Design Guide: TIDEP-01032

通过 EtherCAT® 连接的单芯片双伺服电机驱动器参考设计

说明

此参考设计展示了 AM243x 器件能支持完全集成的实时伺服电机驱动控制和工业通信路径，该路径首先接收 EtherCAT® CiA402 目标的速度命令，然后对连接的两个电机执行闭环 FOC 速度控制，最后将实际速度值传输回 EtherCAT PLC。

资源

TIDEP-01032	设计文件夹
AM243x MCU+ SDK	工具文件夹
AM243x LaunchPad™	工具文件夹
BLDC BP	工具文件夹
AM243x Academy	培训材料

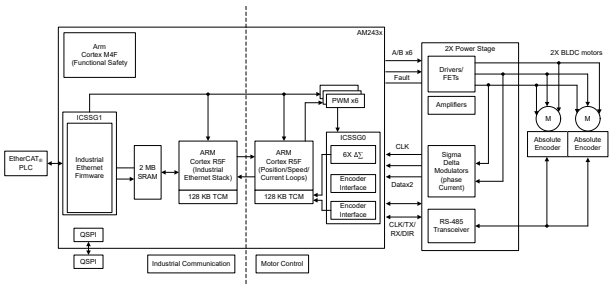
请咨询我司 TI E2E™ 支持专家

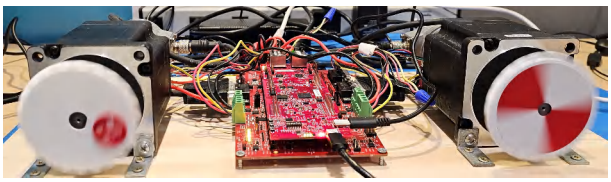
特性

- 支持用于电机速度控制的 EtherCAT CiA402 器件配置文件
- 单芯片双伺服电机控制
- BOOST-XL TI BoosterPack™ 插件模块设计 — 80 个与 AM2x LaunchPad™ 开发套件兼容的数字和模拟 I/O
- 两个轴通过 DRV8316R 24V、8A 单片栅极驱动和放大器桥实现三相 BLDC 电机驱动
- 两个轴（6 通道）通过 AMC1035D Σ-Δ 调制器和 INA241A 电流检测路径实现三相电流反馈
- 两个轴实现符合多种工业编码器标准的 RS-485 绝对编码器反馈

应用

- 伺服驱动器通信模块
- 伺服驱动器控制模块
- 伺服驱动器位置反馈
- 伺服驱动器位置传感器
- 伺服驱动器功率级模块





1 系统说明

该参考设计展示了 AM243x 器件能支持全面的实时伺服电机控制和工业通信路径，此路径首先接收 EtherCAT CiA402 目标的速度命令，然后对连接的两个电机执行闭环 FOC 速度控制，最后将实际速度值传输回给 EtherCAT PLC。

1.1 术语

PLC	可编程逻辑控制器
FOC	磁场定向控制
EtherCAT	用于控制自动化技术的以太网
CiA402	用于驱动和运动控制的 EtherCAT 配置文件
RPM	每分钟转数
EnDAT 2.2	面向增量式和绝对位置编码器的数字双向接口标准
ICSS	工业通信子系统
PRU	可编程实时单元
LP	LaunchPad™
SDFM	Σ-Δ 滤波器模块
SDDF	Σ-Δ 抽取滤波器
IPC	处理器间通信
IEP	工业以太网外设
CMP	事件比较器
ISR	中断服务例程
PWM	脉宽调制
EPWM	增强型脉宽调制器

1.2 主要系统规格

1. 支持用于伺服电机速度控制的 EtherCAT CiA402 器件配置文件
2. 单芯片双伺服电机控制
3. 用于电流和速度控制的 50kHz FOC 环路
4. 两轴（6 通道）提供三相 Σ-Δ 调制电流反馈
5. 两轴提供 EnDat 2.2 编码绝对位置反馈

2 系统概述

图 2-1 展示了 TIDEP-01032 系统的设置。

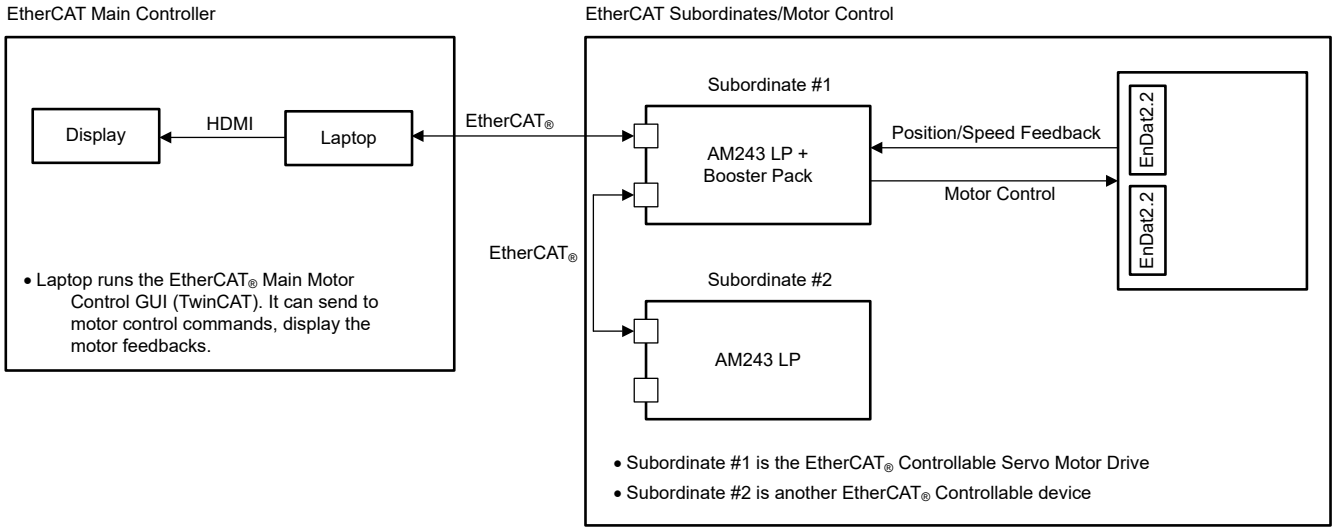


图 2-1. 系统设置

2.1 方框图

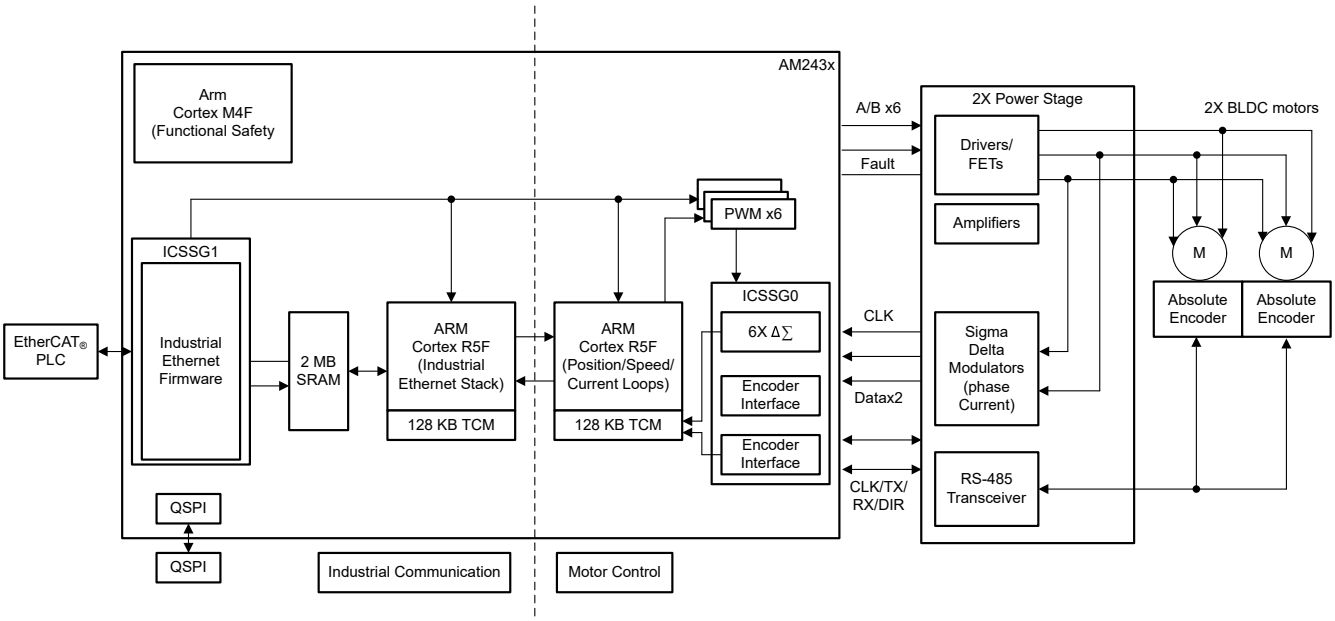


图 2-2. TIDEP-01032 方框图

2.2 设计注意事项

单芯片双轴伺服电机驱动方案是围绕一个核心实时路径来构建的，该路径包括以下部分：

- ICSSG1 — EtherCAT 客户端控制器固件
- ICSSG0 — SDDF 和 EnDAT 2.2 解码功能
  - Σ-Δ 滤波固件，针对两个直接连接的电机的相电流反馈，在 PRU0 中的 RTU 和 PRU 内核之间实现*负载共享*
  - EnDat2.2 解码固件，针对两个直接连接的绝对编码器的角度、位置和速度反馈，在 PRU1 中的 RTU 和 PRU 内核之间实现*负载共享*
- R5FSS1_0 — 使用 FreeRTOS 实现 CiA402 的 EtherCAT 客户端栈

- R5FSS0_0 和 R5FSS0_1 — 两个独立的闭环 FOC，能够通过绝对编码器对两个直接连接电机进行电流、速度或位置闭环控制
- MCU+ SDK 中的 IPC Notify 可提供低延迟内核间同步和通信
- EPWM — 六个通道的增强型 PWM 外设，用于根据两个 FOC 环路的输出生成波形

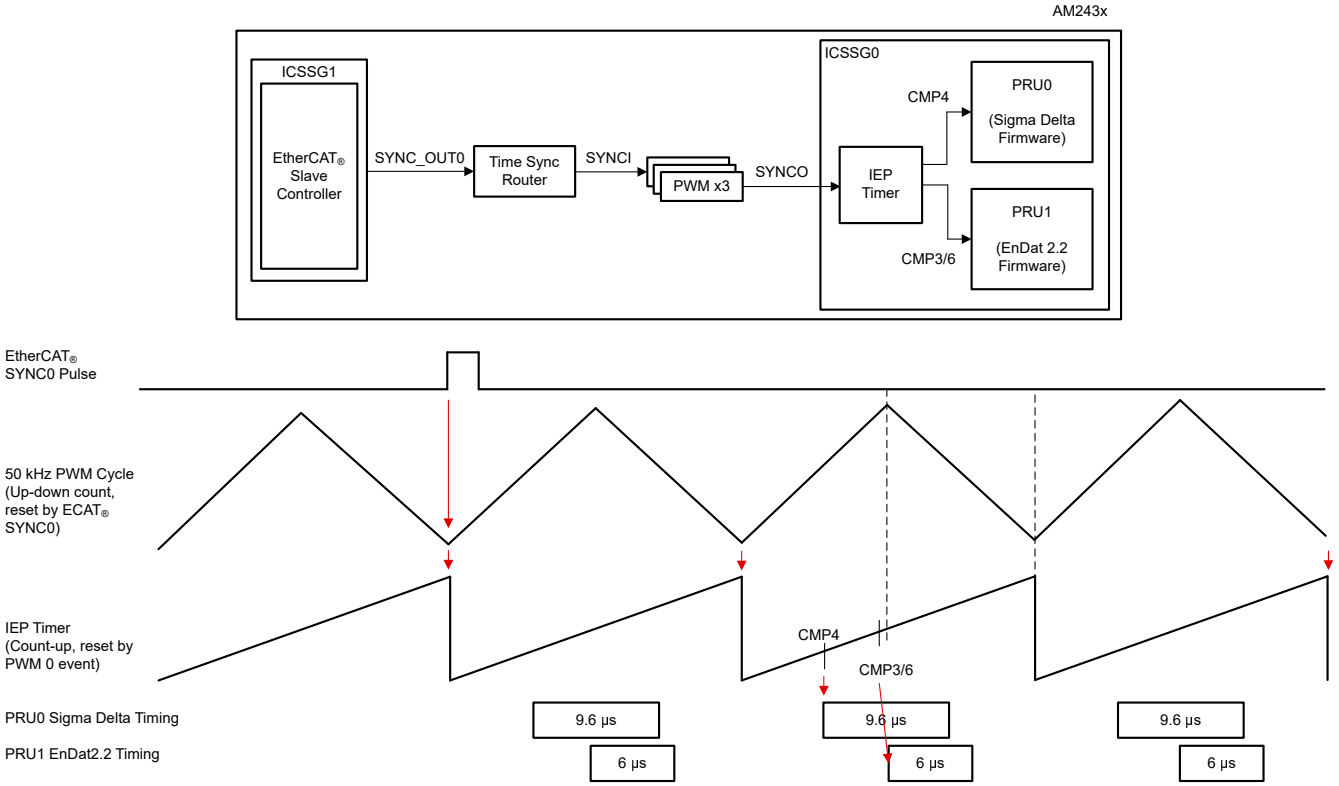


图 2-3. 触发 FOC 时序 — 50kHz

2.3 主要产品

2.3.1 EnDAT 2.2 接口

表 2-1 显示了 EnDAT 2.2 信号参数。

表 2-1. EnDAT 2.2 信号（两个 4 线编码器）

AM243x LP（引脚编号）	BP 连接器	BLDC BP	信号名称
GPIO1_78(C16)	J8.73	VSENSOR1_SW_EN	编码器 1 启用
PRG0_PRU1_GPO0(L5)	J2.11	ENCODER_CLK1	编码器 1 时钟
PRG0_PRU1_GPO2 (M2)	J7.68	ENCODER_DATA_TX_EN1	编码器 1 TX 启用
PRG0_PRU1_GPO1(J2)	J7.67	ENCODER_DATA_TX1	Encoder1 TX
PRG0_PRU1_GPO13(T4)	J8.71	ENCODER_DATA_RX1	Encoder1 RX
GPIO1_77(B17)	J8.74	VSENSOR2_SW_EN	编码器 2 启用
PRG0_PRU1_GPO6(F5)	J7.69	ENCODER_CLK2	编码器 2 时钟
PRG0_PRU1_GPO8(F4)	J6.57	ENCODER_DATA_TX_EN2	编码器 2 TX 启用
PRG0_PRU1_GPO12(P2)	J8.72	ENCODER_DATA_TX2	Encoder2 TX
PRG0_PRU1_GPO11(P1)	J7.70	ENCODER_DATA_RX2	Encoder2 RX

