Application Note

在汽车 SiC 牵引逆变器中使用 Sitara MCU AM263x

摘要

牵引逆变器是电动汽车中的主要电源处理系统。该电源处理块的实时控制是一项关键任务，需要一个具有低延迟外设和确定性处理时间的微控制器单元 (MCU)。新推出的德州仪器 (TI) MCU AM263x 非常适合该应用。但是，为了加快 MCU 的市场推广，必须在与电动汽车牵引逆变器非常匹配的参考设计中对其进行演示。该演示也将帮助客户显著缩短软件开发时间，专注于产品的开发。本文档重点介绍牵引逆变器的软件框架，并借助参考设计*汽车类大功率、高性能 SiC 牵引逆变器参考设计* 演示了 AM263x 的实时控制模块的功能。

内容

插图清单

商标

1 简介

本文档的主要目的是介绍如何在汽车牵引逆变器参考设计中使用德州仪器 (TI) Sitara™ AM263x 系列微控制器 (MCU)。本参考设计是 TI 和 Wolfspeed 提供的基于 SiC 的 800V、300kW 逆变器参考设计，该参考设计尝试为设计人员和工程师提供着手点，来实现高性能、高效率牵引逆变器系统（*汽车类大功率、高性能 SiC 牵引逆变器参考设计*）。本参考设计的应用手册详细介绍了硬件和软件设计注意事项。但是，本应用手册不涵盖与 AM263x 相关的软件设计方面。本文档尝试涵盖该方面，以便能从各个方面完善参考设计文档。

WARNING

TI 建议，本应用手册仅可在实验室环境中使用，不应将此参考设计作为成品供一般消费者使用。

TI 建议，本应用手册仅可由熟悉处理高压电子和机械部件、系统及子系统所存在相关风险的合格工程师和技术人员使用。

高电压！ 电路板中存在可接触的高电压。如电路板的电压和电流处理不当或施加不正确，则可能导致电击、火灾或伤害事故。使用该设备时应特别小心，并采取相应的保护措施，以避免伤害自己或损坏财产。

CAUTION

请勿在无人照看的情况下使该设计通电。

1.1 主要系统技术规格

表 1-1 总结了关键系统规格。

表 1-1. 关键系统规格

参数	规格（单位）	说明
P_{OUT}	300kW	额定输出功率
V_{DSmax}	1200 V	最大漏源电压
V_{DC}	800V	建议的直流总线电压
I_{DC}	300 A	直流总线电流
f_{SWmax}	60kHz	基于栅极驱动器辅助电源
I_L	360 A	交流输出 RMS 电流
L_{PL}	5.3nH	寄生电感，包括直流链路电容器和汇流条
C_{DC}	300 µF	直流链路电容器
L_{DC}	3.5nH	直流总线电容器 ESL
功率密度	32 kW/L	
尺寸	28cm × 29cm × 11.5cm	
重量	6.2 kg	
体积	9.3L	
面积	812 cm^2	
P	5 巴	冷却液工作压力
ΔP	200 毫巴	压力降

- 有关隔离式栅极驱动器的信息，请参阅 UCC5880-Q1 数据表。
- 有关微控制器的信息，请参阅 AM2634-Q1 和 TMSF280039C-Q1 数据表。
- 有关辅助电源的信息，请参阅 UCC14240-Q1 数据表。
- 有关集成模块的信息，请参阅 CAB450M12XM3 数据表。
- 对于较高的环境温度，必须根据所包含的直流链路电容器额定值对直流链路电压和直流链路电流进行降额。有关更多详细信息，请参阅 FTCAP GmbH 提供的 1100V/100 µF CX100µ1100d51KF6 数据表。

- 随附的冷板是 Wieland MicroCool CP3012-XP。要根据流量（升/分钟）计算热阻 (°C/W) 和压力降（巴），请参阅 Wieland MicroCool Inc. 提供的 CP3012-XP 数据表，以了解更多详细信息。
- 随附的电流传感器板使用 LEM LF 510-S。有关更多详细信息，请参阅 LEM USA Inc 提供的 LF 510-S 数据表。

