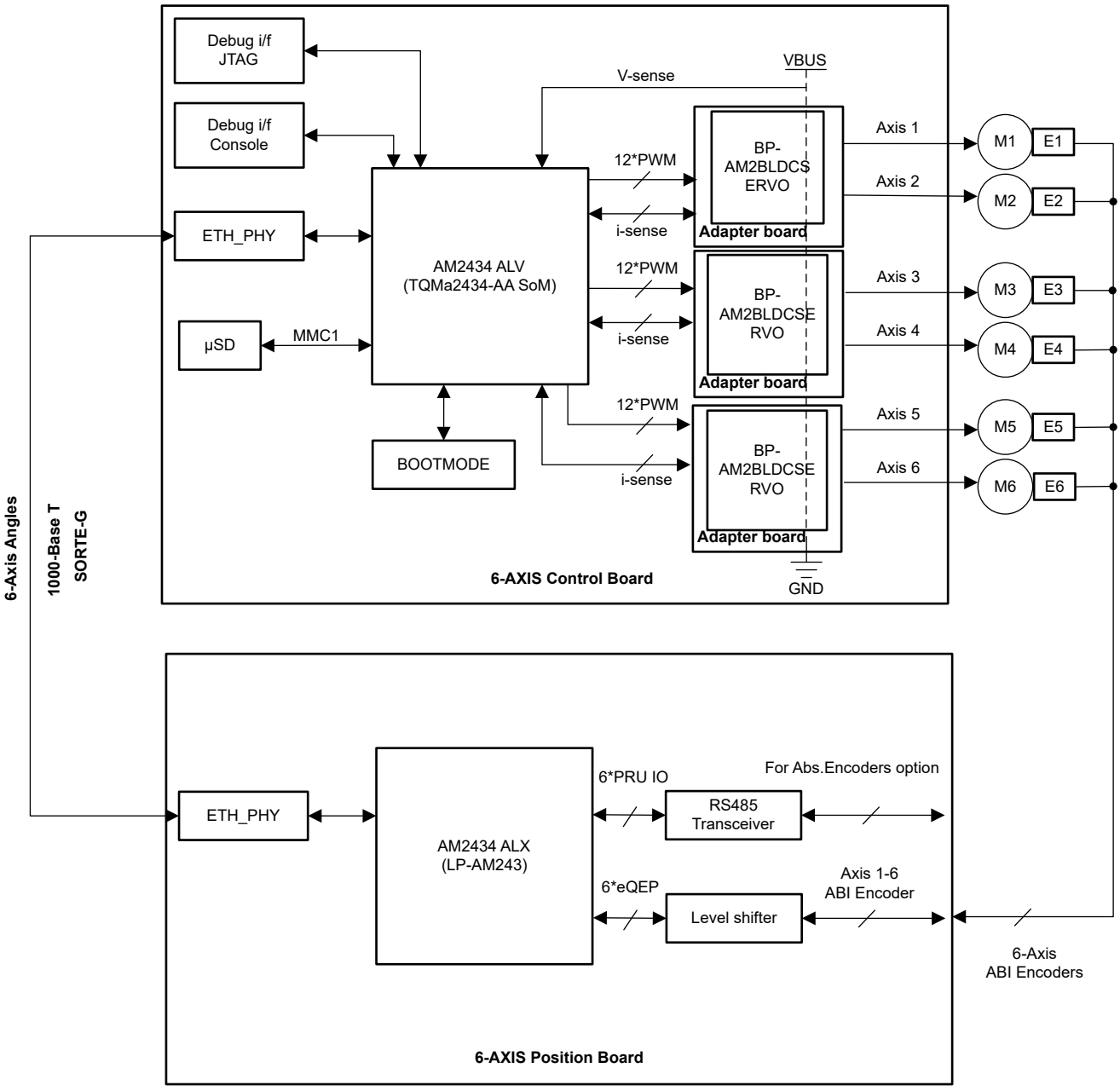


图 2-1. 与 TIDA-010948 配合使用时的系统设置

2.1 方框图

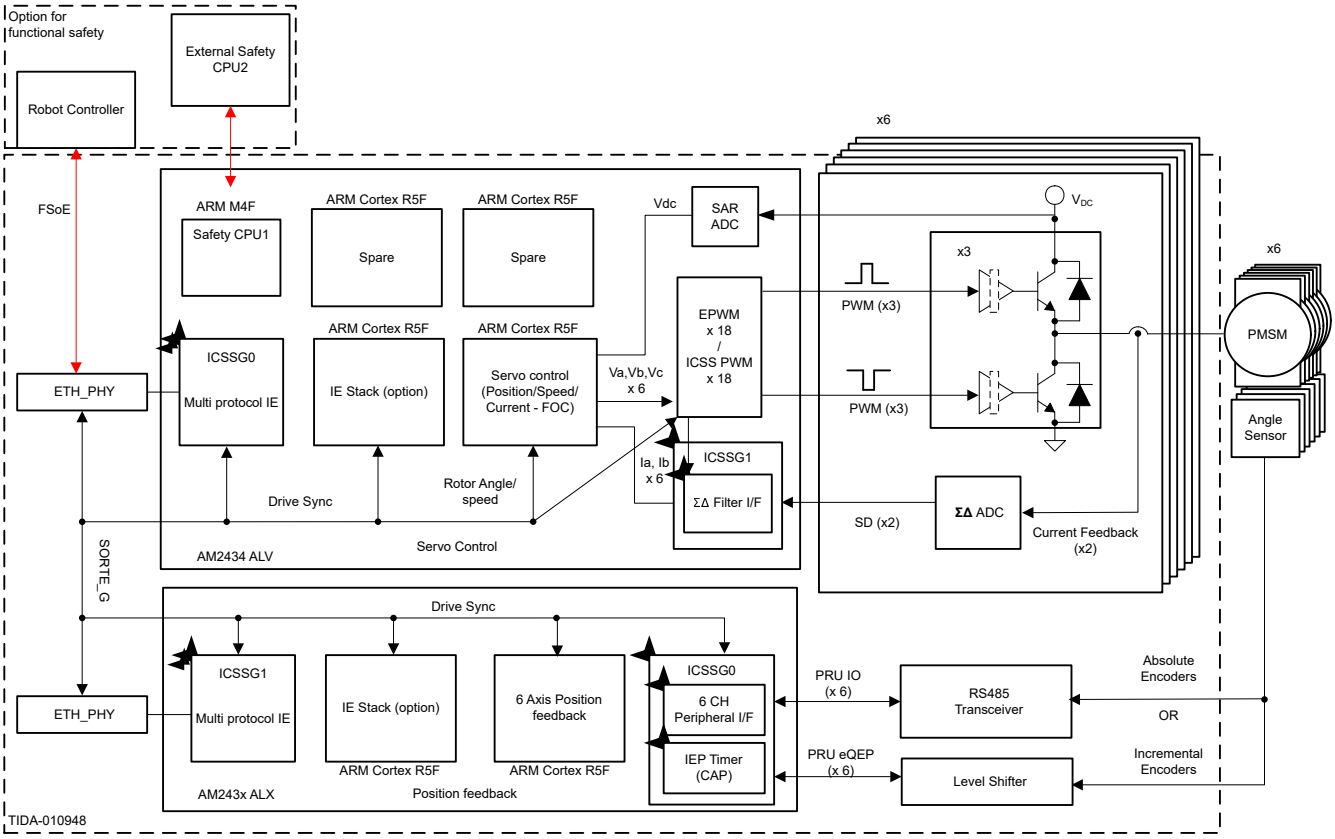


图 2-2. 采用 TIDA-010948 时的系统方框图

2.2 设计注意事项

6 轴伺服控制方案围绕一个核心实时路径构建，该路径包括两个子系统：

- 6 轴控制板，具有：
  - ICSSG0：SORTE_G 控制器固件或 EtherCAT 辅助控制器固件作为扩展选项。
  - R5FSS0_0：六个能够提供电流、速度或位置信息的独立闭环。具有 FOC 的闭环，用于六个具有编码器的直接连接电机。
  - R5FSS1_0：EtherCAT 辅助栈作为扩展选项实现。
  - EPWM：18 个通道的增强型 PWM 外设，用于根据 3 轴 FOC 环路的输出生成波形。
  - ICSSG1：Σ-Δ 滤波固件支持连续采样，并可针对六个直接连接电机的相电流反馈，在 slice0 和 slice1 中的 RTU 和 PRU 内核之间实现负载共享。
  - ICSSG_PRU PWM (ICSSG1)：具有死区置位的互补 ICSSG PWM 信号的 18 个通道，基于其他 3 轴 FOC 环路的输出生成。
- 6 轴位置板，具有：
  - ICSSG1：SORTE_G 器件固件或 EtherCAT 辅助控制器固件作为扩展选项。
  - R5FSS0_0：系统初始化和 LUT 生成。
  - R5FSS1_0：EtherCAT 辅助栈作为扩展选项实现。
  - ICSSG0：用于解码 4 通道编码器数据的 PRU_EQEP 固件。
  - EQEP：2 通道编码器数据解码。

功率级重复使用 BP-AM2BLDCSERVO 板，详细信息请参阅 TIDEP-01032。总共三个 BoosterPack™ 插件模块板加三个适配器板接收来自所有 AMC1035 器件的相电流数据，并从控制板发送 PWM 信号以驱动所有 DRV8316 器件。

图 2-3 展示了使用 IEP 计时器和 TSR 模块通过 SORTE_G 的以太网时间戳功能在位置板和控制板之间实现的时间同步。SORTE_G *IN* 数据包时序是确定性的，预先配置为匹配采样时间加位置数据的计算结果再加以太网延迟。在 1Gbps 时，数据包中的 64 字节耗时不到 1µs，外加用于以太网物理层延迟和线路延迟的 1µs。这样就可以使用双更新来获得更多计算时间或更快的 PWM 周期速度。

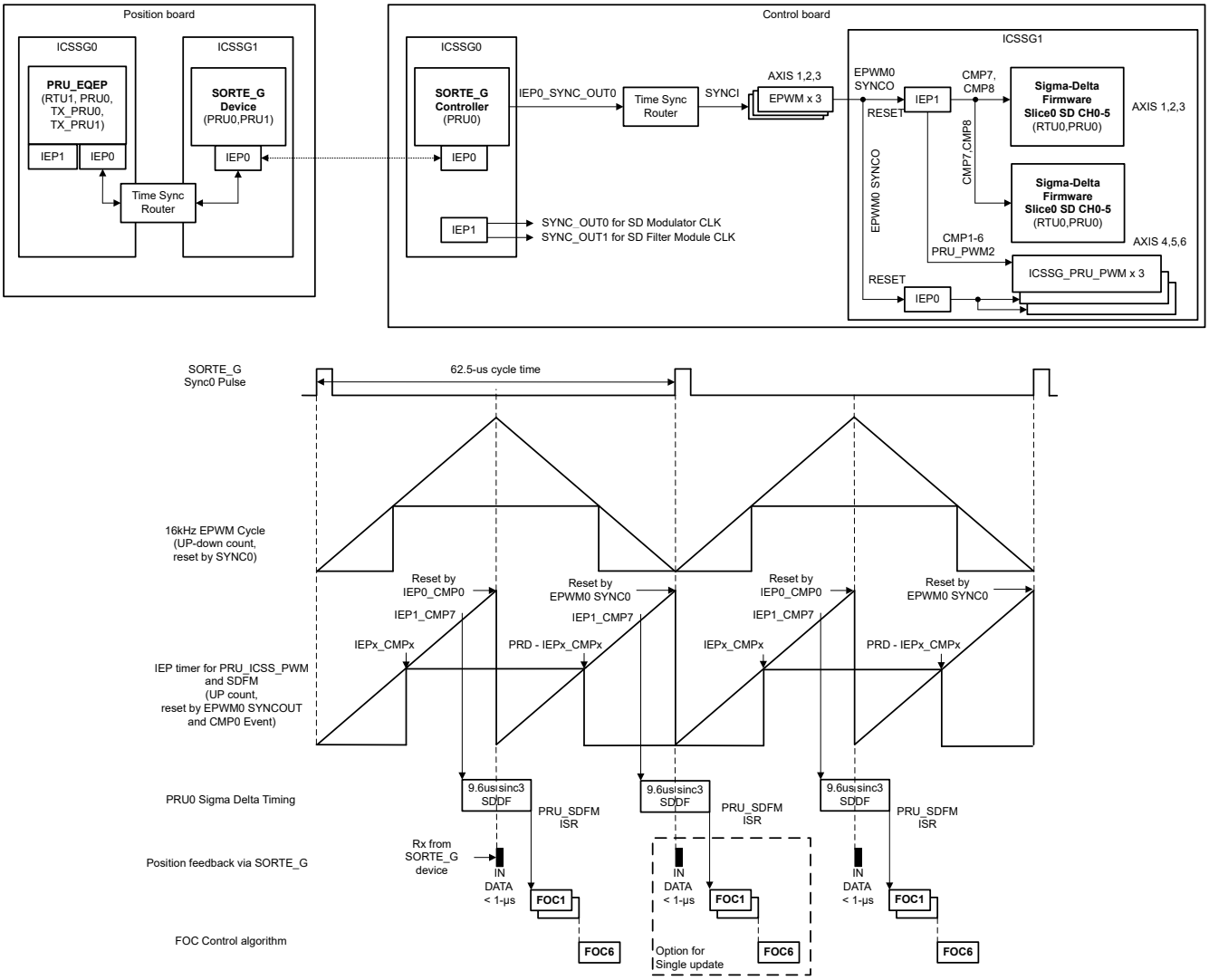


图 2-3. 时间同步和触发 FOC 时序 – 16kHz

2.3 主要产品 - AM243x 子系统

2.3.1 控制板 - SORTE_G 控制器接口

SORTE_G 控制器固件：

- 通过 PRU 项目 `SORTE_g_master` 生成的 `SORTE_g_master_PRU0.h`。
- `SORTE_g_master` 项目中的源代码 `main_PRU0.asm` 包含以下函数：
  - 针对不同状态安排任务管理器
  - 初始化 ICSSG0 PRU0 寄存器
  - 配置为 RGMII 接口和 IEP 时钟
  - 检查以太网链路状态
  - 生成发现数据包、参数化数据包，计算和保存同步延迟，并通过 TX_L2 FIFO 将数据包发送到器件
- `SORTE_g_master` 项目中的源代码 `rx.asm` 使用 PRU0 宽边接口和寄存器传输指令运行接收函数。