EnDat 2.2 中断：

- hwiPrms.intNum = ICSSG_PRU_ENDAT_INT_NUM | ICSSG_PRU_ENDAT_INT_NUM+2;
- hwiPrms.callback = &pruEncoderIrqHandler | &pruEncoderIrqHandler2;

- 电机控制环路 (FOC) 每个电机一个内核

EnDat 2.2 输入数据缓冲器：

- R5F_0_0 TCMB 中的 gEndatChInfo（位于 .gEncChData 中）

ICSSG 引脚 MUX：

- 模式 (ICSSG_GPCFG0_REG[29-26]: PR1_PRU0_GP_MUX_SEL = 1h)
- ICSSG_SA_MX_REG[7] G_MUX_EN = 0

2.3.2 SDFM 接口

图 2-4 展示了时钟源分配，表 2-2 显示了 SDFM 信号。

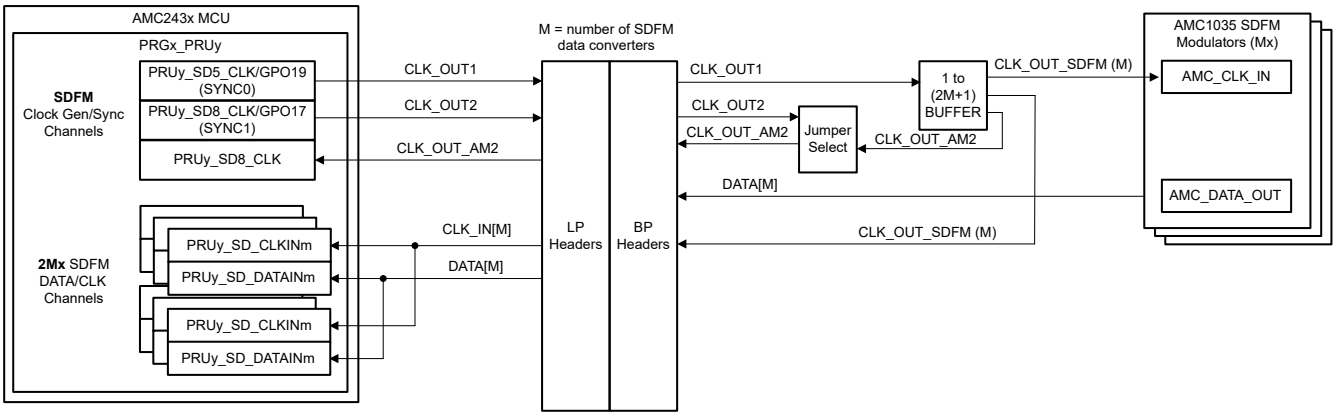


图 2-4. 时钟源分配

表 2-2. SDFM 信号

AM243x LP（引脚编号）	BP 连接器	BLDC BP	信号名称
PRG0_PRU0_GPI1(J4)	J4.32	SDFM 电流高电平 A1	SDFM 电流高电平 A1
PRG0_PRU0_GPI3(H1)	J2.19	SDFM 电流高电平 B1	SDFM 电流高电平 B1
PRG0_PRU0_GPI5(F2)	J2.13	SDFM 电流高电平 C1	SDFM 电流高电平 C1
PRG0_PRU0_GPI7(E2)	J5.44	SDFM 电流高电平 A2	SDFM 电流高电平 A2
PRG0_PRU0_GPI8(H5)	J2.15	SDFM 电流高电平 B2	SDFM 电流高电平 B2
PRG0_PRU0_GPO11(L1)	J2.12	SDFM 电流高电平 C2	SDFM 电流高电平 C2

SDFM 时钟：

- PRG0_PRU0_GPO19(G2) → SYNC0 - SDFM CLOCK_OUT1(J5.45) → SDFM_CLOCK_SOURCE[1|2] (TP31)
  - AMC_CLKIN_A1、AMC_CLKIN_B1、AMC_CLKIN_C1、PR0_PRU0_SD0_CLK(J4.33)、PR0_PRU0_SD1_CLK(J4.31)、PR0_PRU0_SD2_CLK(J2.17)
  - AMC_CLKIN_A2、AMC_CLKIN_B2、AMC_CLKIN_C2、PR0_PRU0_SD3_CLK(J5.48)、PR0_PRU0_SD6_CLK(J1.5)、PR0_PRU0_SD7_CLK(J2.14)

SDFM 中断：

- hwiPrms.intNum = ICSSG_PRU_SDDF_INT_NUM | ICSSG_RTU_SDDF_INT_NUM;
- hwiPrms.callback = &pruSddfIrqHandler | &rtuSddfIrqHandler;

SDFM 输入数据缓冲器：

- gSddfChSamps[0-2]（在 .gSddfChSampsRaw 中）位于电机 1 的 R5F_0_0 的 TCMB 中
- gSddfChSamps[3-5]（在 .gSddfChSampsRaw 中）位于电机 2 的 R5F_0_1 的 TCMB 中

2.3.3 EPWM 接口

表 2-3 和表 2-4 分别显示了 *EPWM0-2 信号电机 1* 和 *EPWM3-5 信号电机 2* 数据。

表 2-3. EPWM0-2 信号电机 1

AM243x LP（引脚编号）	BP 连接器	BLDC BP	信号名称
GPIO1_64(B16)	J5.49	nPWM_EN_M1	DRV1 使能
GPMC0_AD8(U18)	J4.36	DRV1 EPWM 高电平 C	DRV1 PWM 高电平 C
GPMC0_AD9(U20)	J4.35	DRV1 EPWM 低电平 C	DRV1 PWM 低电平 C
GPMC0_AD5(T20)	J4.38	DRV1 EPWM 高电平 B	DRV1 PWM 高电平 B
GPMC0_AD6(T18)	J4.37	DRV1 EPWM 低电平 B	DRV1 PWM 低电平 B
GPMC0_AD3(V21)	J4.40	DRV1 EPWM 高电平 A	DRV1 PWM 高电平 A
GPMC0_AD4(U21)	J4.39	DRV1 EPWM 低电平 A	DRV1 PWM 低电平 A

表 2-4. EPWM3-5 信号电机 2

AM243x LP（引脚编号）	BP 连接器	BLDC BP	信号名称
GPIO1_65(B15)	J5.50	nPWM_EN_M2	DRV2 使能
FSI_TX0_CLK (P21)	J8.79	DRV2 EPWM 高电平 C	DRV2 PWM 高电平 C
FSI_TX0_D0(Y18)	J8.80	DRV2 EPWM 低电平 C	DRV2 PWM 低电平 C
TEST_LED3_RED(D1)	J8.75	DRV2 EPWM 高电平 B	DRV2 PWM 高电平 B
TEST_LED4_GREEN(F3)	J8.76	DRV2 EPWM 低电平 B	DRV2 PWM 低电平 B
TEST_LED1_GREEN(U19)	J8.77	DRV2 EPWM 高电平 A	DRV2 PWM 高电平 A
FSI_RX0_D1(V20)	J8.78	DRV2 EPWM 低电平 A	DRV2 PWM 低电平 A

EPWM 设置 (**init_pwms**)：

- 配置 SYNCI、SYNCO 映射以将三个 PWM 组绑定在一起
- 由时间同步路由器 38 提供 PWM0 SYNC 信号
  - CSL_REG32_WR(CSL_TIMESYNC_EVENT_INTROUTER0_CFG_BASE + ((38 × 4) + 4), (0x10000 | 29));
- 时间同步路由器输入 29 (ICSSG1 IEP0 SYNC0) → 时间同步路由器输出 38
  - CSL_REG32_WR(CSL_CTRL_MMR0_CFG0_BASE + CSL_MAIN_CTRL_MMR_CFG0_EPWM0_CTRL, (2 << CSL_MAIN_CTRL_MMR_CFG0_EPWM0_CTRL_SYNCIN_SEL_SHIFT));
  - TIMESYNC_INTRTR0_IN_29：PRU_ICSSG1_PR1_EDC0_SYNC0_OUT_0（IEP0 同步事件 0）
  - timesync_event_introuter_out_38: epwm0_sync.input2
  - TI E2E：[常见问题解答] *AM64x：时间同步路由器有什么用？如何使用它？*
- 由时间同步路由器 39 提供 PWM3 SYNC 信号
  - CSL_REG32_WR(CSL_CTRL_MMR0_CFG0_BASE + CSL_MAIN_CTRL_MMR_CFG0_EPWM3_CTRL, (2 << CSL_MAIN_CTRL_MMR_CFG0_EPWM3_CTRL_SYNCIN_SEL_SHIFT));
- 时间同步路由器输入 29 (ICSSG1 IEP0 SYNC0) → 时间同步路由器输出 39
  - CSL_REG32_WR(CSL_TIMESYNC_EVENT_INTROUTER0_CFG_BASE + ((39 × 4) + 4), (0x10000 | 29));
  - TIMESYNC_INTRTR0_IN_29：PRU_ICSSG1_PR1_EDC0_SYNC0_OUT_0（IEP0 同步事件 0）
  - timesync_event_introuter_out_39: epwm3_sync.input2
- 由时间同步路由器 40 提供 PWM6 SYNC 信号
  - CSL_REG32_WR(CSL_CTRL_MMR0_CFG0_BASE + CSL_MAIN_CTRL_MMR_CFG0_EPWM6_CTRL, (2 << CSL_MAIN_CTRL_MMR_CFG0_EPWM6_CTRL_SYNCIN_SEL_SHIFT));

- 时间同步路由器输入 29 (ICSSG1 IEP0 SYNC0) → 时间同步路由器输出 40
  - CSL_REG32_WR(CSL_TIMESYNC_EVENT_INTROUTER0_CFG_BASE + ((40 * 4) + 4), (0x10000 | 29));
  - TIMESYNC_INTRTR0_IN_29：PRU_ICSSG1_PR1_EDC0_SYNC0_OUT_0（IEP0 同步事件 0）
  - timesync_event_introuter_out_40: epwm6_sync.input2
  - 为 EPWM0 强制 SW 同步。其他 PWM 通过硬件同步菊花链进行同步
    - Epwm_tbTriggerSwSync(gEpwm0BaseAddr);
      - HW_WR_FIELD16(((gEpwm0BaseAddr + PWMSS_EPWM_OFFSET) + PWMSS_EPWM_TBCTL), PWMSS_EPWM_TBCTL_SWFSYNC, (uint16_t)PWMSS_EPWM_TBCTL_SWFSYNC_FORCE_SYNC);
- 将 EPWM 设置为 50kHz：
  - appEpwmCfg.epwmOutFreq = gEpwmOutFreq;
  - App_epwmConfig(&appEpwmCfg, &epwm2PrdVal, &epwm2CmpAVal);

EPWM0 中断：

- hwiPrms.intNum = EPWM0_INTR;
- hwiPrms.callback = &App_epwmIntrISR;

EPWM0 输出数据：

- gEpwmPrdVal

2.3.4 ICSS-PRU IEP

以下参数适用于 IEP CMP 设置：

- 50kHz EPWM 周期时间
- 50kHz FOC 环路更新（在 EnDAT ISR 中）
- 从 EtherCAT 客户端 (ICSSG1) 到同步 EPWM 时钟的 SYNC_OUT0
- PRU_ICSSG0 IEP0 周期设置为 6000 (300000000/50000)
  - 它也是 EPWM 周期
  - PRU_ICSSG 基于 IEP0 的地址为 0x3002E000
- 设置 CMP4 以触发 Σ-Δ 编码电流反馈数据采样：
  - 每个 IEP 或 EPWM 周期一个 CMP4：10 µs（在 gTestSdfmPrms.firstSampTrigTime 中定义）
  - 在 initPruSddf 中设置
- 设置 CMP3 和 CMP6 以触发 EnDAT 2.2 编码位置反馈数据采样：
  - 每个 IEP 或 EPWM 周期一个 CMP3 和 CMP6：3000ns（在 endat_periodic_interface.cmp3 和 endat_periodic_interface.cmp6 中定义）
  - 在 endat_config_periodic_mode 中设置