2 AM263x 概述

AM263x 是一个片上系统 (SoC)，具有 Arm® Cortex®-R5F 集群和专用于实时控制的加速器。集群可以配置为双核模式或锁步模式。加速器命名为控制子系统，包括 ADC、DAC 和 PWM 等接口模块。本文档介绍了如何设置 R5F 内核、牵引逆变器功率级所需的一组接口模块，以及如何使用参考设计 TIDM-02014 实现功率级控制。有关 SoC 的更多详情，请参阅 *AM263x Sitara™ 微控制器* 数据表和 *AM263x Sitara 处理器技术参考手册*。有关逆变器硬件的详细信息，请参阅*汽车类大功率、高性能 SiC 牵引逆变器参考设计* 的设计指南。

Arm® Cortex®-R5F 集群包括两个 R5F 内核，附带 L1 高速缓存和紧耦合存储器 (TCM) 之类的存储器、标准 Arm CoreSight™ 调试和布线架构、集成式矢量中断管理器 (VIM)、ECC 聚合以及支持协议转换和地址转换的各种其他模块，以便于集成到 SoC。更详细的方框图请参阅 AM263x 技术参考手册。使用 AM263x 进行实时控制的关键在于了解高速缓存和 TCM 的影响。在编写程序时，可以通过链接命令文件将指令和数据分配给片上 RAM 或 TCM。在执行过程中，片上 RAM 中经常使用的指令和数据会自动进入高速缓存。因此，执行时间显著改善。但是，片上 RAM 中的数据直到从高速缓存写回后才更新。当数据在高速缓存中时，只能通过内核中运行的指令访问数据。Code Composer Studio™ (CCS) 等集成开发环境 (IDE) 的存储器视图将无法读取数据。然而，通过在内核内部运行通用异步接收器/发送器 (UART) 的一段程序，也可以使用 CSS 读取高速缓存。本应用手册详细讨论了 UART 方法的详细信息：*用于牵引逆变器的 AM263x*。另一方面，分配给 TCM 的指令和数据保存在地址中，可随时提供给存储器视图。通常情况下，程序在高速缓存和 TCM 中的执行速度相当快，而片上 RAM 中的程序执行则要慢得多。另外，从片上 RAM 向高速缓存转移程序的操作需要一些时间，并导致不可预测的延迟。如果此延迟对于应用要求很重要，强烈建议将应用程序存储在 TCM 中。有关 TCM 地址的详情请参阅 AM263x 技术参考手册。在此示例中，场定向控制的中断程序和软件旋转变压器数字转换器位于 TCM 中。链接命令文件作为示例放在 CCS 工程文件夹中。

实时控制加速器继承了全球广泛使用的德州仪器 (TI) 经典 C2000 控制模块。它包括模数转换器 (ADC)、模拟比较器、缓冲数模转换器、增强型脉宽调制器 (EPWM)、增强型捕捉、增强型正交编码器脉冲、快速串行接口、Σ-Δ 滤波器模块和纵横制。有关这些模块的详情请参阅 AM263x 技术参考手册。可以使用直观的系统配置工具 SYSCONFIG 来配置这些模块，从而减少对详细实施情况的关注。AM263x *软件开发套件* (SDK) 提供了 SYSCONFIG 工具的详细信息。模块同步的关键是在 EPWM 时基部分中配置 PWM 同步输入/输出，在 EPWM 事件触发部分中配置 ADC 转换开始 (SOC) 触发。时基用于对齐多个 PWM 通道，而事件触发用于同步 ADC、DMA 和中断等特性。牵引逆变器演示的 CCS 工程文件夹中有一个牵引逆变器示例。本例中设置了一个 PWM 通道，通过 DMA 和 DAC 以更高的频率触发旋转变压器激励信号更新，还使用三个 PWM 通道生成逆变器信号和 ADC SOC。这样，来自 DAC 的旋转变压器激励信号将对齐到所需的 ADC 采样相位。由于多个 ADC 器件可以共享同一个 SOC，因此可以在多个 ADC 器件中同时采集多个样本。在一个 ADC 器件中，可以在 SOC 配置部分中配置采样序列，并在 INT 配置中设置 ADC 中断。中断可以在一个 ADC 转换开始或结束时触发。有关 PWM 和 ADC 的部分简单示例可在 AM263x SDK 的 *\examples\drivers\epwm* 和 *\examples\drivers\adc* 下找到。有关 API 的更多详情，可在 AM263x SDK 的 *\source\drivers* 下找到。头文件的更多细节都在注释中。