SORTE_G 配置和初始化：

- 源代码 `sorte_g_app_tq_control_board.c` 用于定义以下各项的 *IN* 数据包存储地址：
  - 轴 1 在 TCM 地址 *0x78000808* 收到电机角度数据
  - 轴 2 在 TCM 地址 *0x78000810* 收到电机角度数据
  - 轴 3 在 TCM 地址 *0x78000818* 收到电机角度数据
  - 轴 4 在 TCM 地址 *0x78000820* 收到电机角度数据
  - 轴 5 在 TCM 地址 *0x78000828* 收到电机角度数据
  - 轴 6 在 TCM 地址 *0x78000830* 收到电机角度数据
- 将周期时间配置为 62.5µs、*IN* 数据包偏移配置为 31.25µs。初始化 ICSSG0、MDIO 接口和 IEP 计时器，然后加载固件并通过 `generic_pruss_init()` 函数启用 PRU 内核。
- 使用来自 SORTE_G 客户端的 ICSSG0_IEP0_SYNCOUT0 同步 EPWM 块。

2.3.2 控制板 – SDFM 接口

图 2-4 展示了 Σ-Δ 滤波模块和调制器的时钟源分配。表 2-1 显示了 SDFM 信号。

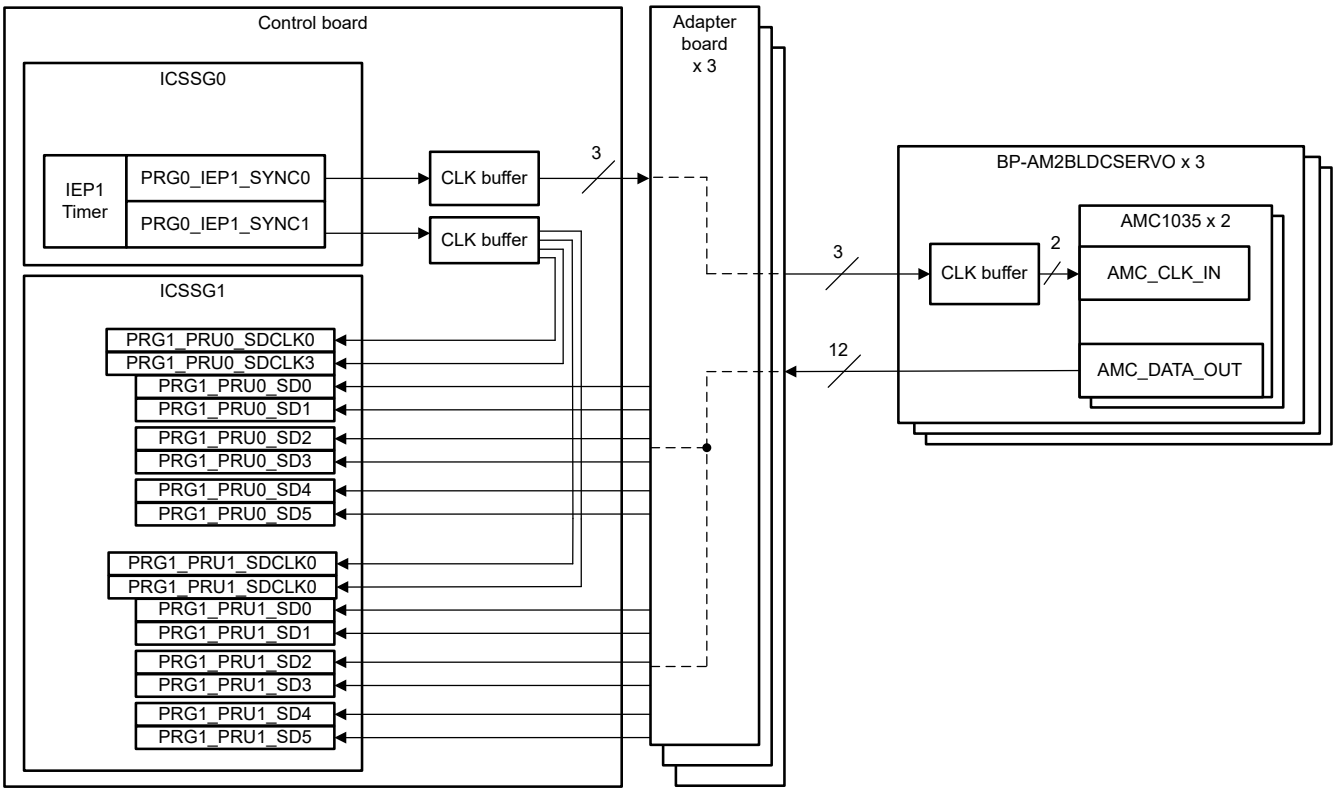


图 2-4. Σ-Δ 时钟和数据分配

表 2-1. SDFM 信号

子系统	信号名称	引脚名称	AM243x 焊球引脚	TQ_SoM 引脚
Axis1 A 相	DOUT_A1	PRG1_PRU0_SD0	U8	F5
Axis1 B 相	DOUT_B1	PRG1_PRU0_SD1	V8	G3
Axis2 A 相	DOUT_A2	PRG1_PRU0_SD2	V13	H1
Axis2 B 相	DOUT_B2	PRG1_PRU0_SD3	U13	H4
Axis3 A 相	DOUT_A3	PRG1_PRU0_SD4	U15	J2
Axis3 B 相	DOUT_B3	PRG1_PRU0_SD5	AA8	J5
Slice0-SD0_CLK	Slice0_SD0_CLK	PRG1_PRU0_SD0_CLK	Y7	F4

表 2-1. SDFM 信号（续）

子系统	信号名称	引脚名称	AM243x 焊球引脚	TQ_SoM 引脚
Slice0-SD3_CLK	Slice0_SD3_CLK	PRG1_PRU0_SD3_CLK	AA7	H3
Axis4 A 相	DOUT_A4	PRG1_PRU1_SD0	V11	M4
Axis4 B 相	DOUT_B4	PRG1_PRU1_SD1	Y12	N2
Axis5 A 相	DOUT_A5	PRG1_PRU1_SD2	AA13	N5
Axis5 B 相	DOUT_B5	PRG1_PRU1_SD3	V15	P2
Axis6 A 相	DOUT_A6	PRG1_PRU1_SD4	V14	P5
Axis6 B 相	DOUT_B6	PRG1_PRU1_SD5	AA10	R3
Slice1-SD0_CLK	Slice1_SD0_CLK	PRG1_PRU1_SD0_CLK	W11	M3
Slice1-SD3_CLK	Slice1_SD3_CLK	PRG1_PRU1_SD3_CLK	U11	P1
SDM_CLK_SOURCE	SDM_CLK	PRG0_IEP1_SYNC0	R2	A2
SDFM_CLK_SHIFT	SDFM_CLK	PRG0_IEP1_SYNC1	V5	B3

以下列表介绍了 SDFM 时钟：

- BoosterPack 板上 Σ-Δ 调制器的时钟源由采用 20MHz 的 ICSSG0 IEP1 的 SYNC0 循环输出提供。
- ICSSG1 Σ-Δ 滤波模块的时钟源由采用 20MHz 的 ICSSG0 IEP1 的 SYNC1 循环输出提供。时钟选择值设置为选项 2，这意味着 PRG1_PRUx_SD0_CLK 用于 SD 通道 0、1、2（SD0、SD1 和 SD2）。PRG1_PRUx_SD3_CLK 用于 SD 通道 3、4、5（SD3、SD4 和 SD5）；此处 x = 0 或 1，用于 slice0 和 slice1。
- SDFM 和 SDM 之间的时钟相移可以通过设置 IEP SYNC0 和 SYNC1 之间的延迟来配置，从而消除该延迟。IEP 的配置通过 `mclk_iep_sync.c` 中的 `init_IEP1_SYNC()` 函数实现。

以下列表中定义了 SDFM 中断：

- `hwiPrms.intNum = ICSSG_PRU_SDDF_INT_NUM0;`
- `hwiPrms.callback = pruSddfIrqHandler0;`

以下列表显示了 SDFM 输出数据缓冲区：

- R5F_0_0 的 TCMB 中的 `gSddfChSamps0[0-5]`（在 `gSddfChSampsRaw0` 中），用于轴 1、2 和 3
- R5F_0_0 的 TCMB 中的 `gSddfChSamps[0-5]`（在 `gSddfChSampsRaw` 中），用于轴 4、5 和 6

以下文件适用于 SDFM 固件：

- `sdfm_rtu_bin.h` 是片 0 的 SD0、SD1 和 SD2 的 PRU 固件
- `sdfm_pru_bin.h` 是片 0 的 SD3、SD4 和 SD5 的 PRU 固件
- `sdfm_rtu1_bin.h` 是片 1 的 SD0、SD1 和 SD2 的 PRU 固件
- `sdfm_pru1_bin.h` 是片 1 的 SD3、SD4 和 SD5 的 PRU 固件
- SDDF 初始化和固件加载通过 `sddf.c` 完成

SDFM 参数配置可以在结构 `gTestSdfmPrms0`（对于轴 1、2 和 3）和 `gTestSdfmPrms`（对于轴 4、5 和 6）中设置，包括 IEP 时钟频率、SD 时钟频率、首次采样触发时间、SD 时钟源选项和正常电流过采样率 (OSR)。对于本演示，默认设置使用：

- 250MHz IEP 时钟
- 20.833333MHz SD 时钟
- 26.45µs 首次采样触发时间
- 触发事件是 RTU 内核的 ICSSG1_IEP1_CMP7 和 PRU 内核的 ICSSG1_IEP1_CMP8
- 在 RTU 和 PRU 内核之间分担负载
- SD 时钟源的选项 2
- 使用 OSR 64 进行的正常电流采样

更多详细信息，请参阅 AM243x 电机控制 SDK：电流检测。

2.3.3 控制板 - EPWM 接口

表 2-2 显示了轴 1、2 和 3 的 EPWM*0* – *8* 信号。

表 2-2. EPWM*0* – *8* 信号

子系统	信号名称	外设	AM243x 焊球引脚 (ALV)	TQ_SoM 引脚
Axis1_PWM	PWM_A1_H	EPWM6	B14	W21
	PWM_A1_L	EPWM6	A15	V19
	PWM_B1_H	EPWM8	V1	B9
	PWM_B1_L	EPWM8	W1	D9
	PWM_C1_H	EPWM7	W20	AA6
	PWM_C1_L	EPWM7	W21	AB6
Axis2_PWM	PWM_A2_H	EPWM5	T19	V9
	PWM_A2_L	EPWM5	W19	U5
	PWM_B2_H	EPWM4	R18	V7
	PWM_B2_L	EPWM4	T21	V6
	PWM_C2_H	EPWM3	V18	W5
	PWM_C2_L	EPWM3	Y21	Y5
Axis3_PWM	PWM_A3_H	EPWM2	V19	Y7
	PWM_A3_L	EPWM2	T17	AA7
	PWM_B3_H	EPWM1	U19	W8
	PWM_B3_L	EPWM1	V20	Y8
	PWM_C3_H	EPWM0	U20	AA9
	PWM_C3_L	EPWM0	U18	AB9

使用 `init_pwms()` 函数的 EPWM 配置，包括：

- 配置 SYNCI 和 SYNCO 映射，以将所有 9 个 EPWM 组作为菊花链连接绑定在一起。将 `CTRLMMR_EPWM0_CTRL` 寄存器的位 [10 – 8] 设置为 2h，然后由 TSR 模块输出的 `TIMESYNC_INTRTR0_OUT_38` 触发 EPWM0_SYNCI。时间同步路由器输入 29 路由到输出 28，这意味着来自 SORTE_G 控制器的 ICSSG0_IEP0_SYNC0 会触发 EPWM0。此外，EPWM0_event 输出通过 *appEpwmCfg.cfgEt = TRUE* 启用，以复位用于 ICSSG_PRU_PWM 和 SDFM 模块的 IEP 计时器。将 `CTRLMMR_EPWM3_CTRL"` 和 `CTRLMMR_EPWM6_CTRL` 寄存器的位 [10 – 8] 从默认值 0h 设置为 1h，然后 EPWM3_SYNCI 和 EPWM6_SYNCI 分别由 EPWM2_SYNCO 和 EPWM5_SYNCO 触发，采用菊花链连接。
- 通过 *appEpwmCfg.epwmOutFreq = gEpwmOutFreq* 将 EPWM 频率设置为 16kHz。
- 通过 *appEpwmCfg.epwmTbCounterDir = EPWM_TB_COUNTER_DIR_UP_DOWN* 将 EPWM 计数器模式设置为向上/向下计数模式。
- EPWM 死区通过 `appEpwmCfg.cfgDb` 和 `appEpwmCfg.dbCfg.x` 的参数配置。
- EPWM 周期和比较值通过 `App_epwmConfig()` 函数计算，输出数据为所有轴的 `gEpwmPrdVal`。
- EPWM 中断由 *hwiPrms.intNum = PWM_C3_INTR* (EPWM0) 配置，回调函数为 *hwiPrms.callback = &App_epwmIntrISR*。

根据通过 `writeCmpA()` 函数获得的 FOC 计算结果更新 EPWM 比较事件。另请参阅 *AM64x/AM243x 技术参考手册 (TRM)*的 *EPWM 模块*部分。