2.3.5 EtherCAT CiA402 速度控制

目标速度（EtherCAT 控制器 → EtherCAT 从属节点 → AL243x LP）：

- gCurTargetVelocity [3]（在 .gEtherCatCia402 中），在非高速缓存片上 RAM 中
- 使用 GUI 通过 EtherCAT 控制器设置
- 发送到 EtherCAT 从属节点
- 通过 EC_SLV_APP_CSV 保存到 gCurTargetVelocity [3]
- 在 pruEncoderIrqHandler 或 pruEncoderIrqHandler2 中用于速度控制

实际速度（EtherCAT 控制器 ← EtherCAT 从属节点 ← AL243x LP）：

- gCurActualVelocity [3]（在 .gEtherCatCia402 中），在非高速缓存片上 RAM 中
- 由 pruEncoderIrqHandler 或 pruEncoderIrqHandler2 保存到 gCurTargetVelocity [3] 表示实际速度
- 通过 EC_SLV_APP_CSV 传输到 EtherCAT 控制器
- 在 GUI 中由 EtherCAT 控制器显示

3 系统设计

R5F_0_0 初始化

使用以下步骤初始化**电机 1 的 R5F_0_0** (single_chip_servo_remote_core_start)：

1. 设置 GPIO 引脚方向和初始值 (init_gpio_state)
2. 禁用 EPWM (enable_pwm_buffers)
3. 为电机 1 设置 EPWM 频率和中断 (init_pwms)
4. 为 EnDAT 2.2 电机 1 设置 ICSSG0 PRU1（通道 0，init_encoder）
   - 在 ICSSG_SA_MX_REG 寄存器中将 g_mux_en 配置为 1。HW_WR_REG32((CSL_PRU_ICSSG0_PR1_CFG_SLV_BASE+0x40), (0x80))
   - 寄存并启用 ICSSG EnDat PRU FW 中断
   - – 中断号： ICSSG_PRU_ENDAT_INT_NUM
     – 回调函数：pruEncoderIrqHandler
   - 设置 EnDat 2.2 参数
     – gEndat_multi_ch_mask = ENDAT_MULTI_CH0 | ENDAT_MULTI_CH2;
     – gEndat_is_multi_ch = CONFIG_ENDAT0_MODE & 1;
     – gEndat_is_load_share_mode = CONFIG_ENDAT0_MODE & 2;
   - 初始化 ICSSG0 PRU1 (endat_pruss_init)
   - 使用编码器驱动程序 API 初始化编码器 (endat_init)
   - 使用编码器驱动程序 API 配置编码器 (endat_config_multi_channel_mask)
   - 根据 ICSSG 频率配置延迟
   - 加载 EnDat 2.2 PRU FW 并运行到 ICSSG0 PRU1 (endat_pruss_load_run_fw)
   - 检查来自固件的初始化确认，超时为 5 秒 (endat_wait_initialization)
   - 将 2.2 编码器的默认频率设置为 16MHz (endat_init_clock)
   - 设置传播延迟，使 16MHz 在 300MHz PRU 下有效 (endat_handle_prop_delay(priv, 265))
   - 将 EnDat 2.2 FW 设置为周期性触发 (endat_config_periodic_trigger)
   - 配置 EnDat 2.2 FW 周期模式的参数 (endat_config_periodic_mode)
     – 通道 0 的 IEP0 CMP3 事件（从 PWM 周期开始算起 3000ns）
     – 通道 2 的 IEP0 CMP6 事件（从 PWM 周期开始算起 3000ns）
   - 开始接收 EnData 2.2 数据 (endat_handle_rx)
5. 为电机 1 的 SDFM 设置 ICSSG0 PRU0 (init_sddf)
   - 初始化 IEP0，配置 SYNC0 SD 时钟 (init_IEP0_SYNC)
   - 初始化 ICSSG0 PRU0 (initIcss)
   - 寄存并启用 ICSSG SDFM RTU FW 中断
     – 中断号： ICSSG_RTU_SDDF_INT_NUM
     – 回调函数：rtuSddfIrqHandler
   - 初始化 SDFM 的 RTU/PRU 内核 (initPruSddf)
     – RTU 示例基址：gTestSdfmPrms.samplesBaseAddress
     – PRU 示例基址：gTestSdfmPrms.samplesBaseAddress+0x80
   - 启动 IEP0 (start_IEP0)
   - 为 EPWM0 强制 SW 同步。其他 PWM 通过硬件同步菊花链进行同步 (EPWM_tbTriggerSwSync)
   - 禁用 EPWM 输出缓冲器 (enable_pwm_buffers)
6. 初始化 FOC 的参数 (init_pids)
7. 为电机 1 启用 EPWM 输出缓冲器 (enable_pwm_buffers(TRUE))

R5F_0_0 初始化

使用以下步骤初始化**电机 2 的 R5F_0_1** (single_chip_servo_remote_core_start)：

1. 为电机 2 设置 EPWM 频率和中断 (init_pwms)
2. 为电机 2 寄存并启用 ICSSG EnDat PRU FW 中断

- 中断号： ICSSG_PRU_ENDAT_INT_NUM+2
- 回调函数：pruEncoderIrqHandler2

3. 为电机 2 寄存并启用 ICSSG SDFM PRU FW 中断
   - 中断号： ICSSG_PRU_SDDF_INT_NUM
   - 回调函数：pruSddfIrqHandler
4. 初始化 FOC 的参数 (init_pids)

设置中断

按照以下说明设置**中断**和**处理程序**：

EPWM 中断 (50kHz)，ISR — App_epwmIntrISR（电机 1）或 App_epwmIntrISR2（电机 2）

- 清除 EPWM 中断。

SDFM 中断 (50kHz)，ISR — rtuSddfIrqHandler（电机 1）或 pruSddfIrqHandler（电机 2）

- 从样本 8192 到 16384，计算 SDFM 通道偏移（0 – 2 或 3 – 5）。当 PRECOMPUTE_LEVEL = NO_PRECOMPUTE 时，在 FOC 环路中使用 SDFM 通道偏移
- 在样本 16384 处，为 A 相、B 相和 C 相写入 EPWM，以将转子锁定为电气 0 并禁用 SDFM 中断
- 清除源上的中断

EnDAT 2.2 中断 (50kHz)，ISR — pruEncoderIrqHandler（适用于电机 1）

1. 清除源上的中断
2. 对于样本 0 – 8192，不执行任何操作
3. 对于样本 8193 – 16383：
   - 计算机械和电角偏移 (localEnDatGetSingleMulti)
4. 对于样本 16384：
   - 找出机械和电角偏移量的平均值
   - 关闭所有相
   - 保存机械和电角度偏移
5. 对于 16384 之后的样本：
   - 启动 FOC 环路并解锁转子
   - 从编码器获取最新的机械 θ 和多转位置 (localEnDatGetSingleMulti)
   - 使用计算得出的相对于电气 0、4 极对的偏移
   - 运行 FOC 环路以计算空间矢量
   - 写入接下来的 CMPA 值。交换 cmp0 和 cmp2，因为硬件将 EPWM0 连接到 C 相，将 EPWM2 连接到 A 相
   - EPWM0 实际上使用 EHRPWM2；EPWM1 使用 EHRPWM1，EPWM2 使用 EHRPWM0
   - 有关详细信息，请参阅 single_chip_servo_am243x-lp_r5fss0-0_nortos_ti-arm-clang 中 example_syscfg 内的 EPWM 设置

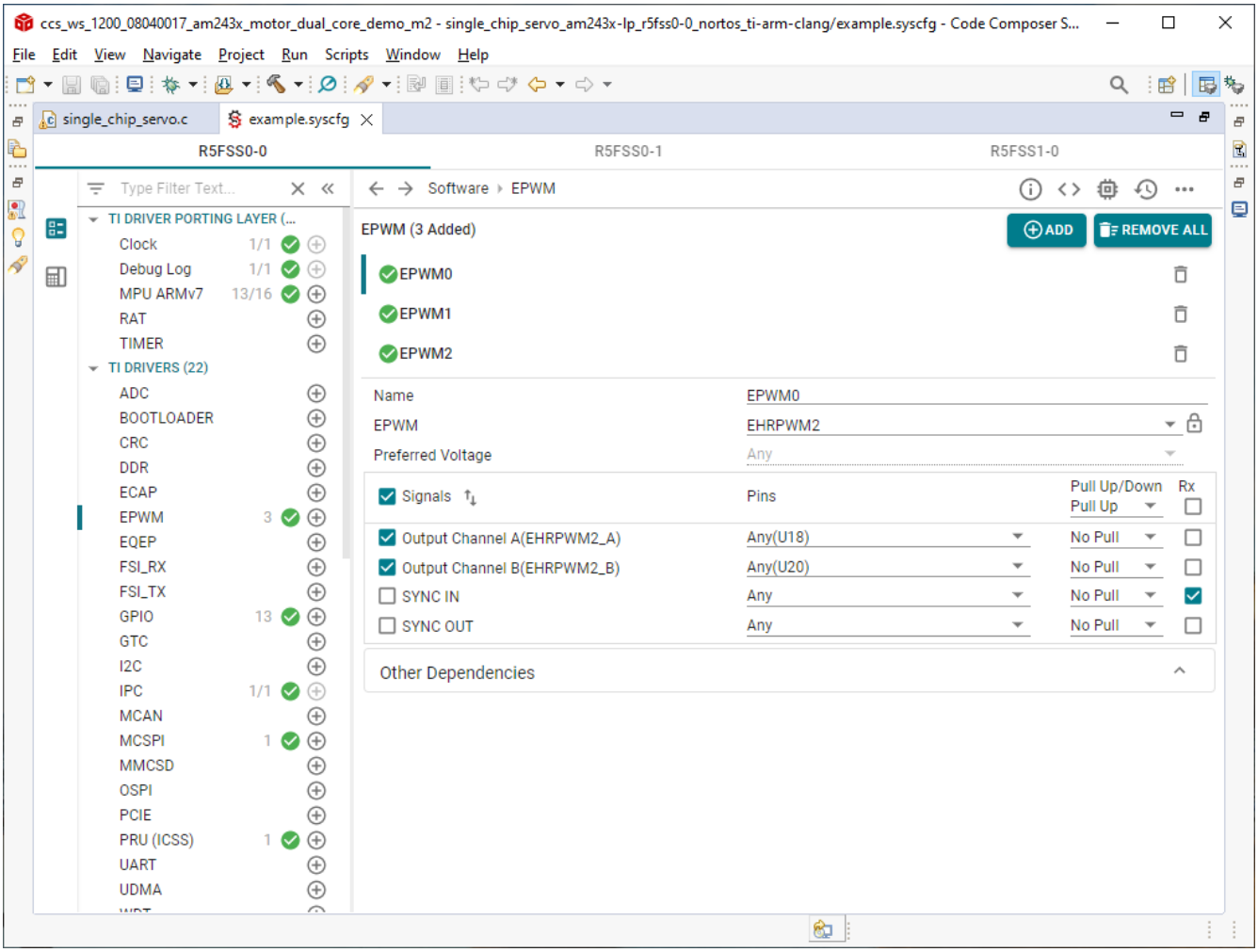


图 3-1. 电机 1 的 EPWM 设置

EnDAT 2.2 中断 (50kHz)，ISR — pruEncoderIrqHandler（适用于电机 2）

- 清除源上的中断
- 对于样本 0 – 8192，不执行任何操作
- 对于样本 8193 – 16383：
  - 计算机械和电角偏移 (localEnDatGetSingleMulti)
- 对于样本 16384：
  - 找出机械和电角偏移量的平均值
  - 关闭所有相
  - 保存机械和电角度偏移
- 对于 16384 之后的样本：
  - 启动 FOC 环路并解锁转子
  - 从编码器获取最新的机械 θ 和多圈位置 (localEnDatGetSingleMulti)
  - 使用计算得出的相对于电气 0、4 极对的偏移
  - 运行 FOC 环路以计算空间矢量
  - 写入接下来的 CMPA 值。交换 cmp0 和 cmp2，因为硬件将 EPWM0 连接到 C 相，将 EPWM5 连接到 A 相
  - EPWM0 实际上使用 EHRPWM5；EPWM1 使用 EHRPWM4，EPWM2 使用 EHRPWM3
  - 有关详细信息、请参阅 single_chip_servo_am243x-lp_r5fss0-1_nortos_ti-arm-clang 中 example_syscfg 内的 EPWM 设置