2.1 AM263x 控制卡和牵引系统框架

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。控制卡还支持通过接头引脚访问关键信号，方法是使用高速边缘连接器 (HSEC)，该连接器可以直接插入到 TIDM-02014 主控制板上。

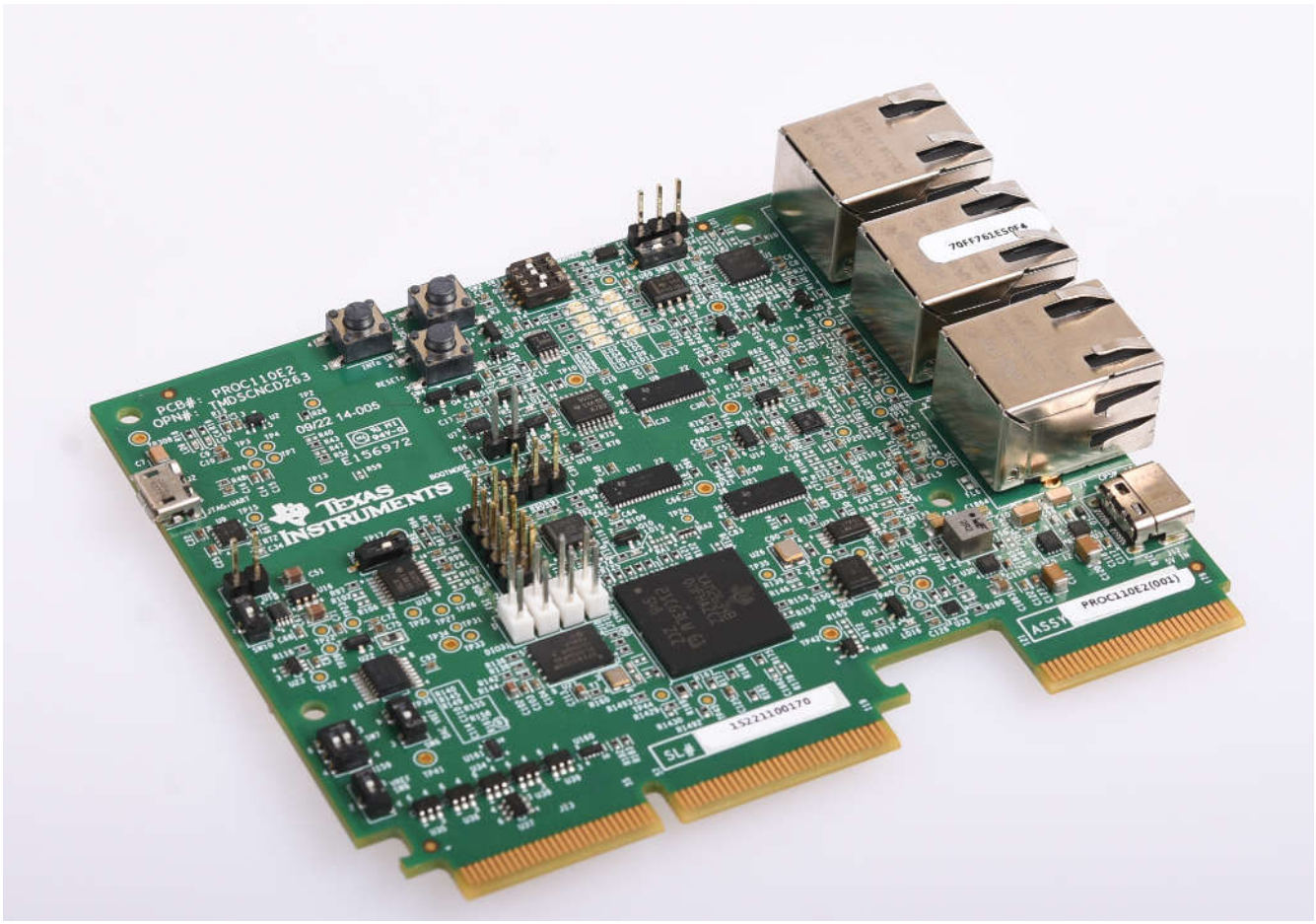

图 2-1. AM263x Sitara 控制卡

牵引框架包括图 2-1 中突出显示的资源，并内置到图 2-2 中的牵引系统。TIDM-02014 是牵引逆变器参考设计硬件。它包括功率级、栅极驱动器、电流和电压采样、旋转变压器模拟前端、控制系统电源树和连接接口。电机包括接受正弦波激励并发送调制反馈、以检测位置的旋转变压器。场定向控制使用集群 0 内核 0 实现。ADC INT1 的实时控制部分将分配给 TCM，以精准确定执行时间。