2.3.4 控制板 - ICSSG_PRU PWM 接口

表 2-3 显示了轴 4、5 和 6 的 ICSSG_PRU PWM0-2 信号。

表 2-3. ICSSG_PRU PWM0-2 信号

子系统	信号名称	外设	IEP_CMP	AM243x 焊球引脚 (ALV)	TQ_SoM 引脚
Axis4_PWM	PWM_A4_H	ICSSG1_PRU_PWM2	IEP1_CMP1	N16	U7
	PWM_A4_L		IEP1_CMP2	N17	U8
	PWM_B4_H		IEP1_CMP3	P17	V10
	PWM_B4_L		IEP1_CMP4	Y18	V4
	PWM_C4_H		IEP1_CMP5	V21	AB8
	PWM_C4_L		IEP1_CMP6	R16	W6
Axis5_PWM	PWM_A5_H	ICSSG1_PRU_PWM1	IEP0_CMP7	V10	R4
	PWM_A5_L		IEP0_CMP8	U10	T2
	PWM_B5_H		IEP0_CMP9	AA11	T3
	PWM_B5_L		IEP0_CMP10	Y11	T5
	PWM_C5_H		IEP0_CMP11	Y10	U1
	PWM_C5_L		IEP0_CMP12	AA14	U2
Axis6_PWM	PWM_A6_H	ICSSG1_PRU_PWM0	IEP0_CMP1	U9	K2
	PWM_A6_L		IEP0_CMP2	W9	K3
	PWM_B6_H		IEP0_CMP3	AA9	K5
	PWM_B6_L		IEP0_CMP4	Y9	L1
	PWM_C6_H		IEP0_CMP5	V9	L3
	PWM_C6_L		IEP0_CMP6	U7	L4

ICSSG_PRU PWM 配置 `app_pruicss_pwm.c` 的 `init_pruIcssPwm()` 函数，包括：

- PWM 信号输出状态，包括由 API 函数 `PRUICSS_PWM_stateInit()` 定义的初始、活动和跳闸状态。
- 通过 API 函数 `PRUICSS_PWM_signalEnable()` 启用 PWM 信号。
- 由 API 函数 `PRUICSS_PWM_config()` 配置的 PWM 初始周期、占空比和死区。
- 由 API 函数 `PRUICSS_PWM_pruIcssPwmFrequencyInit()` 进行的 PWM 频率设置。
- 通过 `PRUICSS_PWM_IEP_Config()` 函数配置的 IEP 计时器可以：
  - 启用 IEP 影子模式。
  - 根据 IEP 时钟和 PWM 频率计算 CMP0 值作为 PWM 周期。
  - 通过 EPWM0_SYNCO 和 IEP CMP0 事件启用 IEP 复位。（IEP CMP0 值相比于 ICSSG_PRU PWM 周期的一半存在一个时钟周期延迟，因此 IEP CMP0 事件仅在该周期、*而非*在 PWM 计时器的零点复位 IEP。）

ICSSG_PRU_PWM IEP_CMP0 中断 `App_pruicssIep1Compare0IrqSet()` 用于设置软件标志，该标志用于根据 FOC 环路计算的结果更新 PWM 信号下一个上升沿的比较事件值：

- `hwiPrms.intNum` = CSLR_R5FSS0_CORE0_INTR_CMP_EVENT_INTROUTER0_OUTP_16
- `hwiPrms.callback` = &App_pruIcssPwmHalfDoneIrq
- 将 `ICSSG1_IEP0` 比较 0 事件编号 `TISCI_PRU_ICSSG1_IEP1_CMP0_SRC_INDEX` 的源索引定义为 12U

PWM 信号下一个下降沿的比较事件在 EPWM0 ISR 内更新。

更多详细信息，请参阅 *AM243x 电机控制 SDK：PRU-ICSS PWM 死区 EPWM 同步*。

2.3.5 控制板 - ICSSG_PRU IEP 计时器

以下参数适用于 IEP 计时器设置：

- ICSSG0：
  - 对于 SORTE_G 控制器，IEP0 设置为 250MHz 时钟。
  - IEP0 SYNC_OUT0，用于同步基于 EPWM 时间的计数器。
  - IEP1 SYNC_OUT0 20MHz 作为 SDM 的时钟源。
  - IEP1 SYNC_OUT1 20MHz 作为 SDFM 的时钟源。
- ICSSG1：
  - 对于 ICSSG_PRU PWM，IEP 时钟设置为 250MHz（与基于 EPWM 时间的计数器相同）。
  - IEP0 和 IEP1 CMP0 设置为 7812 作为 PWM 周期 (250000000/16000/2)。
  - IEP0 CMP1 至 CMP12 作为轴 5 和 6 PWM 的比较事件。
  - IEP1 CMP1 至 CMP6 作为轴 4 的比较事件。
  - IEP1 CMP7 至 CMP8 作为 SDFM 的比较事件（第一个采样触发器）。

2.3.6 控制板 – FOC 环路控制

在每个 PWM 周期的 SDFM ISR 内，六个独立 FOC 环路由 `AxisXFocLoopHandlerX()` 函数调用（对于 6 轴，此处 X = *1 至 6*）。图 2-3 描述了 PWM 更新和电流反馈的时序。在每个 PWM 周期内会生成两次 SDFM ISR。对于单更新，通过判断 EPWM0 ISR 中设置的软件标志 `gEpwmSyncFlag` 和 IEP_CMP0 事件 ISR 中设置的 `gUpdateNextRisingEdgeCmpValue` 来调用 FOC 环路。对于双更新，FOC 环路可以在每个 PWM 周期中调用两次。

`AxisXFocLoopHandlerX()` 函数包括：

- 在开始时通过工业以太网 SORTE_G 接收通过位置反馈板解码的最新机械角，原始角度数据格式为 Q23，需要转换为浮点格式作为变量 `mechThetaX`（对于 6 轴，此处 X = *1 至 6*）
- 通过以下方式校准机械角和用于 FOC 计算的电角之间的偏移：
  - 首先，使用纯开环旋转电机几个周期，以便通过位置板获得正确的角度，因为 *I* 脉冲可触发电机角度达到原始 *0* 度。
  - 第二，将电角（变量 `elecTheta`）强制设置为 *0* 度，并将 q 轴电流（变量 `parkIqOut`）强制设置为 *0*。为 d 轴电流设置一个恒定值（变量 `parkIdOut`），然后为电机仅生成没有反电动势 (EMF) 的扭矩电流。此时，电机不会旋转，但强制设置为 *0* 度电角，以便通过使用 `fmod()` 函数计算 `mechThetaX` 和 *90.0* 度之间的余数来了解机械偏移。然后，将机械偏移存储为变量 `mechAngleOffsetX`（对于 6 轴，此处 X = *1 至 6*）。
- 使用纯开环、闭合电流环路和闭合速度环路选项进行的 FOC 计算：
  - 用于 FOC 计算的 `elecTheta` 需要通过 `mechAngleOffsetX` 进行补偿，然后增大 4 倍，因为电机具有 4 对电极。确保 `elecTheta` 处于 0 至 360 度范围内，作为 `ti_r5fmath_sincos()` 函数的输入。
  - Clarke 和 Park 通过 `CLARKE_run_twoInput()` 和 `PARK_run()` 函数运行变换。
  - Park 逆变换使用 `IPARK_run()` 函数，空间向量生成由 `SVGEN_runCom()` 函数组成。
  - 对于纯开环，增量电机角度由固定值 `myMechDeltaX` 给出，q 轴电流由 `gIq` 给出。
  - 对于闭环，电流和速度的 PI 控制器都通过 `DCL_runPIParallel()` 函数实现。电流和速度目标在 `gIqArray` 和 `gSpdArray` 数组中定义。PI 常量可以在 `init_pids()` 函数中调整。

2.3.7 位置板 – SORTE_G 器件接口

SORTE_G 器件软件包括：

- 固件 `SORTE_g_device_PRU0.h` 和 `SORTE_g_device_PRU1.h` 通过 PRU 项目 `SORTE_g_device` 生成。使用 LP-AM243 上的跳线短接 J6 引脚 59 和 J6 引脚 60，使电路板成为器件。
- 源代码 `main.asm` 包括以下函数：
  - 初始化 PRU 寄存器
  - 针对不同状态配置任务管理器
  - 复位和配置 MII_G_RT 模块
  - 生成 MDIO 链路事件以读取当前以太网链路状态

  - 执行调用中断事件管理器和 SORTE_G 状态的控制循环
- 源代码 `sorte_g_app.c` 定义从器件发送的 IO 交换数据，这些数据：
  - 通道 0 EQEP 将电机角度发送到 ICSSG1 DRAM0 地址 *0x30081504*
  - 通道 1 EQEP 将电机角度发送到 ICSSG1 DRAM0 地址 *0x3008150C*
  - 通道 2 EQEP 将电机角度发送到 ICSSG1 DRAM0 地址 *0x30081514*
  - 通道 3 EQEP 将电机角度发送到 ICSSG1 DRAM0 地址 *0x3008151C*
  - 通道 4 EQEP 将电机角度发送到 ICSSG1 DRAM0 地址 0x30081524
  - 通道 5 EQEP 将电机角度发送到 ICSSG1 DRAM0 地址 *0x3008152C*

2.3.8 位置板 – PRU_EQEP 接口

图 2-5 展示了用于从增量编码器提供 4 通道位置反馈的 PRU EQEP 固件架构。PRU_EQEP 设计包括一个接口，用于连接四个包含 A、B 和 I 信号的增量编码器。这些信号通过 TXB0106RGYR 电平转换器传递，以将编码器信号电平与微控制器的逻辑电平相匹配。GPIO 配置为在信号 A 和 B 的上升沿和下降沿以及在信号 I 的上升沿触发中断。这些中断路由到 ICSSG 中的 IEP。IEP 有一个 64 位计数器，用于捕获计数器值以测量电机 RPM 和角度。PRU 内核配有任务管理器，用于配置在 PRU 中执行的多项任务。这些任务捕获计数器值，并计算电机 RPM 和角度。另外，这些任务会触发 *XFR2VBUS* 小工具来读取 GPIO 的状态，这些状态基本上表示用作计算基本输入的编码器信号的状态。

表 2-4 显示了 LP-AM243 侧的 4 通道 ABI 信号，这些信号是位置板上电平转换器的输出。

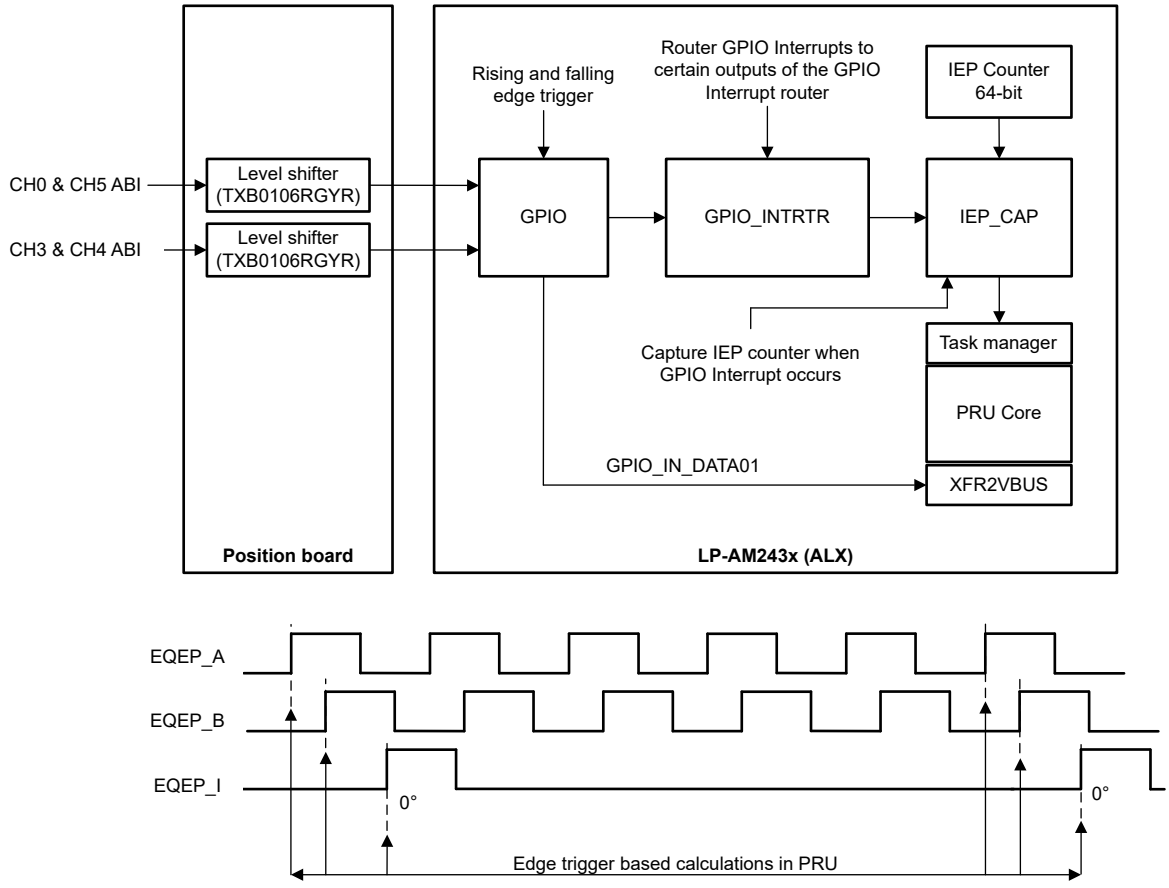


图 2-5. PRU_EQEP 子系统架构和接口

表 2-4. PRU_EQEP 信号

子系统	信号名称	外设	PRU 内核	LP-AM243 HEADER	AM243x 焊球引脚 (ALX)
PRU_EQEP CH0	EQEP_A_CH0	GPIO1_20	ICSSG0_RTU1	BP.11	L5
	EQEP_B_CH0	GPIO1_21		BP.67	J2
	EQEP_I_CH0	GPIO1_33		BP.71	T4
PRU_EQEP CH3	EQEP_A_CH3	GPIO1_0	ICSSG0_PRU0	BP.33	J3
	EQEP_B_CH3	GPIO1_1		BP.32	J4
	EQEP_I_CH3	GPIO1_6		BP.48	H2
PRU_EQEP CH4	EQEP_A_CH3	GPIO1_2	ICSSG0_TXPRU0	BP.31	G1
	EQEP_B_CH3	GPIO1_3		BP.19	H1
	EQEP_I_CH3	GPIO1_7		BP.44	E2
PRU_EQEP CH5	EQEP_A_CH3	GPIO1_4	ICSSG0_TXPRU1	BP.17	K2
	EQEP_B_CH3	GPIO1_5		BP.13	F2
	EQEP_I_CH3	GPIO1_8		BP.51	H5