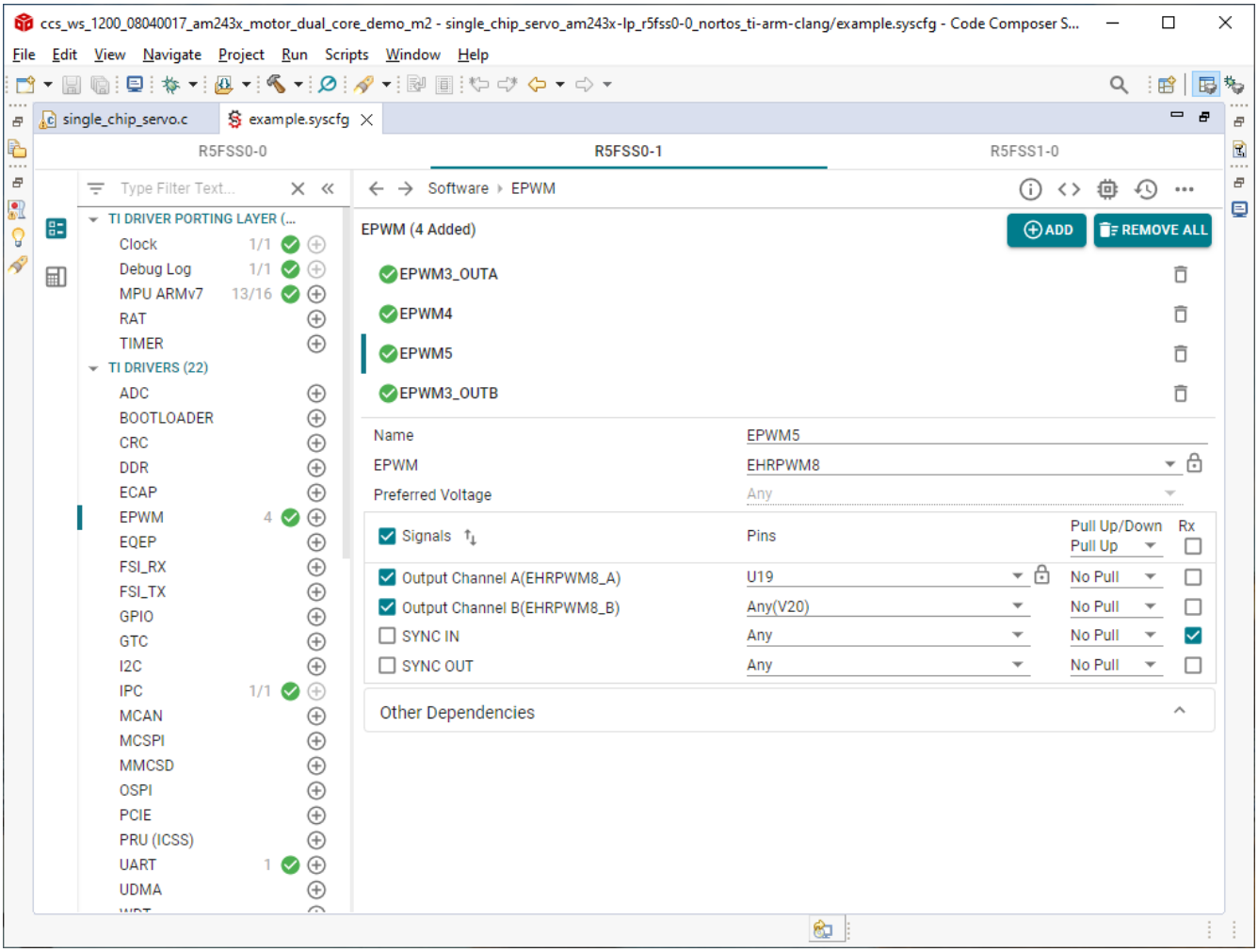


图 3-2. 电机 2 的 EPWM 设置

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：

- 一台安装了 TwinCAT 自动化软件的 Microsoft® Windows® 个人计算机
- 一块 LP-AM243 评估板 | 德州仪器 TI.com.cn
- 一个 BP-AM2BLDCSERVO — AM2x 无刷直流 (BLDC) 伺服电机 BoosterPack
- 两个 BLY342D-48V-3200 Anaheim Automation 三相无刷直流电机
- 两个 ROQ-437 EnDat2.2 编码器及电缆

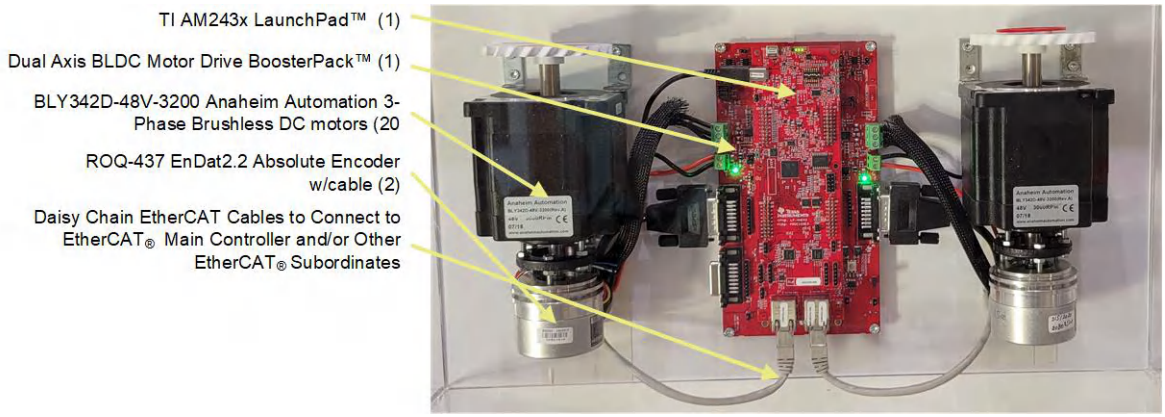


图 4-1. 系统硬件配置

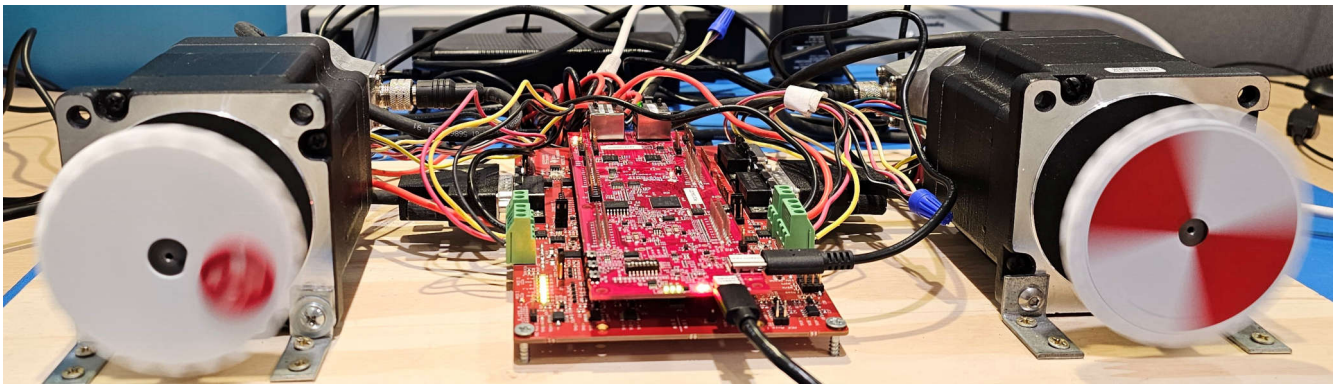


图 4-2. 双电机驱动系统设置

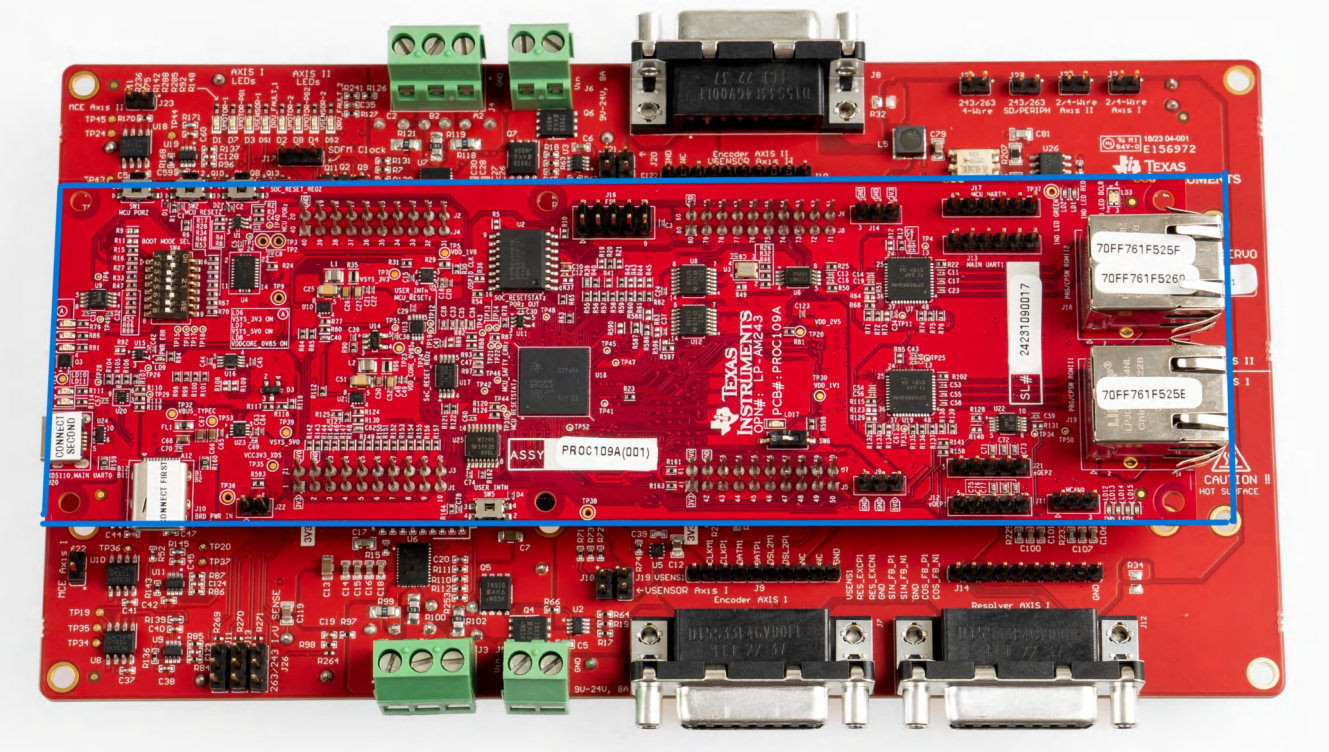


图 4-3. AM243x LP Rev A 和 BLDC BP E2

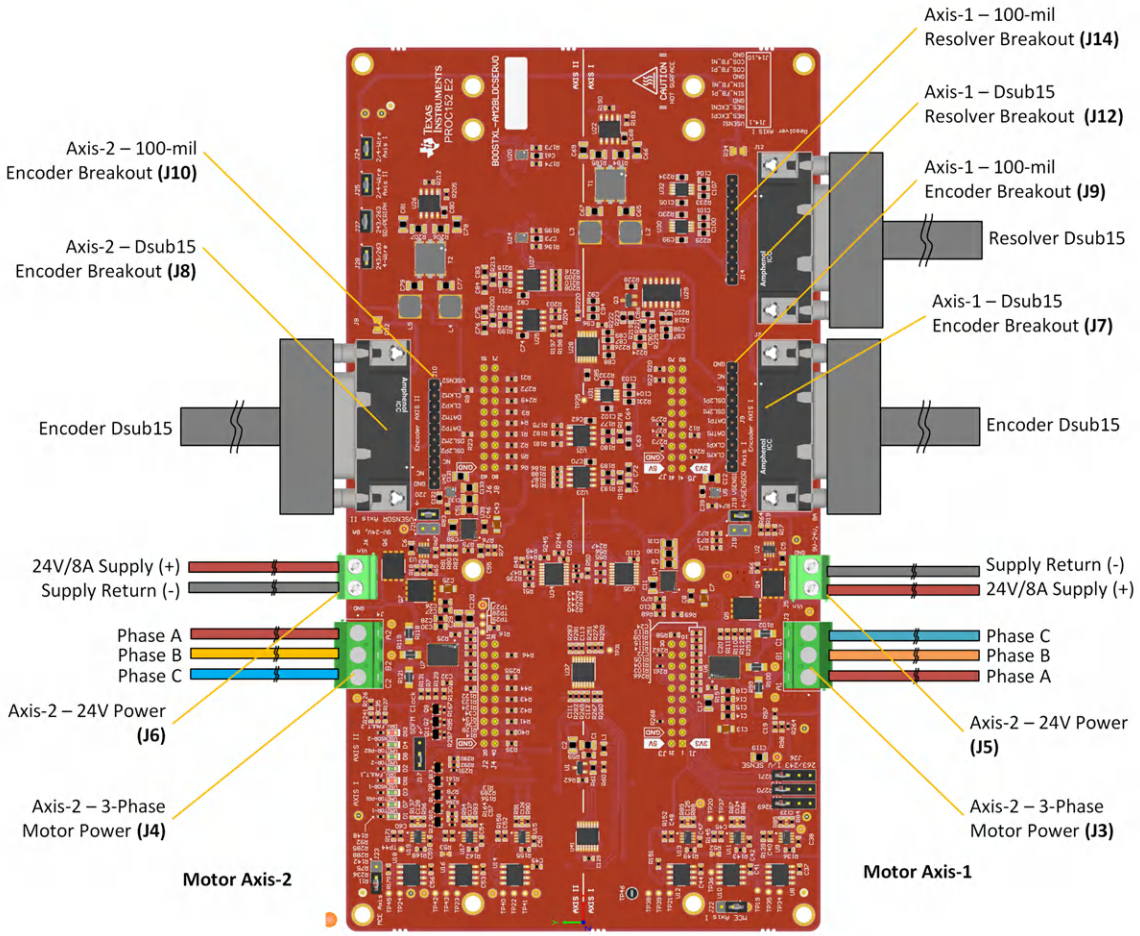
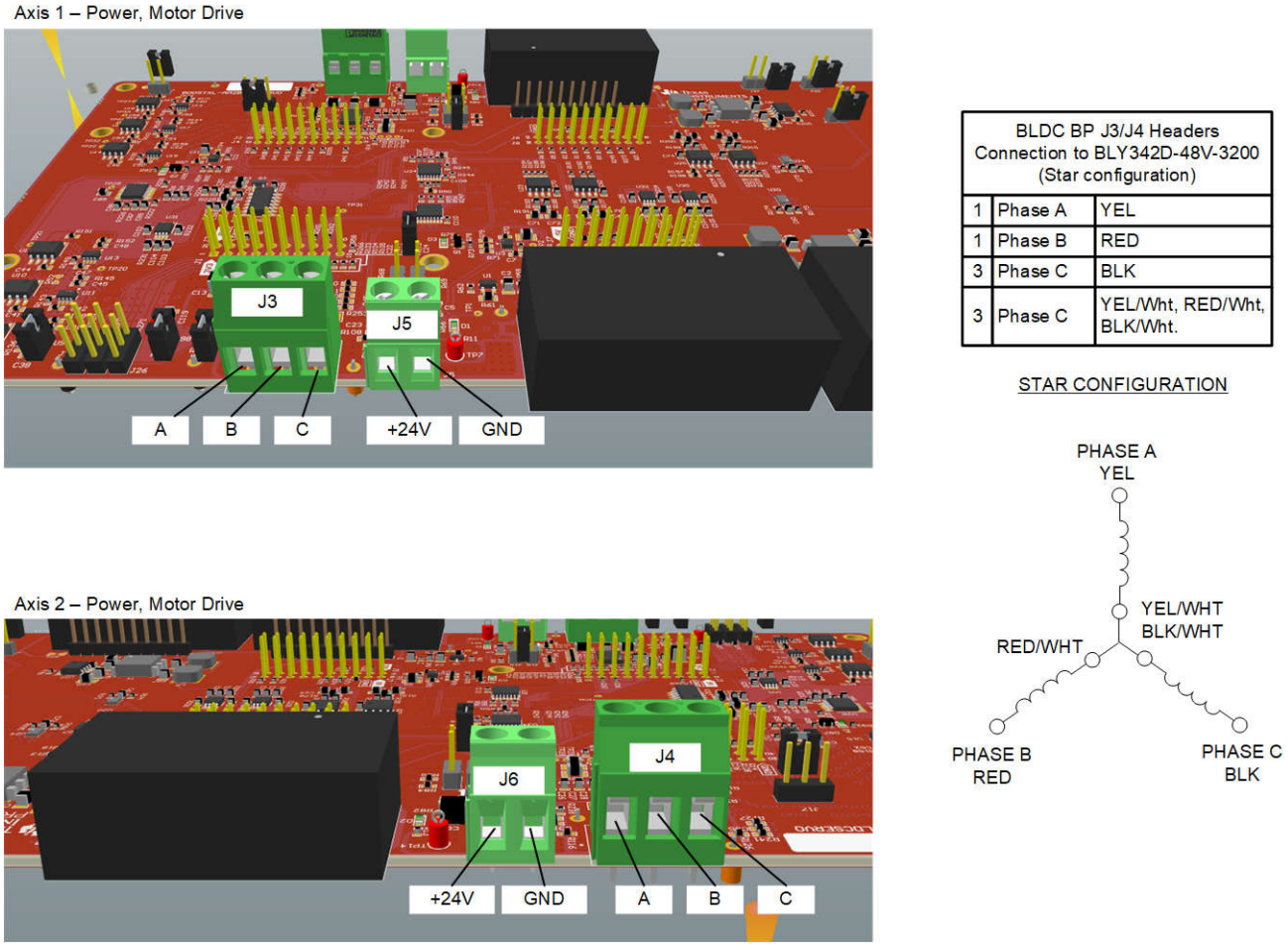


图 4-4. 电机 1 和电机 2 的 BLDC BP E2 连接器



BLDC BP J3/J4 Headers Connection to BLY342D-48V-3200 (Star configuration)		
1	Phase A	YEL
1	Phase B	RED
3	Phase C	BLK
3	Phase C	YEL/Wht, RED/Wht, BLK/Wht.

STAR CONFIGURATION

图 4-5. 电机 1 和电机 2 的星型配置

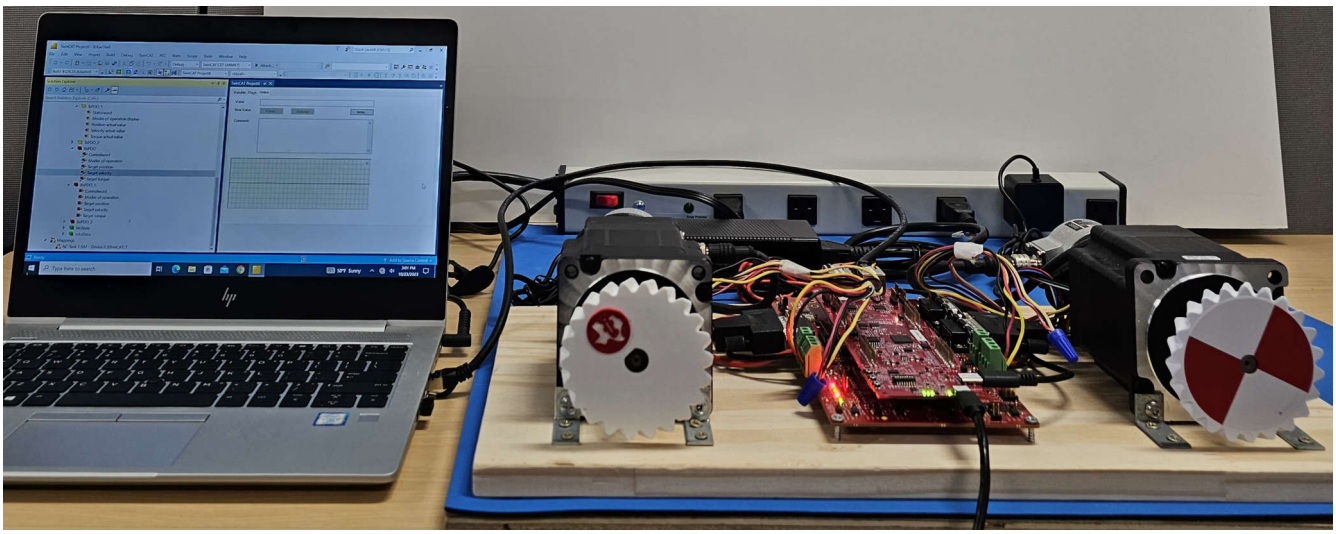
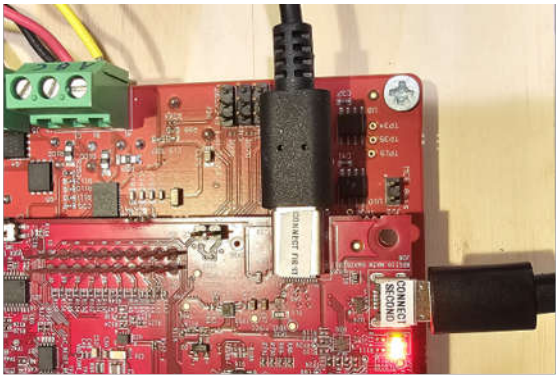


图 4-6. 通过 EtherCAT 连接的电机控制设置



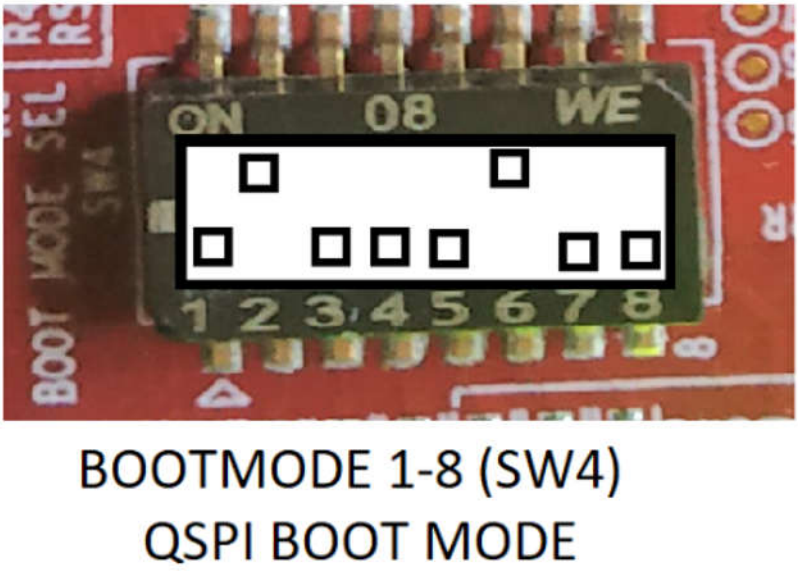


图 4-7. AM243x LP 电源、JTAG 和引导模式

4.2 软件要求

要探索参考设计源代码，请下载电机控制 SDK 09.01.00.xx 并安装在 `C:\ti\motor_control_sdk_am243x_09_01_00_xx`。在电机控制 SDK 09.01.00 中，tidep_01032_dual_motor_drive 保存参考设计的源代码。

从 C:\ti\motor_control_sdk_am243x_09_01_00_xx\examples\tidep_01032_dual_motor_drive\single_chip_servo\am243x-lp\system_freertos_nortos 导入并编译系统项目。该项目有 3 个子项目：

1. `C:\ti\motor_control_sdk_am243x_09_01_00_xx\examples\tidep_01032_dual_motor_drive\single_chip_servo\am243x-lp\system_freertos_nortos \r5fss0-0_nortos`（single_chip_servo_am243x-lp_r5fss0-1_nortos_ti-arm-clang：轴 1 的电机驱动代码）
2. `C:\ti\motor_control_sdk_am243x_09_01_00_xx\examples\tidep_01032_dual_motor_drive\r5fss0-1_nortos`（single_chip_servo_am243x-lp_r5fss0-1_nortos_ti-arm-clang：轴 2 的电机驱动代码）
3. `C:\ti\motor_control_sdk_am243x_09_01_00_xx\examples\tidep_01032_dual_motor_drive\r5fss1-0_freertos`（ethercat_slave_cia402_demo_am243x-lp_r5fss1-0_freertos_ti-arm-clang：EtherCAT CiA402 客户端代码）

4.3 测试设置

本节介绍了如何加载并运行测试软件：

在导入和编译系统项目后，R5F_0_0、R5F_0_1 和 R5F_1_0 的可执行二进制文件会显示在 CCS 工作区目录中：`C:\ti\ccs_ws_1250_am243x_mcsdk_09.01.00.01`

1. 使用目标配置文件连接到目标 AM243x LP
2. 加载并运行电机控制 1 — R5F_0_0
   - 停止 R5F_0_0
   - 加载并运行 single_chip_servo_am243x-lp_r5fss0-0_nortos_ti-arm-clang
   - 电机 1 应以 120RPM 的速度开始旋转