PRU_EQEP 软件包含以下内容：

- 固件文件：
  - 通道 0 编码器的固件 `EQEP_position_feedback_CH0_ICSS_G0_RTU_PRU1.h`
  - 通道 3 编码器的固件 `EQEP_position_feedback_CH3_ICSS_G0_PRU0.h`
  - 通道 4 编码器的固件 `EQEP_position_feedback_CH4_ICSS_G0_TX_PRU0.h`
  - 通道 5 编码器的固件 `EQEP_position_feedback_CH5_ICSS_G0_TX_PRU1.h`
- GPIO 配置：
  - 对于 `EQEP_A` 和 `EQEP_B`，通过 `GPIO_SET_RIS_TRIG` 寄存器启用上升沿触发中断，通过 *GPIO_SET_FAL_TRIG* 启用下降沿检测。*I* 信号只需要上升边沿中断，因为此信号代表每转中用作参考的单脉冲。
  - GPIO 中断路由器模块充当对输入到输出目的地的 GPIO 中断信号实现多路复用所必需的中间媒介。配置 GPIO 边沿触发的中断后，这些中断会路由到 GPIO 中断路由器的输入。然后，路由器会将这些输入映射到特定输出。GPIO 中断路由器的输出随后传送到 ICSSG 内的 IEP 模块。

备注

GPIO 中断路由器内的中断配置和路由只能由 SCI 客户端执行。

- ICSSG_PRU 配置：
  - 将 ICSSG0 IEP 计时器设置为 333MHz，每个计数器为 1ns。
  - 使用绕回模式将 CMP0 配置为最大值，并将 CMP5 设置为 62.5 µs 以触发任务管理器进行速度计算。
  - `EXT_CAP_EN[5:0]` 设置为启用 GPIO 中断路由器输出的 IEP 捕获。
  - 任务 `sub_task_TS2_S0` 处理速度计算。`sub_task_TS2_S1` 使用 `XFR2VBUS` 小工具处理电机角度计算。`sub_task_TS2_S2` 使用编码器 A 转换的精确时间戳处理 EQEP_A 信号检测。`sub_task_TS2_S3` 使用编码器 B 转换的精确时间戳处理 EQEP_B 信号检测。`sub_task_TS2_S4` 处理 EQEP_I 信号检测，以标记编码器旋转周期中的基准位置，这样可为角度值到零度提供复位点。

2.3.9 位置板 – SoC EQEP 模块接口

增量编码器的两个位置反馈通道使用 SoC EQEP 模块。表 2-5 显示了 LP-AM243 侧的 2 通道 ABI 信号。

表 2-5. SoC EQEP 模块信号

子系统	信号名称	外设	LP-AM243 HEADER	AM243x 焊球引脚 (ALX)
SoC EQEP CH1	EQEP_A_CH1	EQEP1	J12.1	L2
	EQEP_B_CH1		J12.2	L3
	EQEP_I_CH1		J12.3	R5
SoC EQEP CH2	EQEP_A_CH2	EQEP2	J21.1	B14
	EQEP_B_CH2		J21.2	A15
	EQEP_I_CH2		J21.3	B13

SoC EQEP 在 `generic_pruss_init()` 函数中配置，包括：

- 将 QEP 周期配置为 16kHz，QEP 时钟为 125MHz。
- 根据电机规格，使用零值初始化位置计数器，并将计数器的最大值设置为 4000。
- 配置 QEP 位置计数器源，锁存条件并利用索引事件进行复位。
- 通过单位超时配置并启用中断：
  - 通道 1 的中断号为 144，中断回调函数为 `EQEP1_ISR`
  - 通道 2 的中断号为 145，中断回调函数为 `EQEP2_ISR`

另请参阅 *AM64x/AM243x 技术参考手册 (TRM)*的 *EQEP 模块*部分。

3 系统设计原理

3.1 位置板 – 系统初始化

按照以下步骤将位置板初始化为 SORTE_G 器件，并使用 `generic_pruss_init()` 函数对 R5F_0_0 内核中的所有六个编码器通道的 ABI 信号进行解码。设置控制板*之前*加载并运行位置板代码。

1. 使用映像 `sbl_null_sciclient.release.hs_fs.tiimage` 通过次级引导加载程序 (SBL) 预加载配置，以启用系统命令解释程序（SCI 客户端）。
2. 复制工作区文件夹下的 `include.zip` 和 `pru_fw_common.zip` 文件以了解 SORTE_G 固件的使用情况。
3. 通过清除数据 RAM 并设置入口点来初始化 ICSSG PRU。
4. 将电机方向和速度 LUT 写入到 PRU 数据 RAM。
5. 将 GPIO 引脚模式设置为 PRU_EQEP 的输入和 SoC QEP 的 EQEP，将 GPIO 中断模式设置为上升沿检测。
6. 为 SORTE_G 器件设置 RGMII 接口和 MII_G_RT 模块。
7. 为 PRU_EQEP 设置 ICSSG0 IEP 计时器，为 SORTE_G 设置 ICSSG1 IEP 计时器。
8. 设置 SoC QEP 模块参数和中断。
9. 加载并运行 SORTE_G 器件和 PRU_EQEP 固件。然后，将 6 通道解码的电机角度数据复制到预定义地址的 PRU 数据存储器，并准备好在每个预定义的 PWM 周期中发送。

3.2 位置板 – 中断

- 通道 1 EQEP1 中断号为 144，中断回调函数 `EQEP1_ISR` 用于通过读取通道 1 QEP 节拍数计算电机角度，并将角度数据写入预定义的 PRU 数据存储器中。
- 通道 2 EQEP2 中断号为 145，中断回调函数 `EQEP2_ISR` 用于通过读取通道 2 QEP 节拍数计算电机角度，并将角度数据写入预定义的 PRU 数据存储器中。
- 当检测到相对 IO 引脚上的 AB 信号的上升沿和下降沿时，通道 0 和通道 3 至通道 5 使用 GPIO 中断。

3.3 控制板 – 系统初始化

按照以下步骤将控制板初始化为 SORTE_G 控制器，并使用 `single_chip_servo_remote_core_start()` 函数接收用于控制 R5F_0_0 内核中所有 6 轴电机的六通道电机角度数据。设置位置板*之后*加载并运行控制板代码。

1. 控制板使用 `TQ-SoM`，因此 SOM 上的闪存需要手动配置。闪存必须使用 `tq_sbl_uart_uniflash.hs_fs.tiimage` 映像和 Python® `uart_uniflash.py`（在 `mcu_plus_sdk\tools\` 文件夹下）工具来刷写 SBL。将 `tq_sbl_uart_uniflash.hs_fs.tiimage` 和 `default_sbl_null_tq.cfg` 复制到 SDK 文件夹 `mcu_plus_sdk\tools\boot\sbl_prebuilt\am243x-evm` 中。
2. 使用 `init_gpio_state()` 函数设置 GPIO 引脚方向和初始值。
3. 通过 `enable_pwm_buffers(FALSE)` 函数禁用 PWM。
4. 使用 `init_pruIcssPwm()` 函数配置 ICSSG_PRU_PWM 以使轴 4、5、6 为三相互补，并将初始占空比设置为 50%。
5. 使用 `init_pwms()` 函数配置 EPWM 以使轴 1、2、3 为三相互补，并将初始占空比设置为 50%。
6. 通过 `init_sddf()` 函数为所有 6 个轴的 SDFM 配置 ICSSG1 RTU0、RTU1、PRU0、PRU1 内核，并通过负载共享模式加载 4 个 SDFM 固件。由 `init_IEP1_SYNC()` 函数设置 SD 时钟的初始 ICSSG0 IEP1 计时器 SYNC0 和 SYNC1。通过 `start_ICSSG1_IEPx()` 函数启动 ICSSG1 IEP 计时器。
7. 为 SORTE_G 控制器配置 ICSSG0 PRU0 内核，并通过 `generic_pruss_init()` 函数加载 SORTE_G 控制器固件。
8. 使用 `init_pids()` 函数初始化 FOC 控制的参数。
9. 为所有 6 个轴启用 EPWM 输出缓冲器。

3.4 控制板 – 中断

- EPWM 中断 (16kHz) – 中断号为 108 (EPWM0)，中断回调函数 `App_epwmIntrISR` 用于复位 IEP 计时器并将轴 4、轴 5 和轴 6 的下一个 ICSSG_PRU_PWM 下降沿更新为 PWM 周期的一半减去上升沿。该中断将值作为单次更新方案进行比较，或根据 FOC 计算结果作为双更新方案进行比较。

- ICSSG_PRU_PWM 中断 (16kHz) – IEP_CMP0 中断 `App_pruicssIep1Compare0IrqSet()` 用于设置软件标志 `gUpdateNextRisingEdgeCmpValue`，以根据 FOC 环路计算的结果更新 PWM 信号下一个上升沿的比较事件值。
- SDFM 中断 (32kHz) – 中断号为 251 (PRU_ICSSG1_PR1_HOST_INTR_PEND_3)，中断回调函数 `pruSddfIrqHandler0` 用于触发 6 个轴的 FOC 环路。从样本 8192 到 16384，计算 SDFM 通道偏移 `gSddfChOffset[x]`，并补偿 FOC 环路中的电流反馈。

4 硬件、软件、测试要求和测试结果

4.1 硬件要求

需要以下设备来测试此参考设计：

- 一个控制板 - TIDA-010948_CB，如 图 4-1 和 图 4-2 所示
- 三个适配器板 – TIDA-010948_DB，如 图 4-3 和 图 4-4 所示
- 三个 BP-AM2BLDCSERVO — AM2x 无刷直流 (BLDC) 伺服电机 BoosterPack™
- 一个位置板 – TIDA-010948_PB，如 图 4-5 和 图 4-6 所示
- 一个 LP-AM243 评估板 – AM243x 通用 LaunchPad™ 开发套件，适用于基于 Arm® 的 MCU
- 六个 LVSERVOMTR 电机 - 低压伺服电机 - 低压伺服（编码器）电机和线束

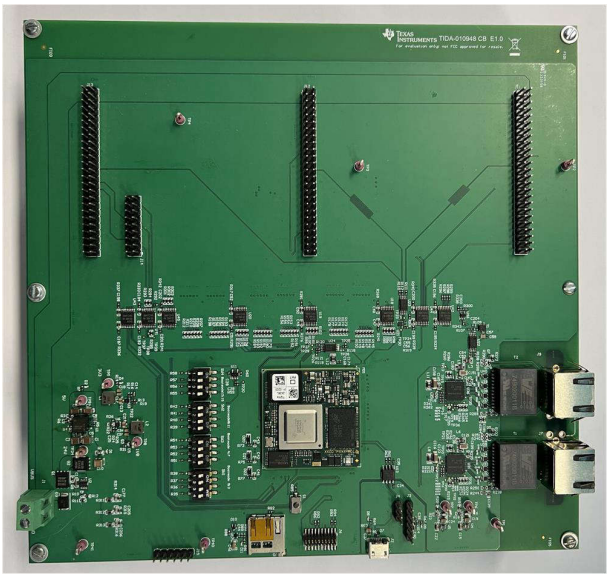


图 4-1. TIDA-010948_CB PCB 顶视图概览

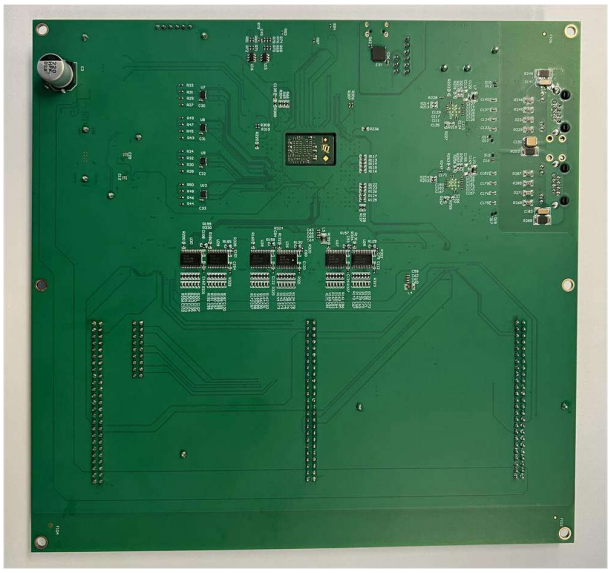


图 4-2. TIDA-010948_CB PCB 底视图概览

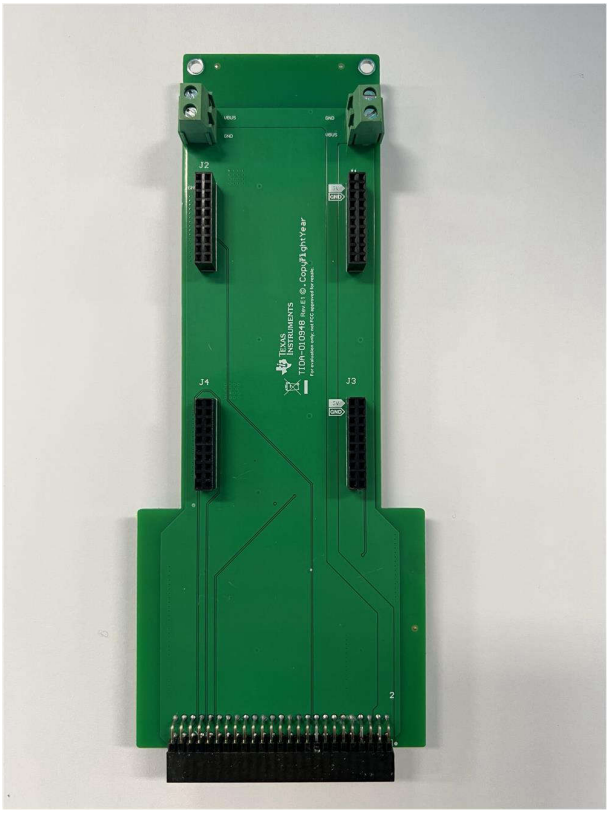


图 4-3. TIDA-010948_DB PCB 顶视图概览

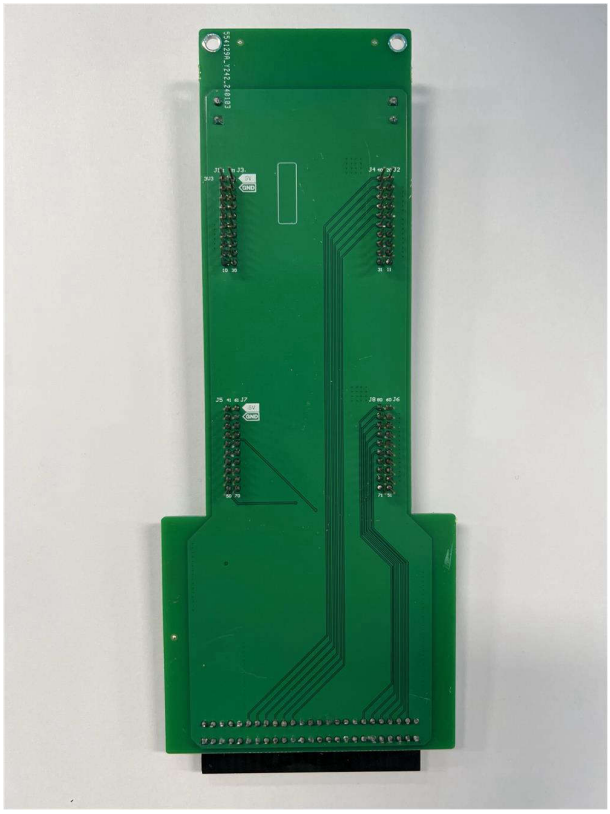


图 4-4. TIDA-010948_DB PCB 底视图概览

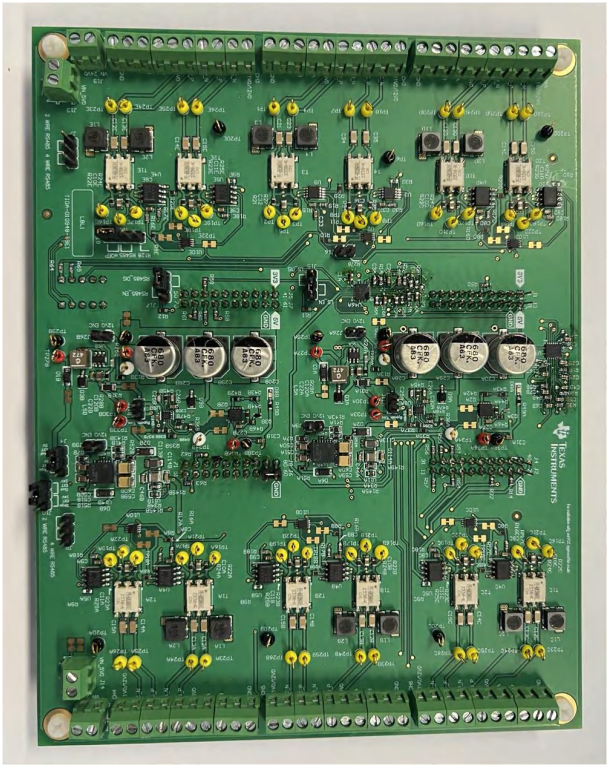


图 4-5. TIDA-010948_PB PCB 顶视图概览

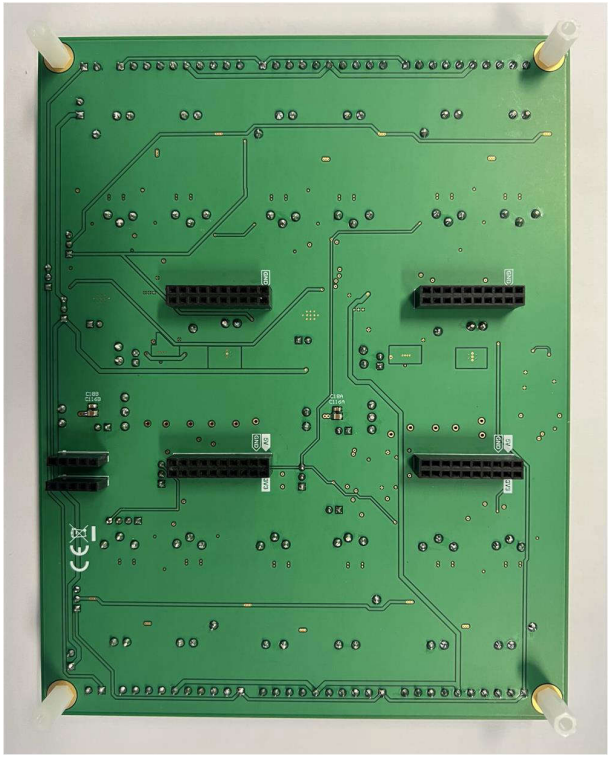


图 4-6. TIDA-010948_PB PCB 底视图概览

图 4-7 展示了系统演示概览。

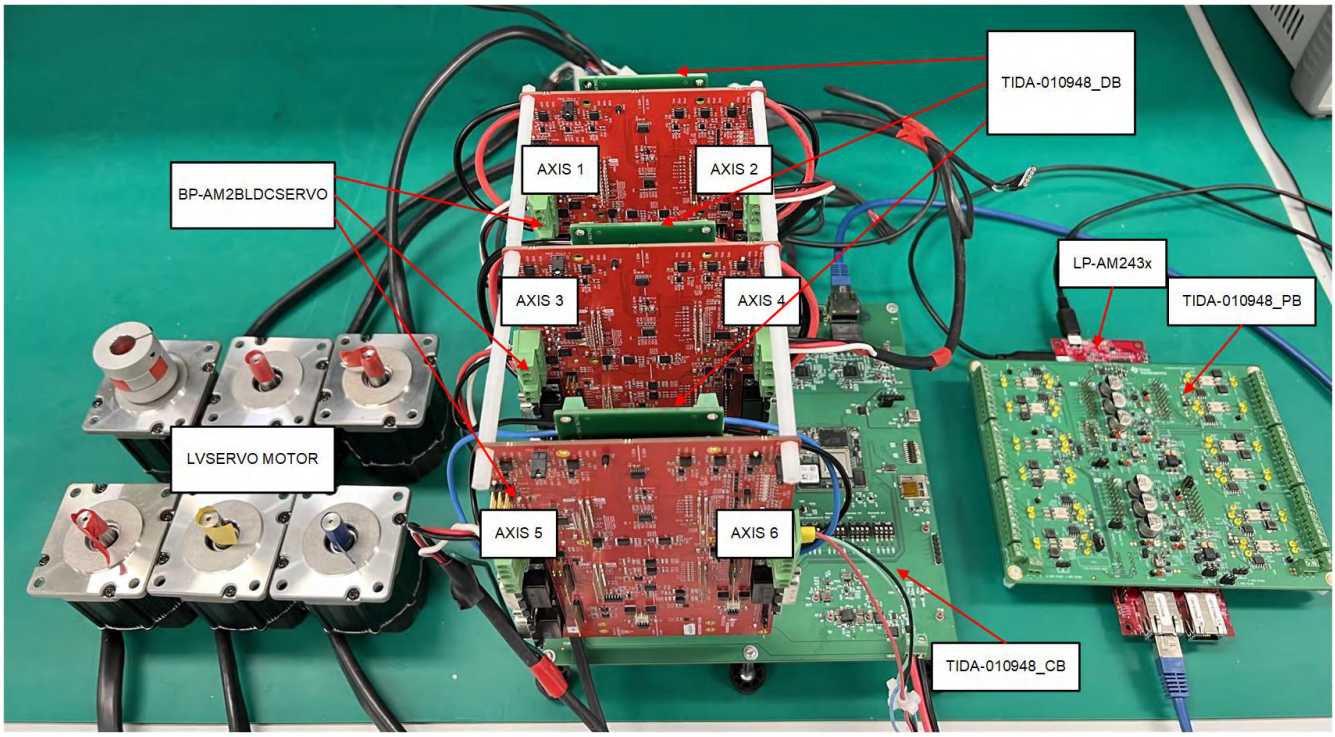


图 4-7. TIDA-010948 系统演示概览

4.1.1 系统演示设置

以下步骤介绍了系统演示设置：

1. 将 3 个适配器板 (TIDA-010948_DB) 与 3 个 BP-AM2BLDCSERVO 板（BoosterPack 板）相连，然后将此装置连接到基本控制板 (TIDA-010948_CB)。一个 BoosterPack 板加一个适配器板用作两个轴的功率级（适配器板的 J1 – J4 分别连接到 BoosterPack 板的 J1 – J4）。
2. 将一个适配器板的 J5 连接到控制板上的 J11，作为轴 1 和轴 2 的功率级。第二个适配器板的 J5 连接到控制板上的 J12，作为轴 3 和轴 4 的功率级。第三个适配器板的 J5 连接到控制板上的 J13，作为轴 5 和轴 6 的功率级。
3. 控制板的 J1 连接是系统输入 $24V_{DC}$。
4. 适配器板上的 J6 连接从控制板获得直流链路电源。使用电缆将该 J6 连接连到 BoosterPack 板的 J5，以便将电源传递到 BoosterPack 板。适配器板上的 J7 连接从控制板获得直流链路电源。使用电缆将该 J7 连接连到 BoosterPack 板的 J6，以便将电源传递到 BoosterPack 板。完成所有 3 个适配器板的相同连接，以便为所有六个轴供电。两个 BoosterPack 板之间的距离约为 83mm，需要使用支柱来固定 BoosterPack 板加适配器板与控制板之间的安装。图 4-8 和 图 4-9 显示了直流链路和电机电源线的连接。
5. 控制板引导：
   - 短接控制板上 J4 的引脚 1 和引脚 2 以向 UART 供电
   - 将 USB 电缆连接到 UART 终端的 J2 上
   - 将 JTAG 电缆连接到 J6 上（默认公接头是 0.05 英寸的 CM20 引脚，需要阻断引脚 6）
   - 将引导模式设置为 UART BOOT，设置 SW4 *“0000”*、SW2 *“1011”*、SW3 *“1100”*、SW1 *“1101”*
   - 打开 UART 终端，字符 *C* 会每 2 到 3 秒打印一次。该进程完成后，关闭 UART 终端。
   - 使用 Python® `uart_uniflash.py` 刷写 `SBL_null`。对 `uniflash` 使用 TQ 映像，将 `tq_sbl_uart_uniflash.hs_fs.tiimage` 和 `default_sbl_null_tq.cfg` 复制到 SDK 文件夹 `mcu_plus_sdk\tools\boot\sbl_prebuilt\am243x-evm`

- 关闭电路板电源，通过设置 SW4 “*0000*”、SW2 “*0100*”、SW3 “*1110*”、SW1 “*1100*” 来设置 OSPI 引导模式，然后为电路板供电。SBL_NULL 信息出现在 UART 终端中。图 4-10 显示了引导模式开关以及 JTAG 和 UART 的连接。
- 在目标 ccxml 文件中将 JTAG 电源电压设置为 1.8V，如 图 4-11 所示。

6. 位置板 (TIDA-010948_PB) 由 J13 上 的 $5V_{DC}$ 供电。为 J4 上的双通道选择外部 5V，为 J10 上的双通道选择外部 24V。启用 J15 上的电平转换器，并禁用 J17 上的 RS-485 接口。将 J6 引脚 59 短接至 J6 引脚 60 (GND)，以将位置板设置为 SORTE_G 器件。J3A 至 J3F 分别用于连接 6 个通道的编码器信号。J1、J2、J5、J6、J12 和 J21 连接到 LaunchPad LP-AM243x。图 4-12 显示了位置板的设置。
7. LP-AM243x 用作对六轴编码器信号和 SORTE_G 器件进行解码的 MCU 平台。有关设置和引导初始化的信息，请参阅 AM243x MCU+ SDK：EVM 设置。如 节 2.3.8 所述，LP-AM243x 需要通过 SBL 使用 `sbl_null_sciclient.release.hs_fs.tiimage` 映像预加载配置来启用 SCI 客户端，请将映像文件复制到 *mcu plus sdk* 文件夹：
`mcu sdk folder\tools\boot\sbl_prebuilt\am243x-lp\`。