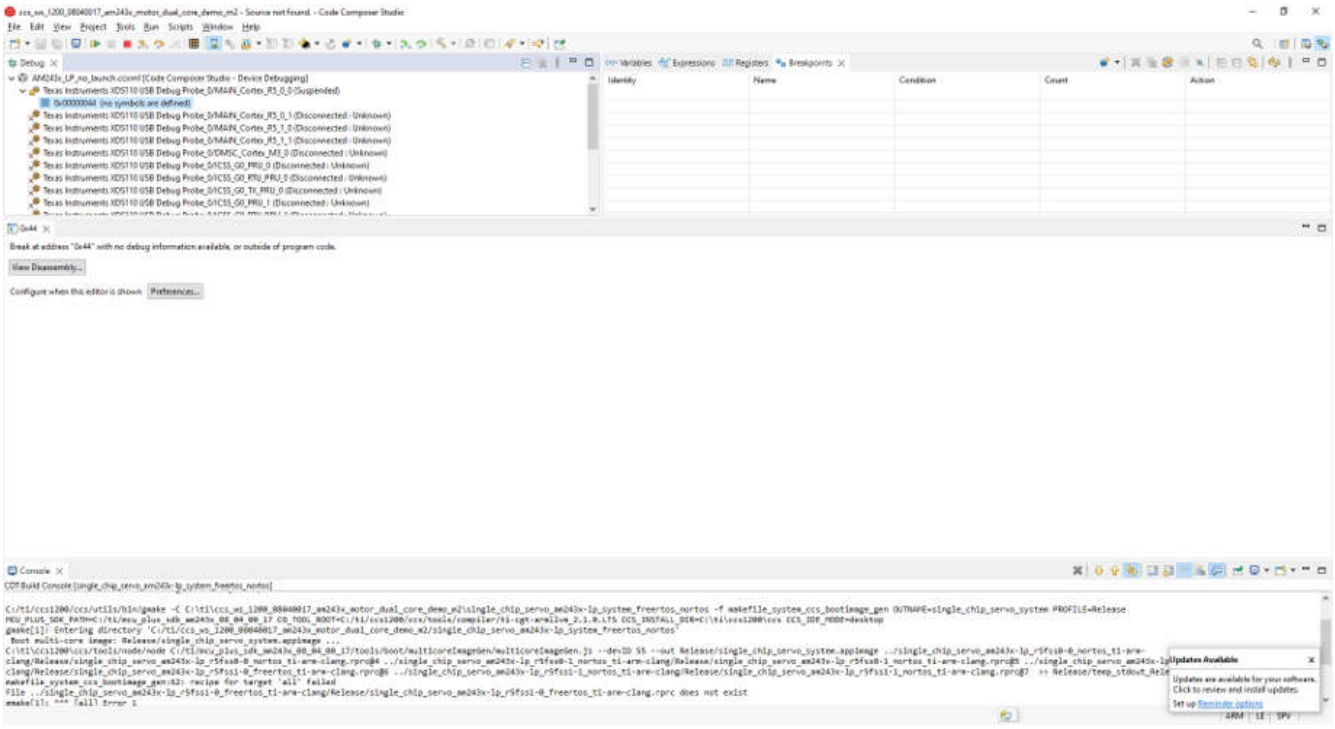


图 4-8. 连接到 R5F_0_0

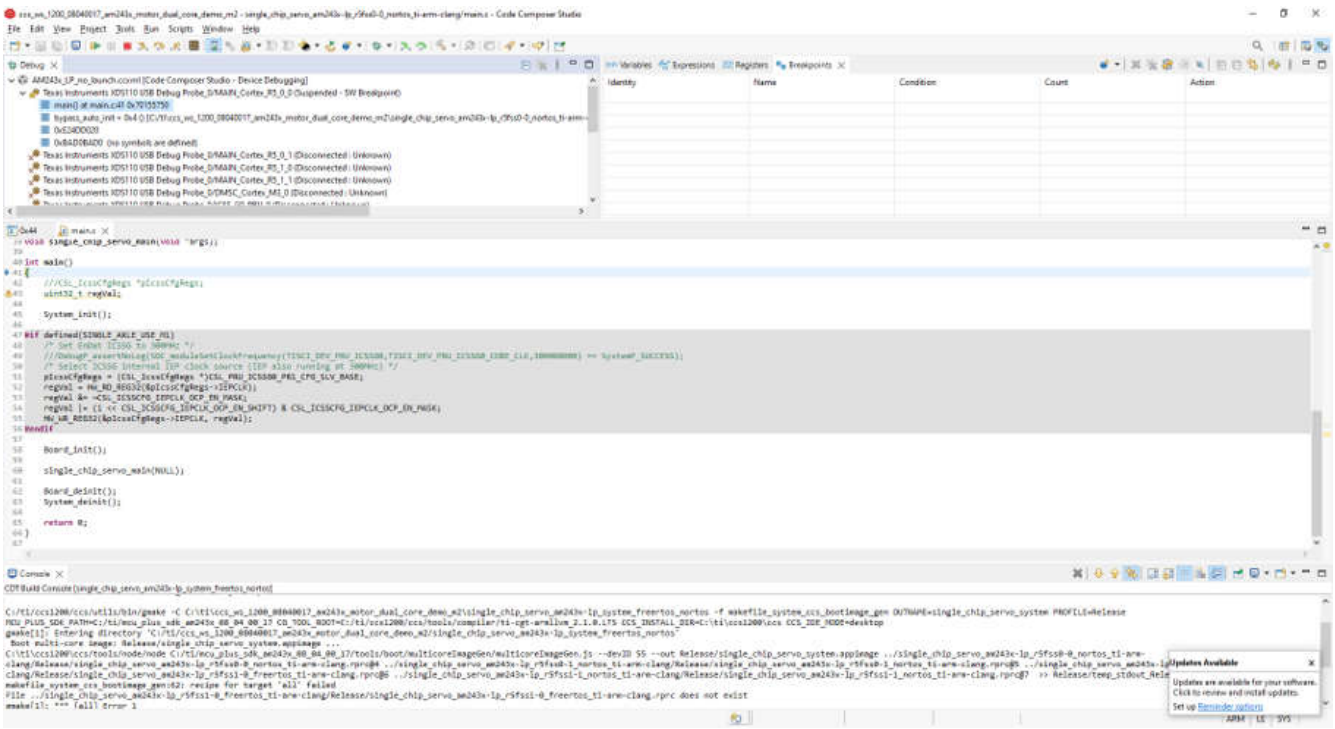


图 4-9. 加载并运行 R5F_0_0

3. 加载并运行电机控制 2 — R5F_0_1
   - 停止 R5F_0_1
   - 加载并运行 single_chip_servo_am243x-lp_r5fss0-1_nortos_ti-arm-clang
   - 电机 2 应以 120RPM 的速度开始旋转

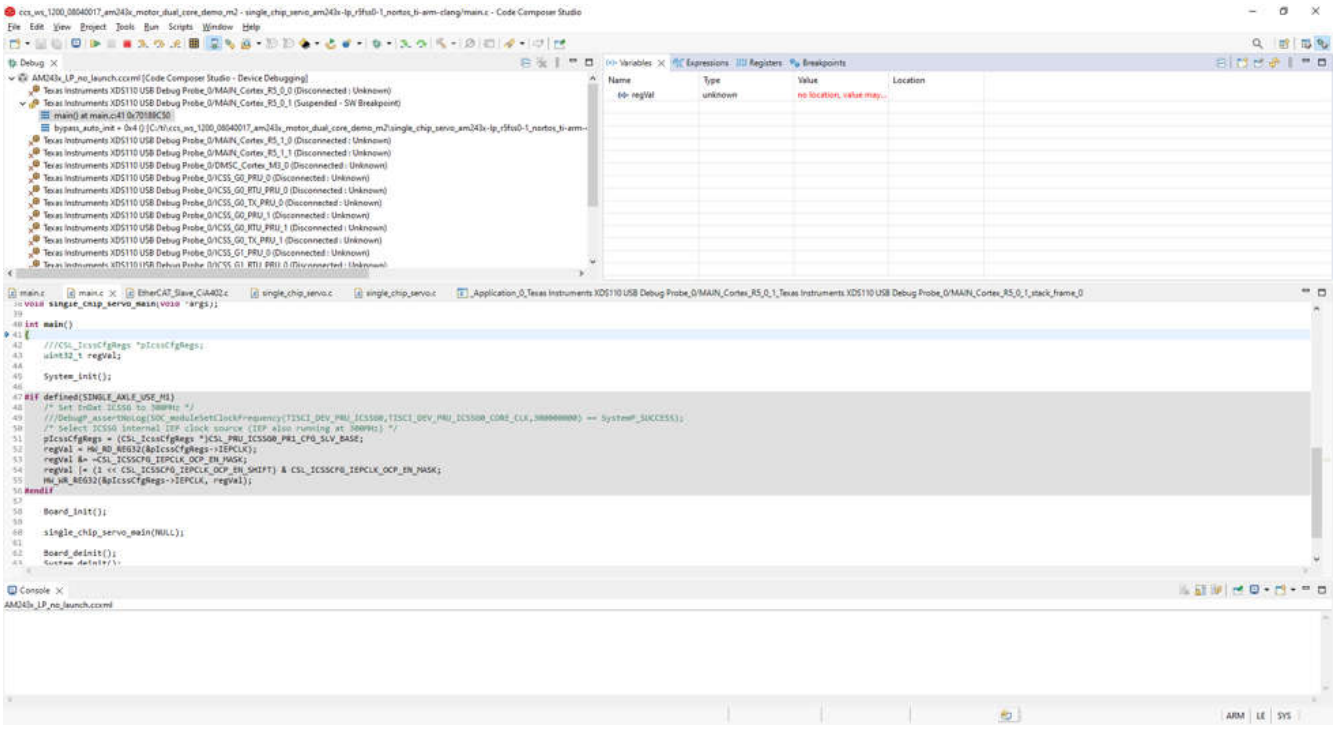


图 4-10. 加载并运行 R5F_0_1

4. 加载并运行 EtherCAT CiA402 客户端 — R5F_1_0
   - 停止 R5F_1_0
   - 加载并运行 ethercat_slave_cia402_demo_am243x-lp_r5fss1-0_freertos_ti-arm-clang

- EtherCAT CiA402 客户端器件现在可以由 TwinCAT (PLC) 进行检测了

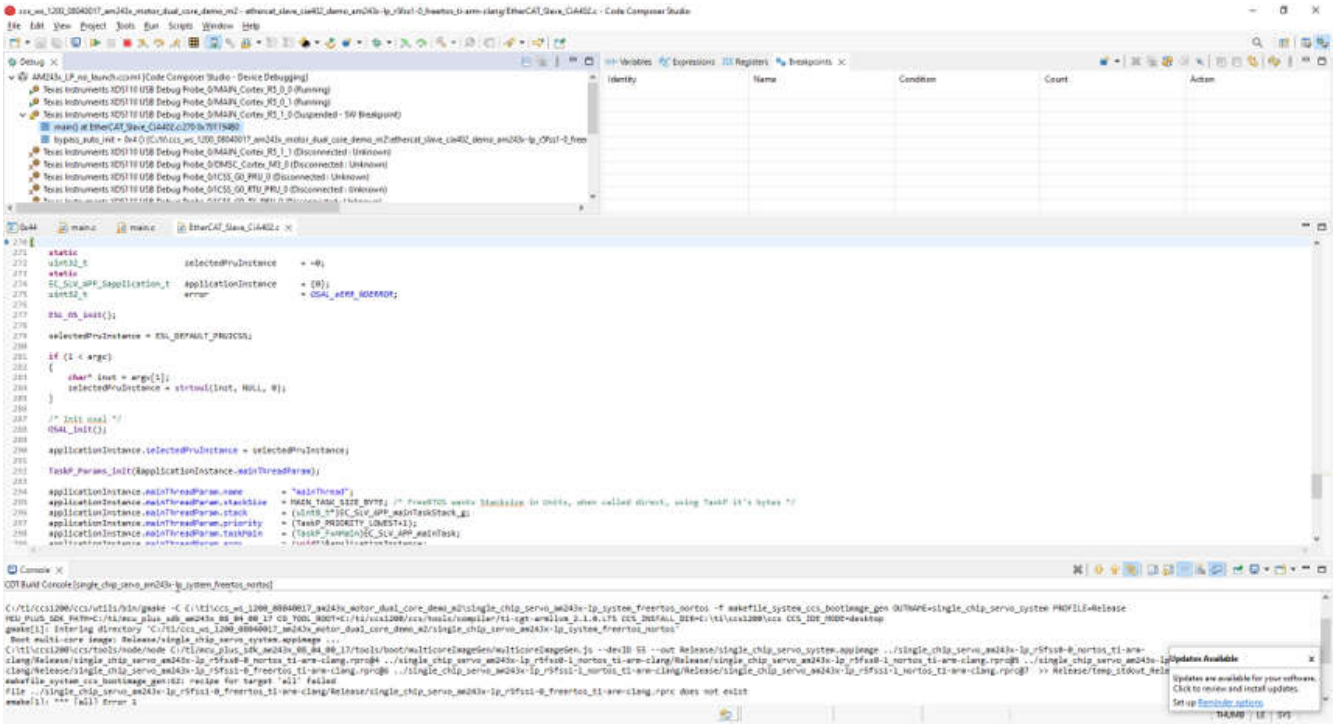


图 4-11. 加载并运行 R5F_1_0

4.4 测试结果

要评估此参考设计，请完成以下步骤：

1. 在 Windows PC 上下载并安装 TwinCAT
2. 启动 TwinCAT 自动化软件
3. 按照 TwinCAT 软件 GUI 中所示创建 EtherCAT 项目：

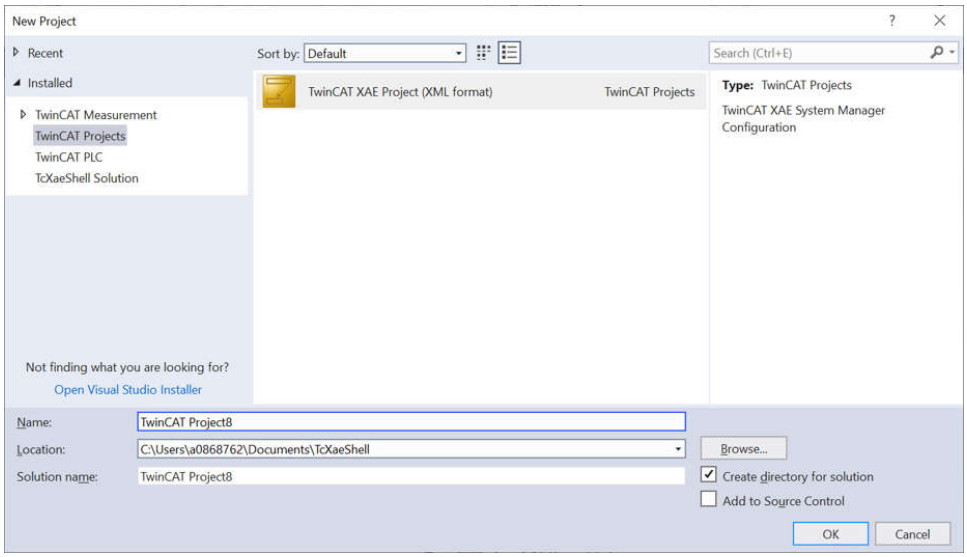


图 4-12. 在 TwinCAT 中创建 EtherCAT® 项目

4. EtherCAT CiA402 — 通过以下操作扫描器件：右键点击 *Devices* → *Scan*。使用以下图像逐步完成该过程。

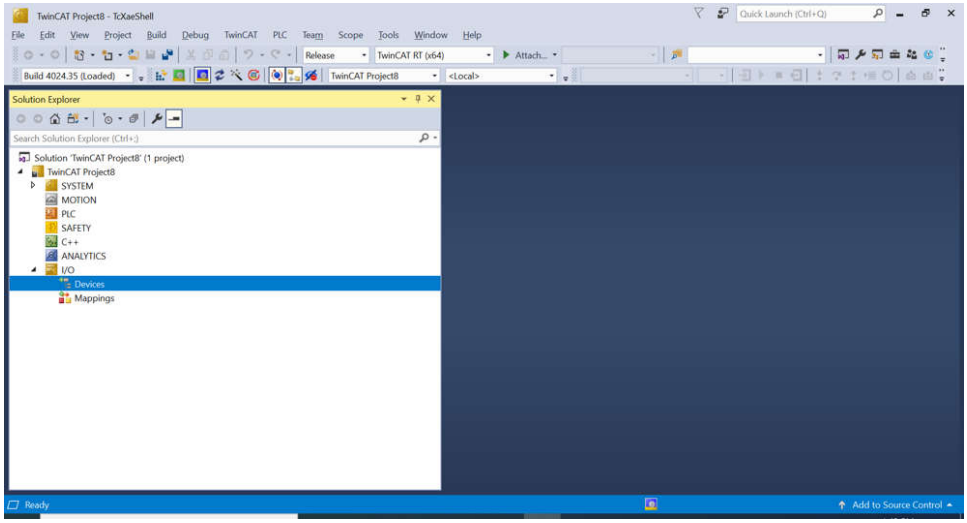


图 4-13. 在 TwinCAT 中扫描 EtherCAT® 器件

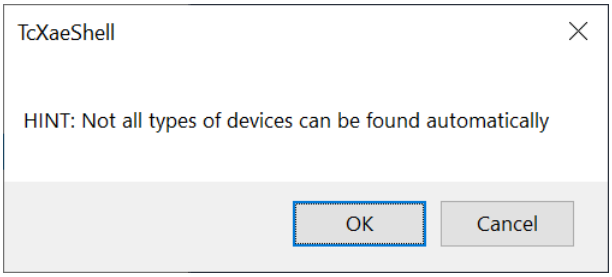


图 4-14. 在 TwinCAT 中扫描 EtherCAT® 器件 (2)