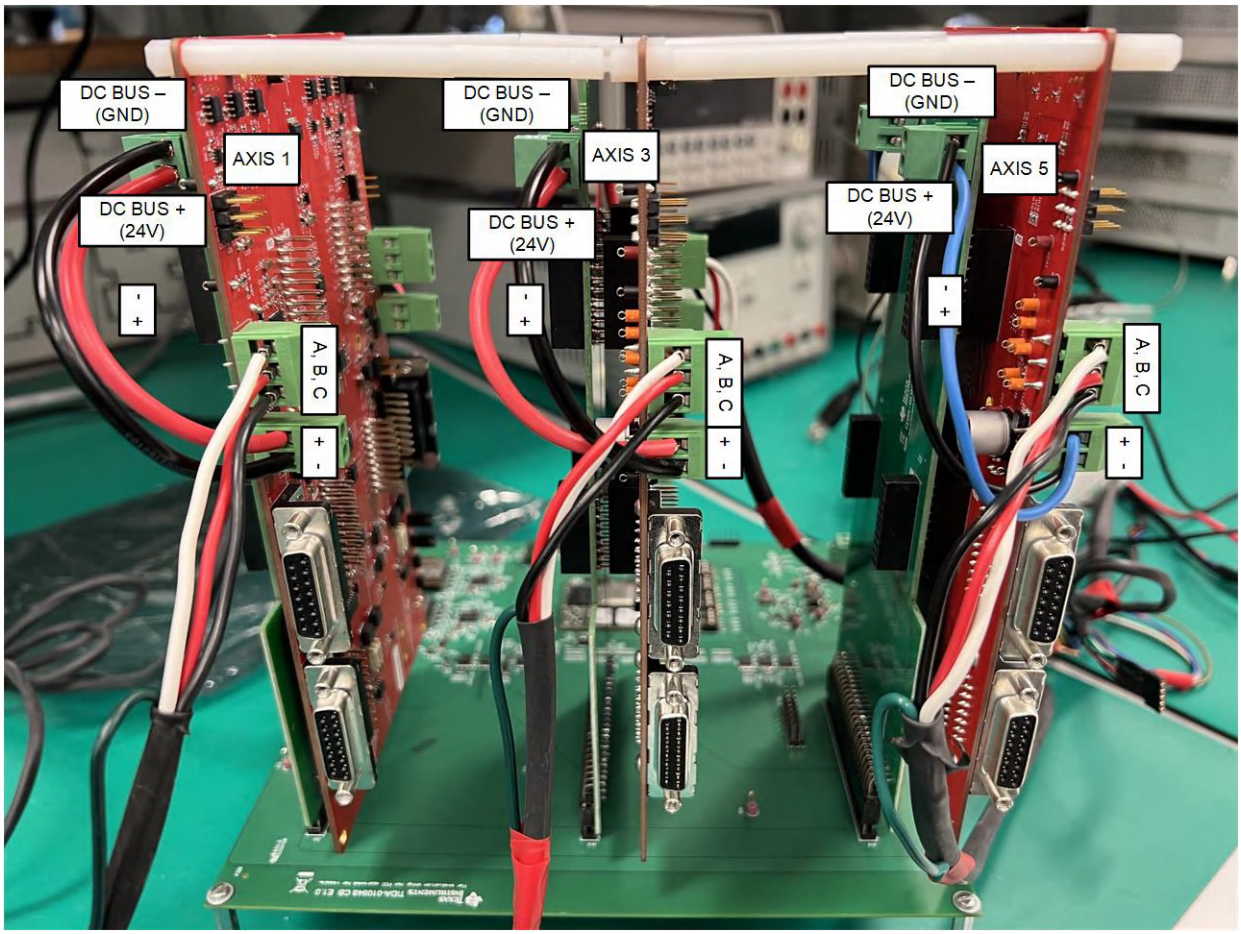


图 4-8. 直流链路连接和电机相电源 – 轴 1、3、5

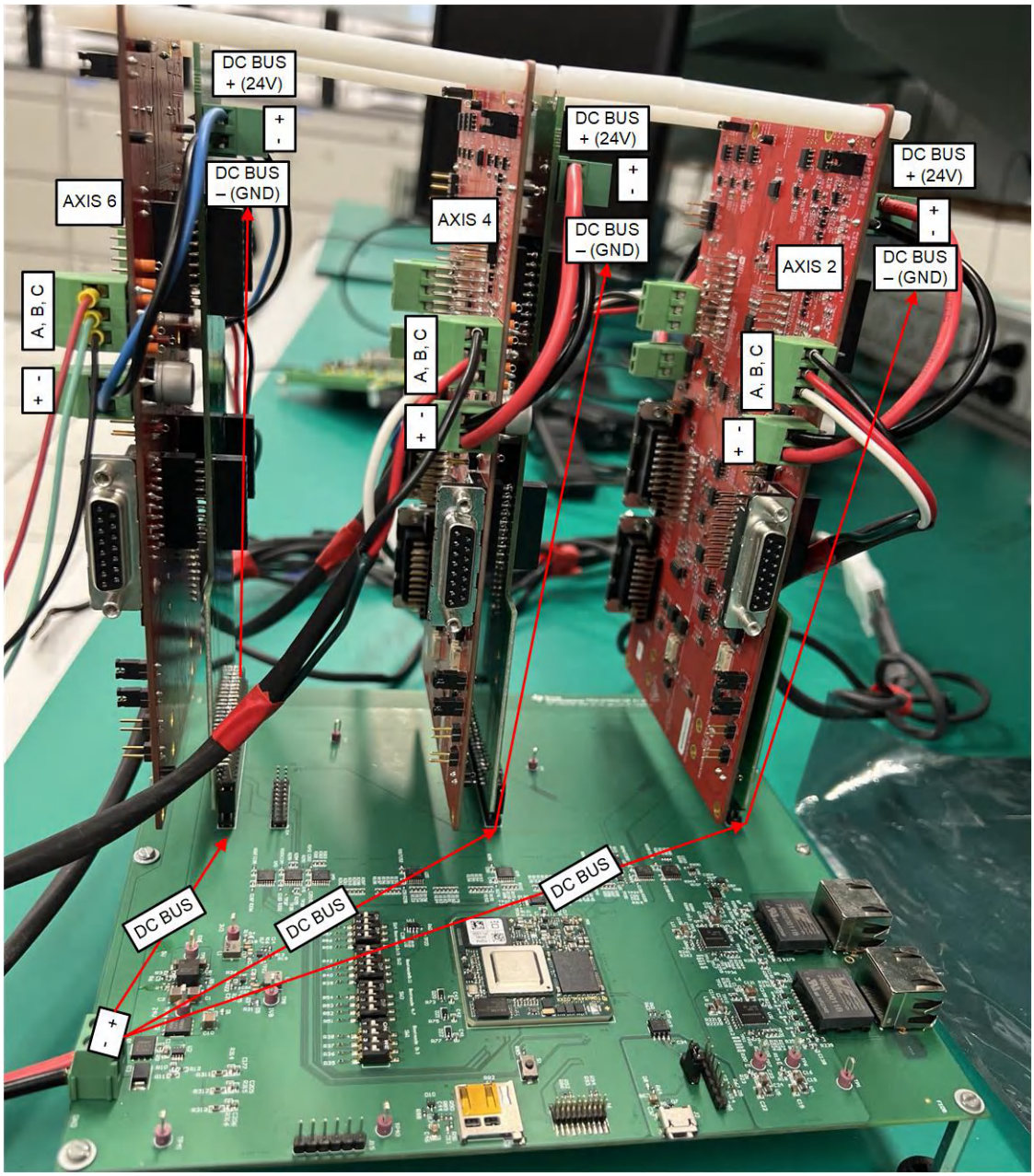


图 4-9. 直流链路连接和电机相电源 – 轴 2、4、6

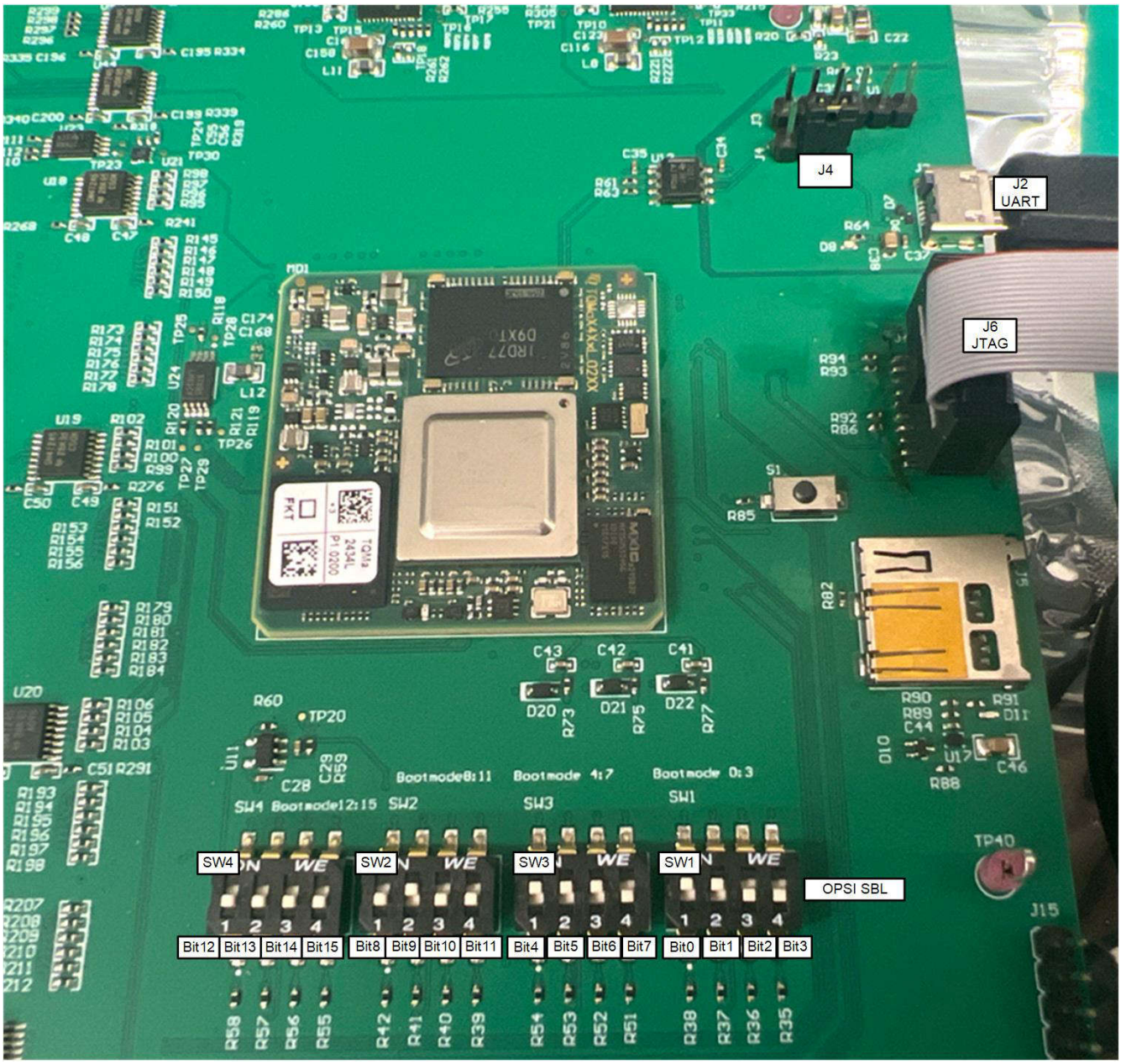


图 4-10. 控制板上的 JTAG 和 UART 的引导模式开关和连接

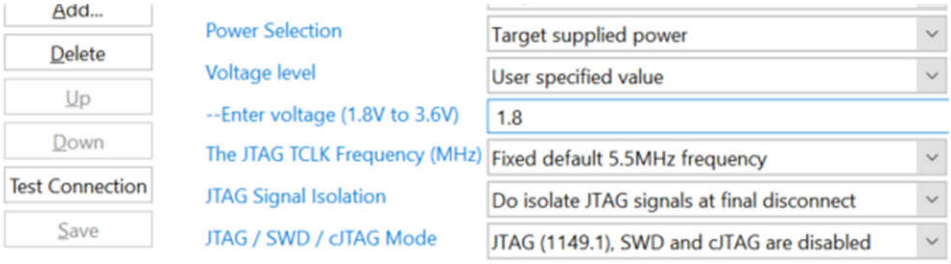


图 4-11. JTAG 电源的目标文件设置

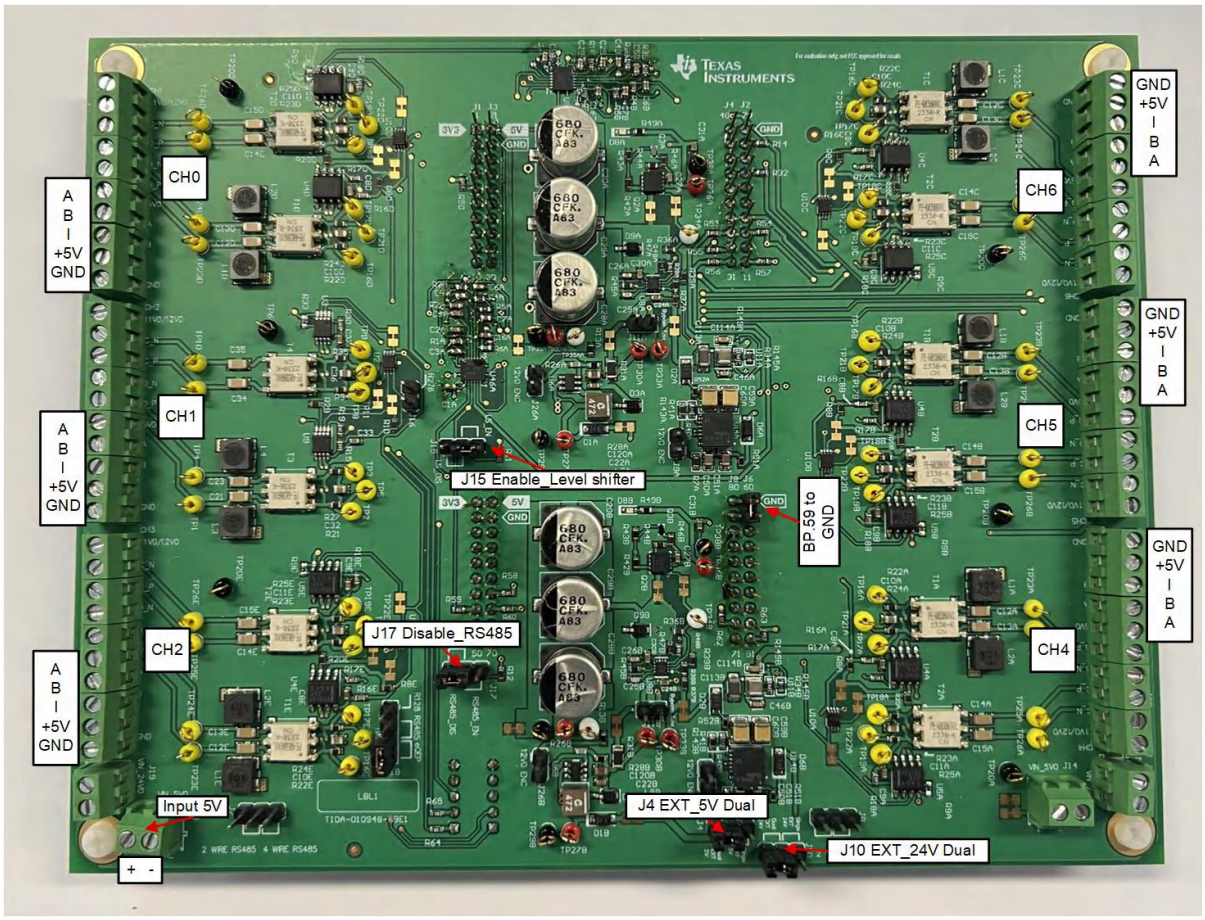


图 4-12. 位置板设置

4.2 软件要求

为了验证此参考设计，我们使用 AM243x ALV 和 ALX 封装以及适用于 AM243x MCU 的 `motor_control_sdk_am243x_09_02_00_09` 软件开发套件开发了一款 TI 内部测试软件。该软件不可供公众使用。有关 AM243x 软件支持的信息，另请参阅 MCU-PLUS-SDK-AM243X 软件开发套件 (SDK) 和基于 Arm 的微控制器论坛 - 基于 Arm 的微控制器 - TI E2E 支持论坛。

表 4-1. 主要软件配置

子系统	规格	值
SoC EPWM 和 ICSSG_PRU PWM	频率	16kHz
	同步或相移模式	同步模式
	计数模式	向上/向下计数
	死区	200ns
电流反馈 – ICSS SDFM	正常电流 OSR	OSR 64
	采样模式	连续采样
	单或双更新功能	是
位置反馈 – PRU EQEP 和 SoC QEP	QEP 的最大通道数	6
控制算法 – FOC 环路	周期时间	62.5µs 或 31.25µs 作为双更新的选项
工业通信 – SORTE_G	周期时间	62.5µs
	实时控制	确定性网络时间

4.3 测试设置和结果

4.3.1 电流反馈 – SDFM

图 4-13 显示了调制器 AMC1035 侧的 SDFM 时钟和数据。蓝色曲线的通道 1 是控制板 ICSSG0_IEP1_SYNC0 在时钟缓冲器 LMK1C1104 之后生成的 20MHz 时钟。红色曲线的通道 2 是 AMC1035 生成的 SD 数据。