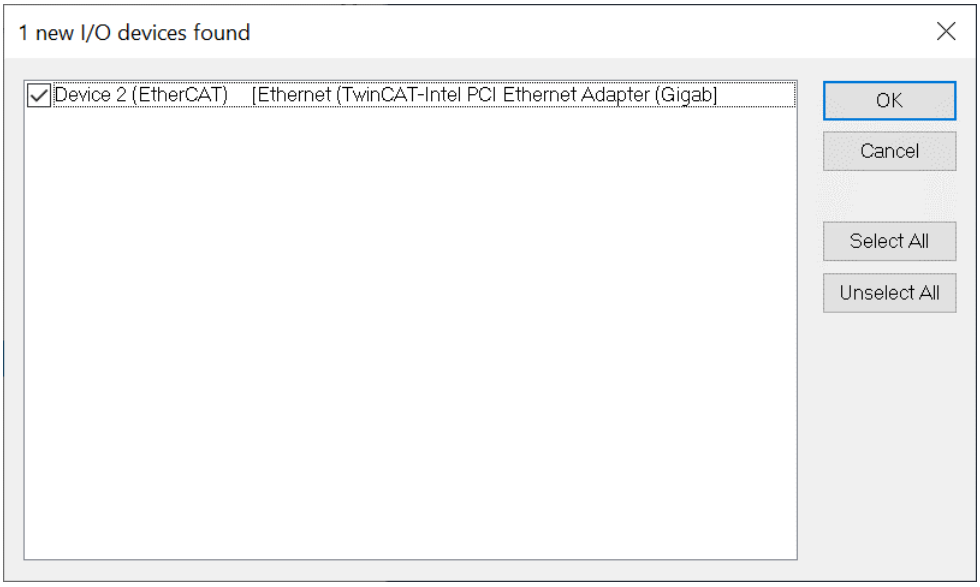


图 4-15. 在 TwinCAT 中扫描 EtherCAT® 器件 (3)

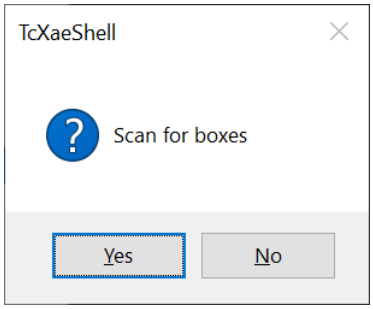


图 4-16. 在 TwinCAT 中扫描 EtherCAT® 器件 (4)

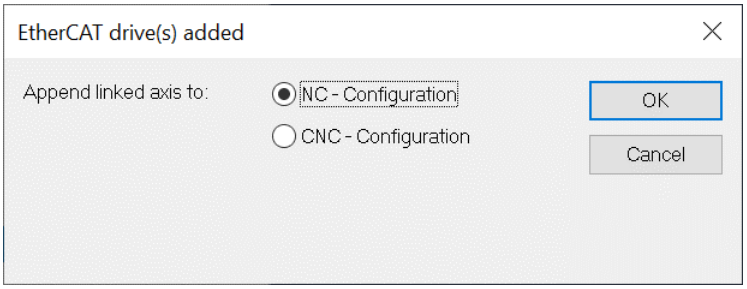


图 4-17. 在 TwinCAT 中扫描 EtherCAT® 器件 (5)

5. EtherCAT CiA402 — 找到器件 1（AM243X.R5F 的 TI EtherCAT Toolkit CiA402）

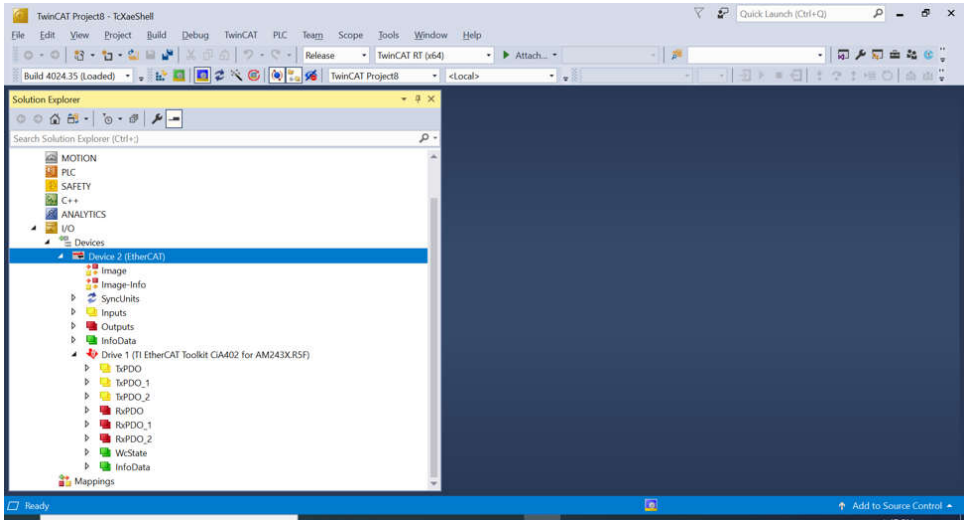


图 4-18. 通过 TwinCAT 找到 EtherCAT® 器件

6. EtherCAT CiA402 — 将 RxPDO (电机 1) *Target velocity* 更改为 240 (240RPM)

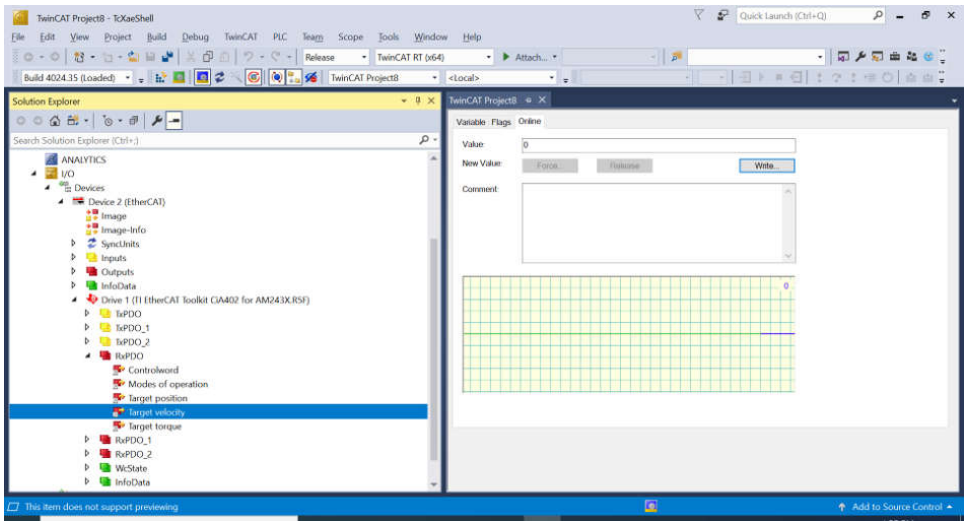


图 4-19. 在 TwinCAT 中更改电机 1 的目标速度

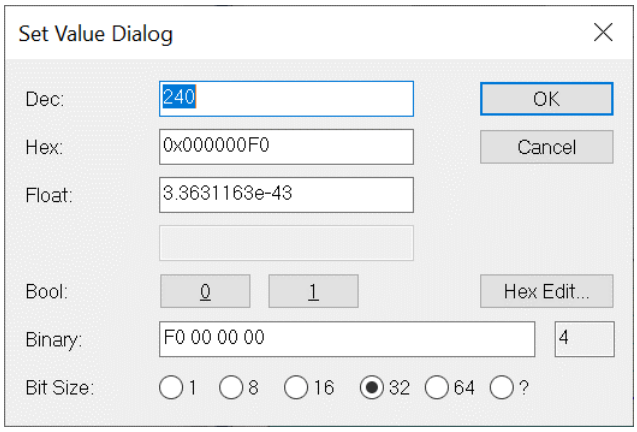


图 4-20. 在 TwinCAT 中更改电机 1 的目标速度 (2)

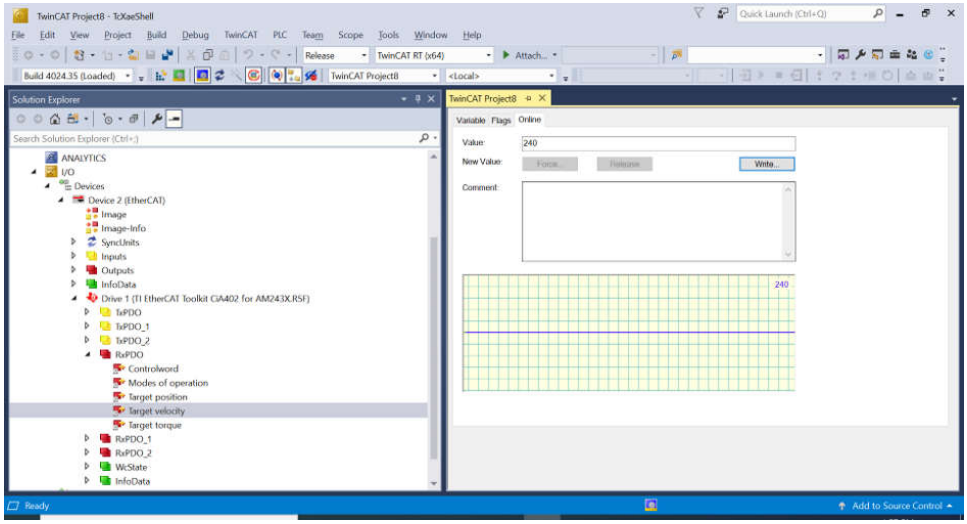


图 4-21. 在 TwinCAT 中更改电机 1 的目标速度 (3)

7. EtherCAT CiA402 — 将 RxPDO（电机 1）*Modes of Operation* 更改为 "9" (*Cyclic synchronous velocity mode*)

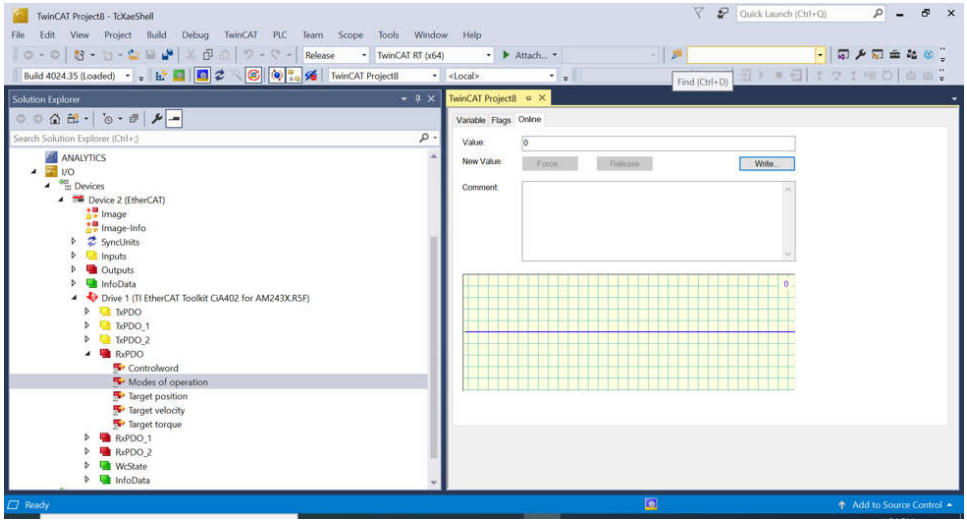


图 4-22. 在 TwinCAT 中更改电机 1 的工作模式

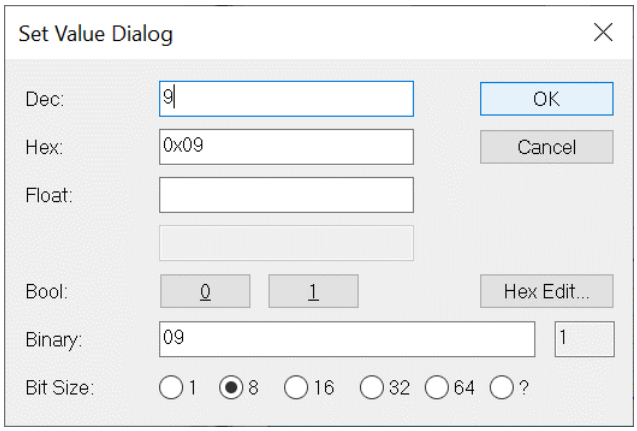


图 4-23. 在 TwinCAT 中更改电机 1 的工作模式 (2)

8. EtherCAT CiA402 — 将 RxPDO（电机 1）*Controlword* 更改为 "15" (*Switch On* | *Enable Voltage* | *Quick Stop* | *Enable Operation*)