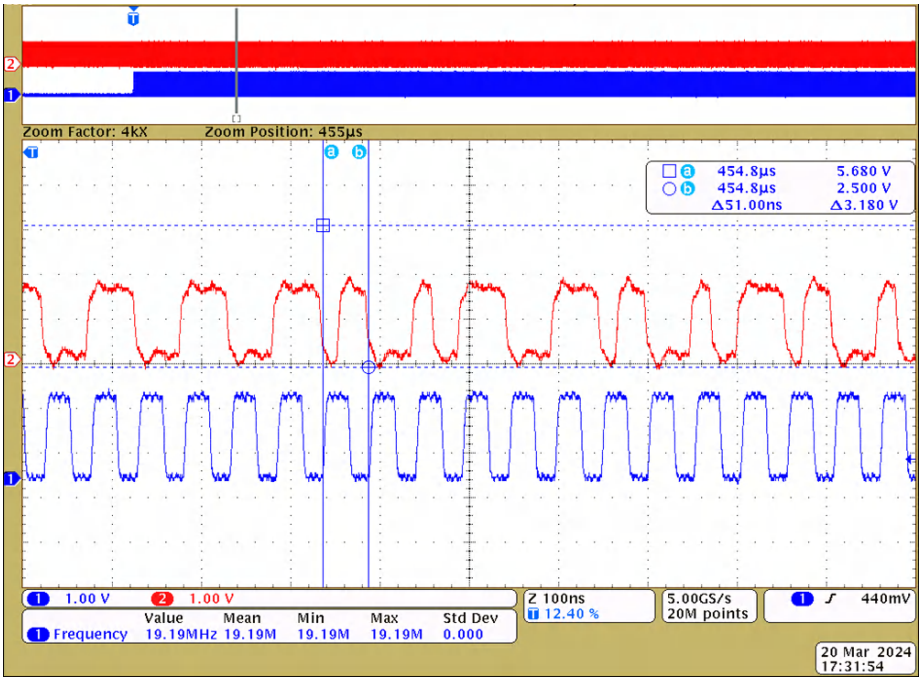


图 4-13. SDM 时钟和数据线信号

图 4-14 和 图 4-15 显示了测试设置以及使用带开环控制的电流探头测试的两相电流。

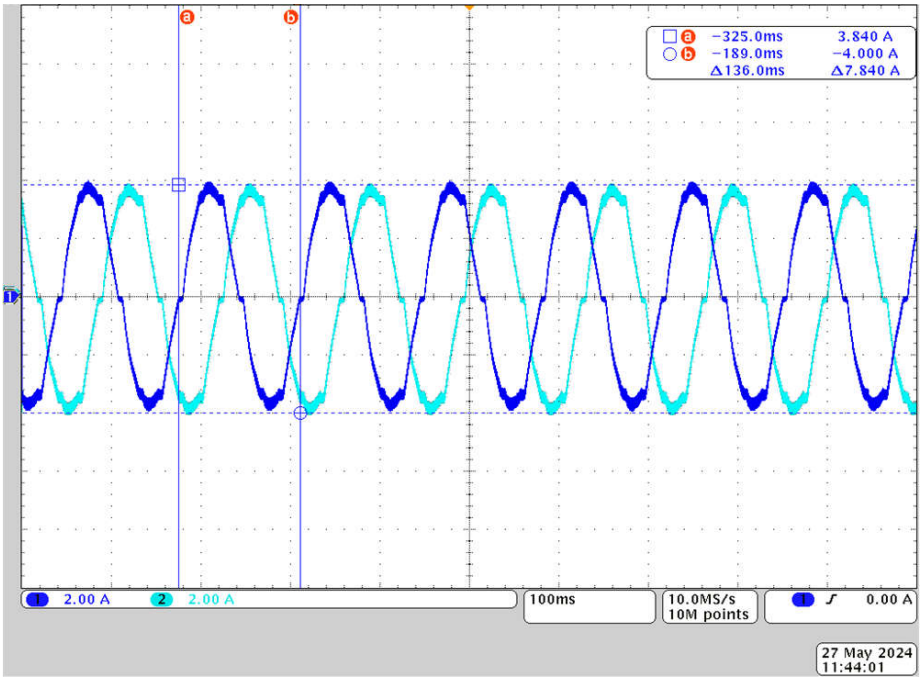


图 4-14. A 相和 B 相电流

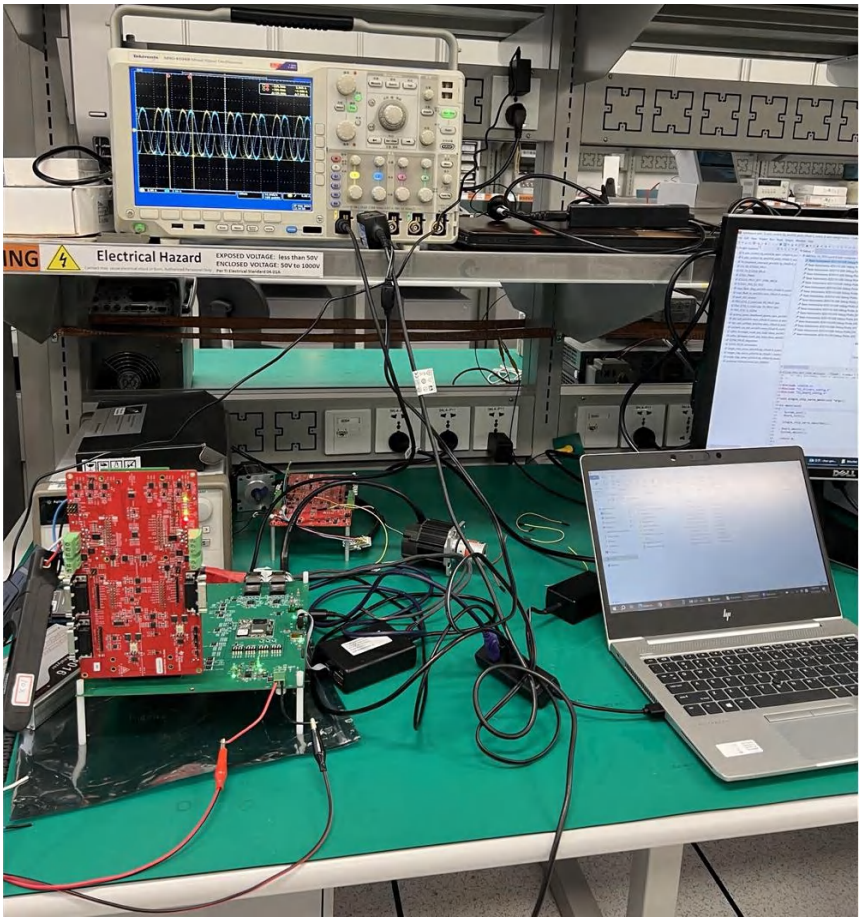


图 4-15. SDFM 测试设置

4.3.2 工业以太网 (SORTE_G) 和 PWM 接口之间的时间同步

图 4-16 显示了由位置板（SORTE_G 器件）上的 ICSSG1_IEP0_SYNCOUT0、SoC EPWM 信号和 ICSSG_PRU_PWM 信号生成的同步脉冲。通道 0 是位置板上的同步脉冲，通道 8 至 13 是控制板上的 EPWM0、1、2 信号，通道 14 至 15 是控制板上的 ICSSG_PRU_PWM 信号。图 2-3 显示了时间同步流程。SORTE_G 生成具有预定义周期时间的循环脉冲，以通过 TSR 模块与 EPWM0 同步。用于 ICSSG_PRU_PWM 的 IEP 计时器由 EPWM0 SYNC0 复位。

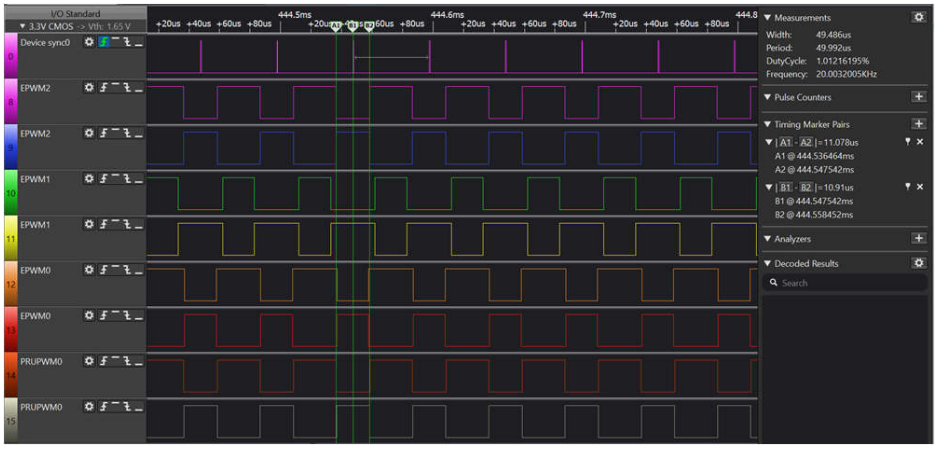


图 4-16. 工业以太网 (SORTE_G) 和 PWM 接口之间的时间同步

4.3.3 FOC 环路验证

4.3.3.1 FOC 环路时序

图 4-17 展示了使用 PWM 和 SDFM 接口的 FOC 环路（双更新时为 20kHz）的时序测试数据。

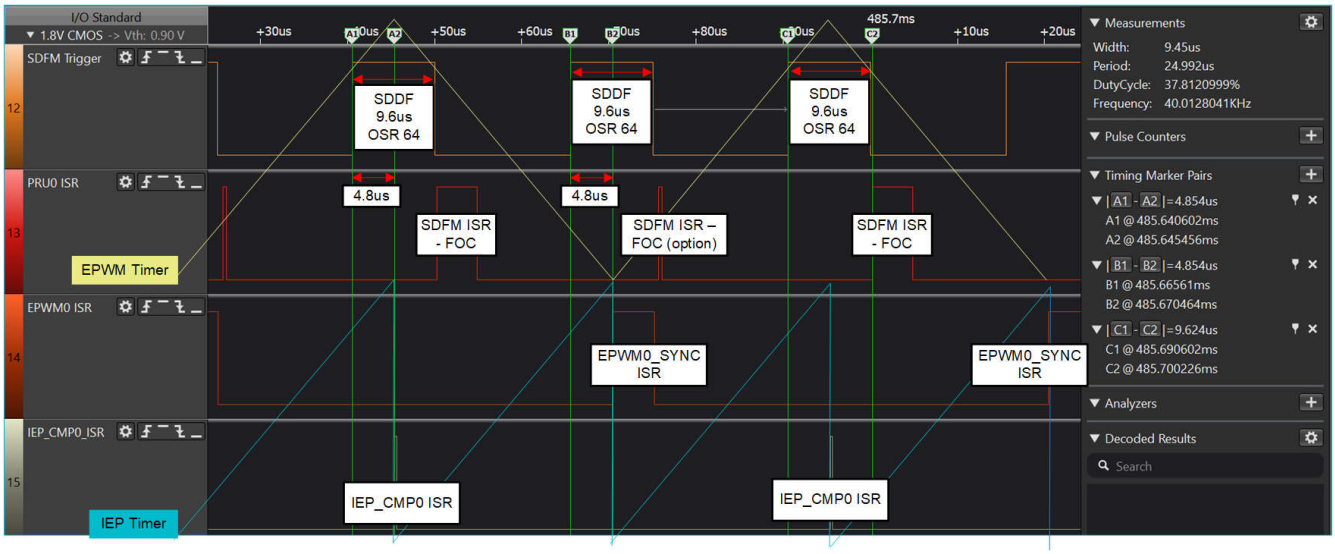


图 4-17. FOC 时序验证

4.3.3.2 FOC 环路处理时间验证

图 4-18 显示了具有闭合速度环路的 FOC 环路的处理时间的测试数据。一个 FOC 环路计算时间约为 1.056µs，6 轴闭合速度环路约为 7.584µs，周期时间为 62.5µs。