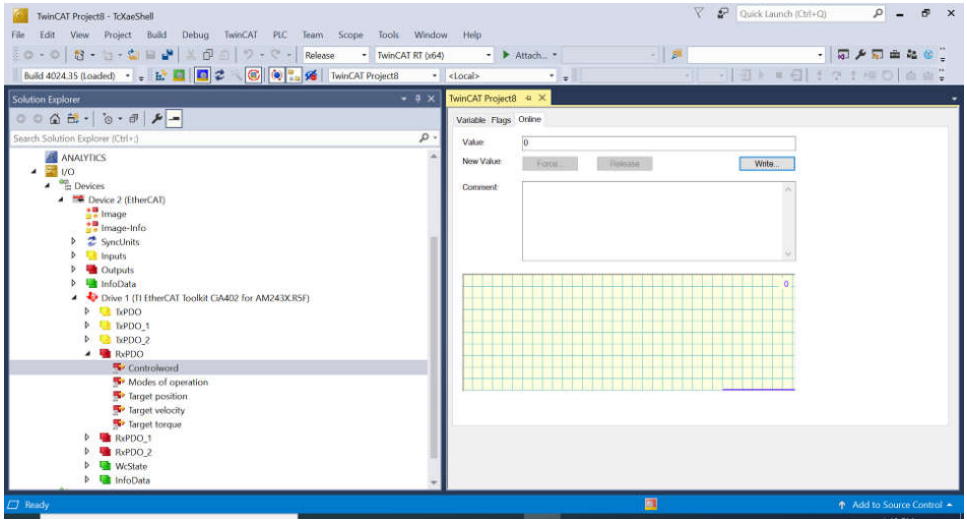


图 4-24. 在 TwinCAT 中更改电机 1 的控制字

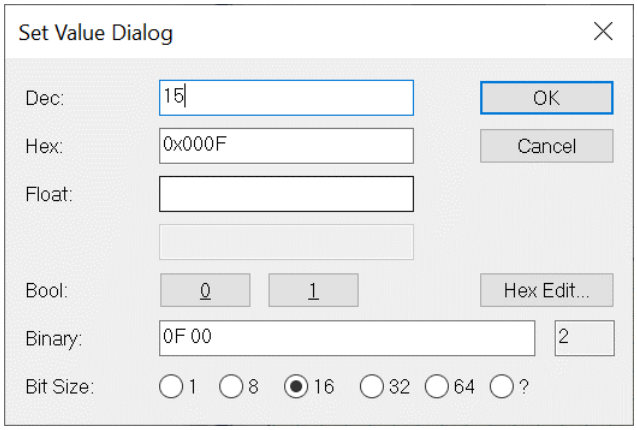


图 4-25. 在 TwinCAT 中更改电机 1 的控制字 (2)

9. 更改此设置后，电机 1 的转速从 120RPM 更改为 240RPM
10. EtherCAT CiA402 — 检查 TxPDO（电机 1），确保 *Velocity actual value* 为 240 (240RPM)

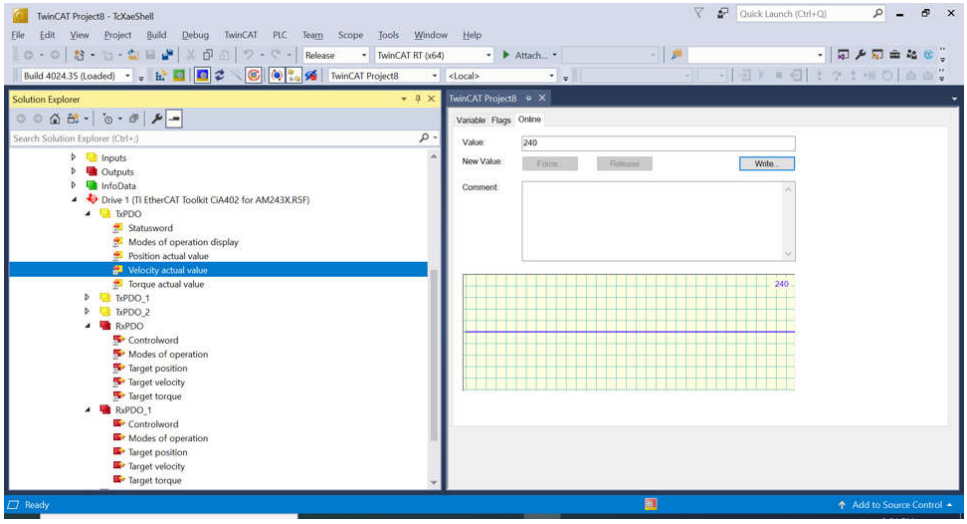


图 4-26. 在 TwinCAT 中检查电机 1 的实际速度

11. EtherCAT CiA402 — 将 RxPDO_1（电机 2）*Target velocity* 更改为 180 (180RPM)

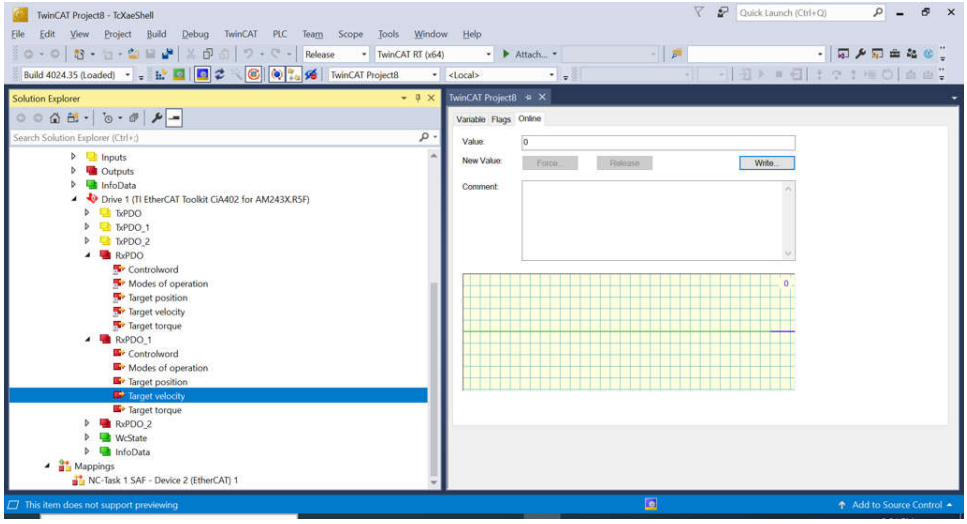


图 4-27. 在 TwinCAT 中更改电机 2 的目标速度

12. EtherCAT CiA402 — 将 RxPDO_1（电机 2）*Modes of Operation* 更改为 "9" (*Cyclic synchronous velocity mode*)

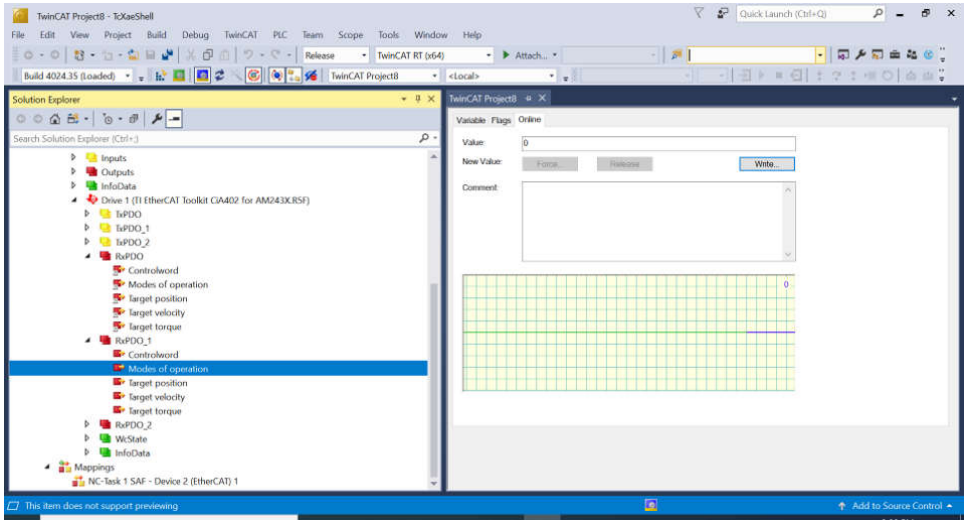


图 4-28. 在 TwinCAT 中更改电机 2 的工作模式

13. EtherCAT CiA402 — 将 RxPDO_1（电机 2）*Controlword* 更改为 "15" (*Switch On* | *Enable Voltage* | *Quick Stop* | *Enable Operation*)

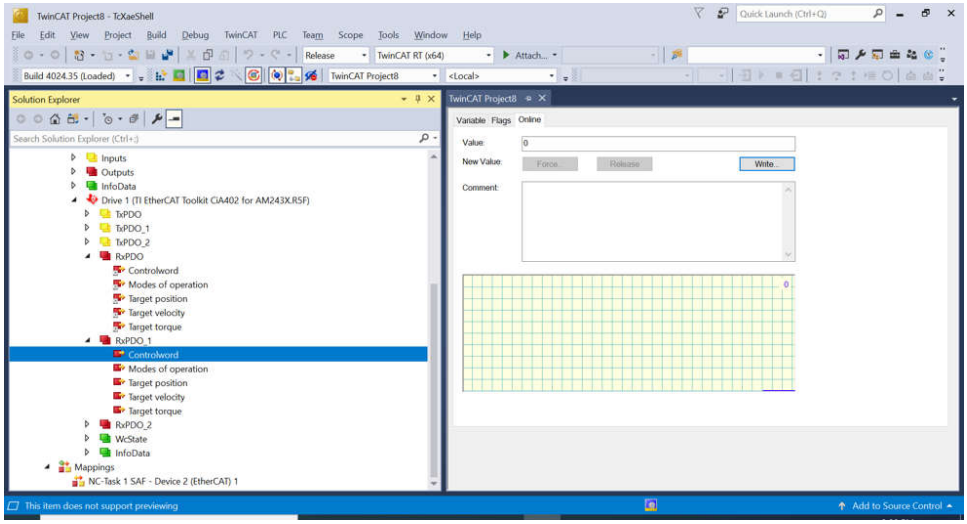


图 4-29. 在 TwinCAT 中更改电机 2 的控制字

14. 在进行前面的更改后，电机 2 的目标速度为 180RPM

15. EtherCAT CiA402 — 检查 TxPDO1（电机 2），确保 *Velocity actual value* 为 180 (180RPM)

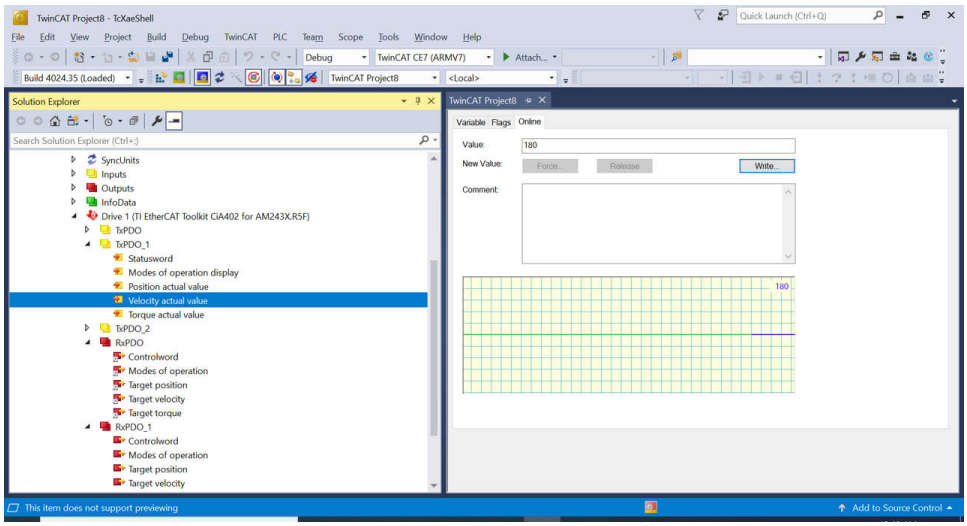


图 4-30. 在 TwinCAT 中检查电机 2 的实际速度

5 设计和文档支持

5.1 设计文件

5.1.1 原理图

此参考设计有两个相关的原理图：

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 BLDC BP 的物料清单 (BOM)，请参阅 BP-AM2BLDCSERVO 设计包页面上的设计文件

要下载 AM243x LP 的物料清单 (BOM)，请参阅 LP-AM243 设计包页面上的设计文件

5.2 工具与软件

工具

CCSTUDIO	Code Composer Studio™ 集成开发环境 (IDE)：下载适用于 Windows 或 Linux 的 CCS 12.5.0 版
ARM-CGT-CLANG	Arm® 代码生成工具 — 编译器：下载适用于 Windows 或 Linux 的 TI ARM CLANG 3.2.0.LTS
SYSCONFIG	SysConfig 独立桌面版本：下载适用于 Windows 或 Linux 的 SysConfig 1.18.0

软件

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

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

6 作者简介

MING WEI (MGTS) 是 Sitara MCU 的高级软件工程师，他为 Sitara MPU/MCU 和 DSP 系列的 SOC 器件开发和支持 Processor SDK RTOS/MCU+ SDK/电机控制 SDK。Ming 将他在电机控制、实时系统、信号处理和代码优化方面的丰富经验和知识运用到工作中。Ming 分别从西安交通大学和北德克萨斯大学获得计算机科学学士、硕士和博士学位。

重要声明和免责声明

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 产品的应用。严禁对这些资源进行其他复制或展示。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