图 4-18. 6 轴 FOC 环路处理时间

4.3.4 使用 PI 控制器进行的闭环控制验证

对于电流和速度环路，PI 控制器都通过 `DCL_runPIParallel()` 函数实现。PI 常量可以通过 `init_pids()` 函数调整。

图 4-19 显示了闭环 FOC 方框图的验证。在闭合速度环路期间，I_Q 基准是速度 PI 控制器的输出。但为了验证 PI 控制器，`gIqRef` 作为常量目标给出，而 `parkIqMeasured` 是 Park 和 Clarke 变换之后的电机相电流值，用作 PI 调整后的反馈值。现在，记录 `parkIqMeasured` 以查看变量是否会跟随目标值 `gIqRef`。

图 4-20 显示了使用 PI 控制器的电机 A 相反馈波形和电流环路的阶跃响应，其中包括以下参数：

- `gPiIq.Kp` = 0.245
- `gPiIq.Ki` = 0.09
- `gPiIq.Umax` = 0.2
- `gPiIq.Umin` = 0.2

在此，根据 DRV8316 功能，电流环路 PI 控制输出限制为 0.2。为了进行验证，将 `gIqRef` = 0.3 作为闭合电流环路的电流目标。通过以下行实现包含该函数的 PI 控制器：

- *parkIqOut = DCL_runPIParallel(&gPIlq, gIqRef, parkIqMeasured);*

结果表明，相电流可很好地由 PI 控制（峰值约为 0.3A，等于目标 I_Q 基准），电流环路阶跃响应时间约为 90µs（控制频率为 32kHz）。因此，电流环路带宽约为 3.54kHz 并采用 方程式 1：

$$\text{Bandwidth} = \frac{1}{\pi \times t_r} \quad (1)$$

其中

- t_r = 响应时间

电机使用 LVSERVO 以及以下参数：

- 相间电阻 = 0.72 Ω
- 相间电感 = 0.4mH
- 电气时间常数 = 0.56
- 反电动势 (V_{peak} / K_{rpm}) = 4.64

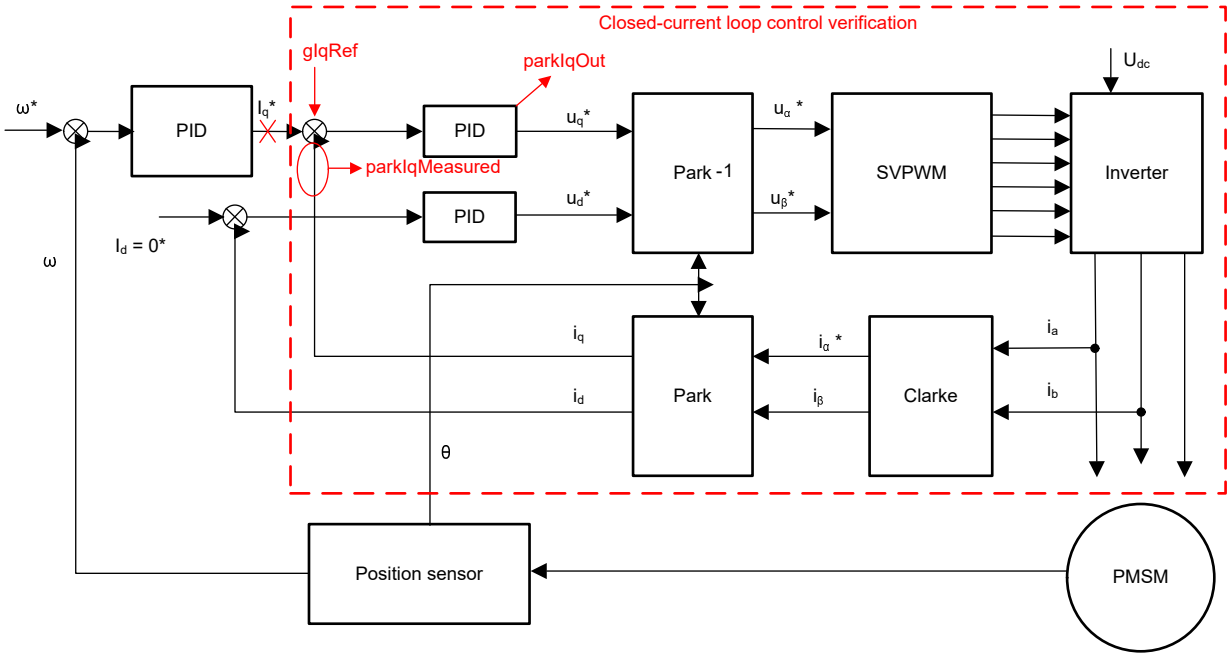


图 4-19. 具有 PI 控制器的闭合电流环路验证图

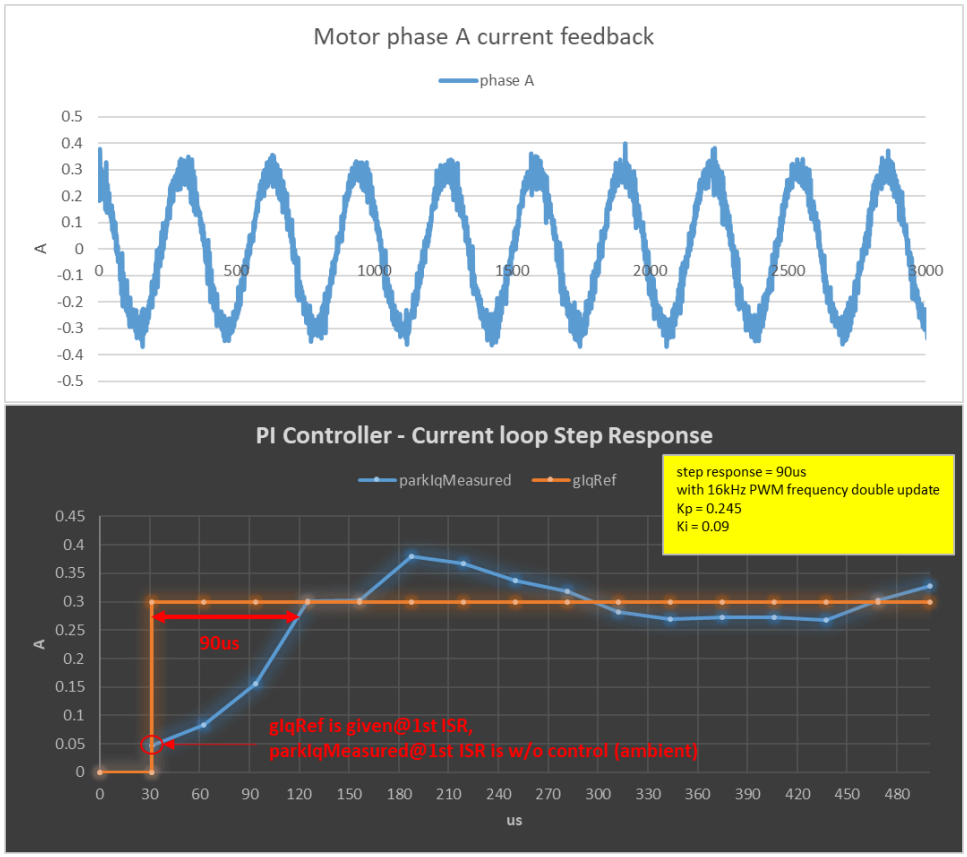


图 4-20. 电机相电流反馈和 PI 控制器阶跃时间响应

5 设计和文档支持

5.1 设计文件

5.1.1 原理图

要下载原理图，请参阅 TIDA-010948 中的设计文件。

要下载 BLDC BoosterPack 原理图，请参阅 BP-AM2BLDCSERVO 设计包中的设计文件。

要下载 AM243x LaunchPad 原理图，请参阅 LP-AM243 设计包中的设计文件。

5.1.2 BOM

要下载物料清单 (BOM)，请参阅 TIDA-010948 中的设计文件。

要下载 BLDC BoosterPack BOM，请参阅 BP-AM2BLDCSERVO 设计包中的设计文件。

要下载 AM243x LaunchPad BOM，请参阅 LP-AM243 设计包中的设计文件。

5.1.3 板层图

要下载板层图，请参阅 TIDA-010948 中的设计文件。

要下载 BLDC BoosterPack 板层图，请参阅 BP-AM2BLDCSERVO 设计包中的设计文件。

要下载 AM243x LaunchPad 板层图，请参阅 LP-AM243 设计包中的设计文件。

5.1.4 Altium 工程

要下载 Altium Designer® 工程文件，请参阅 TIDA-010948 中的设计文件。

要下载 BLDC BoosterPack Altium 工程文件，请参阅 BP-AM2BLDCSERVO 设计包中的设计文件。

要下载 AM243x LaunchPad Altium 工程文件，请参阅 LP-AM243 设计包中的设计文件。

5.1.5 Gerber 文件

要下载 Gerber 文件，请参阅 TIDA-010948 的设计文件。

要下载 BLDC BoosterPack Gerber 文件，请参阅 BP-AM2BLDCSERVO 设计包中的设计文件。

要下载 AM243x LaunchPad Gerber 文件，请参阅 LP-AM243 设计包中的设计文件。

5.1.6 装配图

要下载装配图，请参阅 TIDA-010948 中的设计文件。

要下载 BLDC BoosterPack 装配图，请参阅 BP-AM2BLDCSERVO 设计包中的设计文件。

要下载 AM243x LaunchPad 装配图，请参阅 LP-AM243 设计包中的设计文件。

5.2 工具与软件

工具

CCSTTUDIO	Code Composer Studio™ 集成开发环境 (IDE)：下载适用于 Microsoft® Windows® 或 Linux® 的 CCS 20.0.0 版
ARM-CGT-CLANG	Arm® 代码生成工具 — 编译器：下载适用于 Microsoft Windows 或 Linux 的 TI ARM CLANG 3.2.0.LTS
SYSCONFIG	SysConfig 独立桌面版本：下载适用于 Microsoft Windows 或 Linux 的 SysConfig 1.22.0

软件

AM243x 电机控制 SDK	电机控制 SDK Microsoft Windows 安装程序
AM243x 工业通信 SDK	工业通信 SDK Microsoft Windows 安装程序
AM243x MCU+ SDK	MCU PLUS SDK Microsoft Windows 安装程序

5.3 文档支持

1. 德州仪器 (TI)，*AM64x/AM243x 处理器器件技术参考手册*
2. 德州仪器 (TI)，*AM2x BLDC 伺服电机 BoosterPack (BPAM2BLDCSERVO) EVM 用户指南*
3. 德州仪器 (TI)，*TIDEP-01032 通过 EtherCAT® 连接的单芯片双伺服电机驱动参考设计*
4. 德州仪器 (TI)，*AM243x 电机控制 SDK 09.02.00 下的 TIDEP-01032 示例*

5.4 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。

5.5 商标

Sitara™, E2E™, BoosterPack™, LaunchPad™, Code Composer Studio™, and TI E2E™ are trademarks of Texas Instruments.
EtherCAT® is a registered trademark of Beckhoff Automation GmbH.
PROFINET® is a registered trademark of PROFIBUS Nutzerorganisation e.V..
Arm® is a registered trademark of Arm Limited.
Python® is a registered trademark of Python Software Foundation.
Altium Designer® is a registered trademark of Altium LLC.
Microsoft® and Windows® are registered trademarks of Microsoft Corporation.
Linux® is a registered trademark of Linus Torvalds.
所有商标均为其各自所有者的财产。

6 作者简介

CHEN GAO 是德州仪器 (TI) 工业系统电机驱动器团队的系统工程师，负责为工业驱动器和机器人指定并开发参考设计

SABARI KANNAN MUTHALAGU 是德州仪器 (TI) 工业系统机器人团队的系统工程师，负责为机器人指定并开发参考设计

THOMAS LEYRER 是德州仪器 (TI) 工业系统工厂自动化控制团队的系统架构师和杰出技术研究员，负责为工业驱动器、机器人和工厂自动化指定并开发参考设计

致谢：

作者感谢 **Pratheesh Gangadhar TK**、**Dhaval Khandla**、**Achala Ram** 和 **Manoj Koppolu** 为支持 TIDA-010948 参考设计的软件开发所做出的出色贡献

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