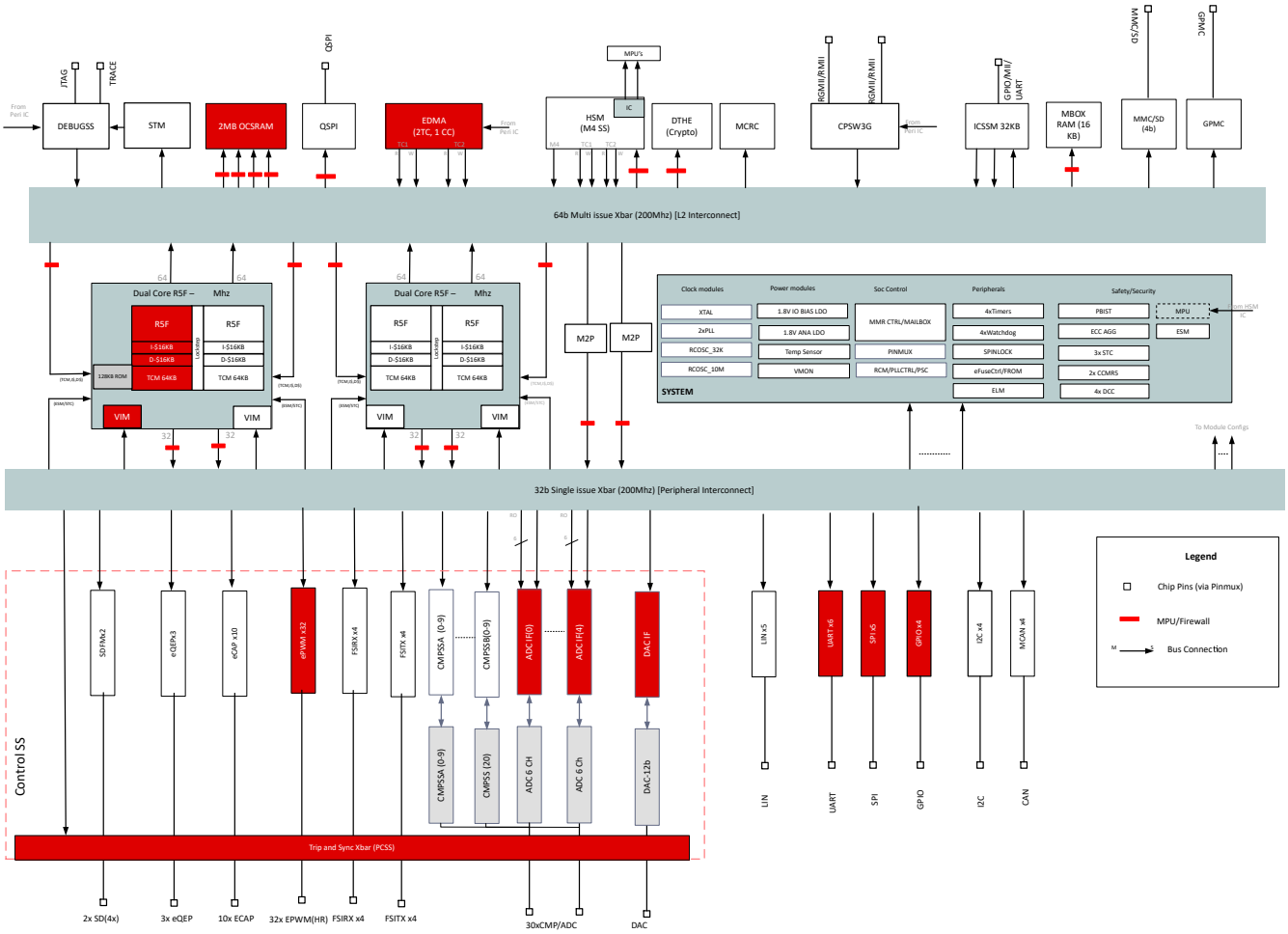

图 2-2. 牵引框架资源

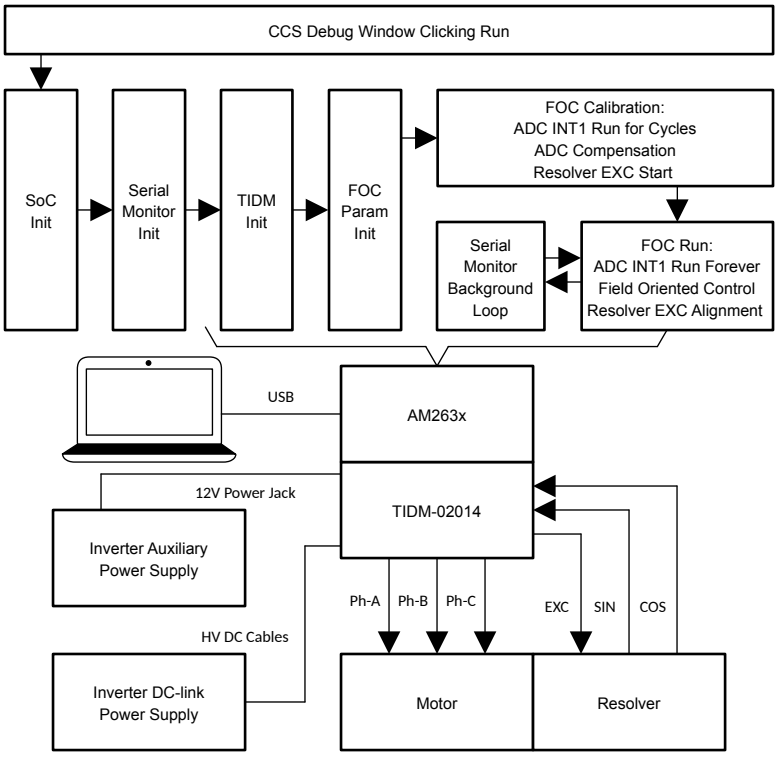

图 2-3. 牵引系统图

3 运行 TIDM-02014 牵引逆变器的指南

3.1 软件设置

如图 2-3 所示设置牵引系统后，TIDM-02014 参考设计需要设置软件环境才能运行该系统。为此，我们使用了 *Code Composer Studio* (CCS) 作为集成开发环境。

第一步，请下载并安装建议用于该工程的 CCS 版本 12.2.0 或更高版本。*CCS 用户指南* 提供了有关 CCS 安装和使用的更多详细信息。

虽然严格意义上没有必要建立实时调试接口，但是在调试任何使用 MCU 的功率转换应用时，建立一个实时调试接口有很多帮助。对于 AM263x，通过 CCS 与 AM263x 之间的 UART 连接启用实时调试。AM263x 控制卡的用户界面如图 3-1 所示。有关硬件连接的详细信息，请参阅 *AM263x 控制卡用户指南*。控制卡在一个 USB Micro-B 连接器中同时提供 JTAG 和 UART 调试端口。有关为调试端口创建目标配置文件的详细说明，请参阅应用手册*适用于牵引逆变器的 AM263x*。由于本应用手册还详细介绍了如何配置控制外设、中断和驱动程序接口，因此我们在此不再重复这些部分。

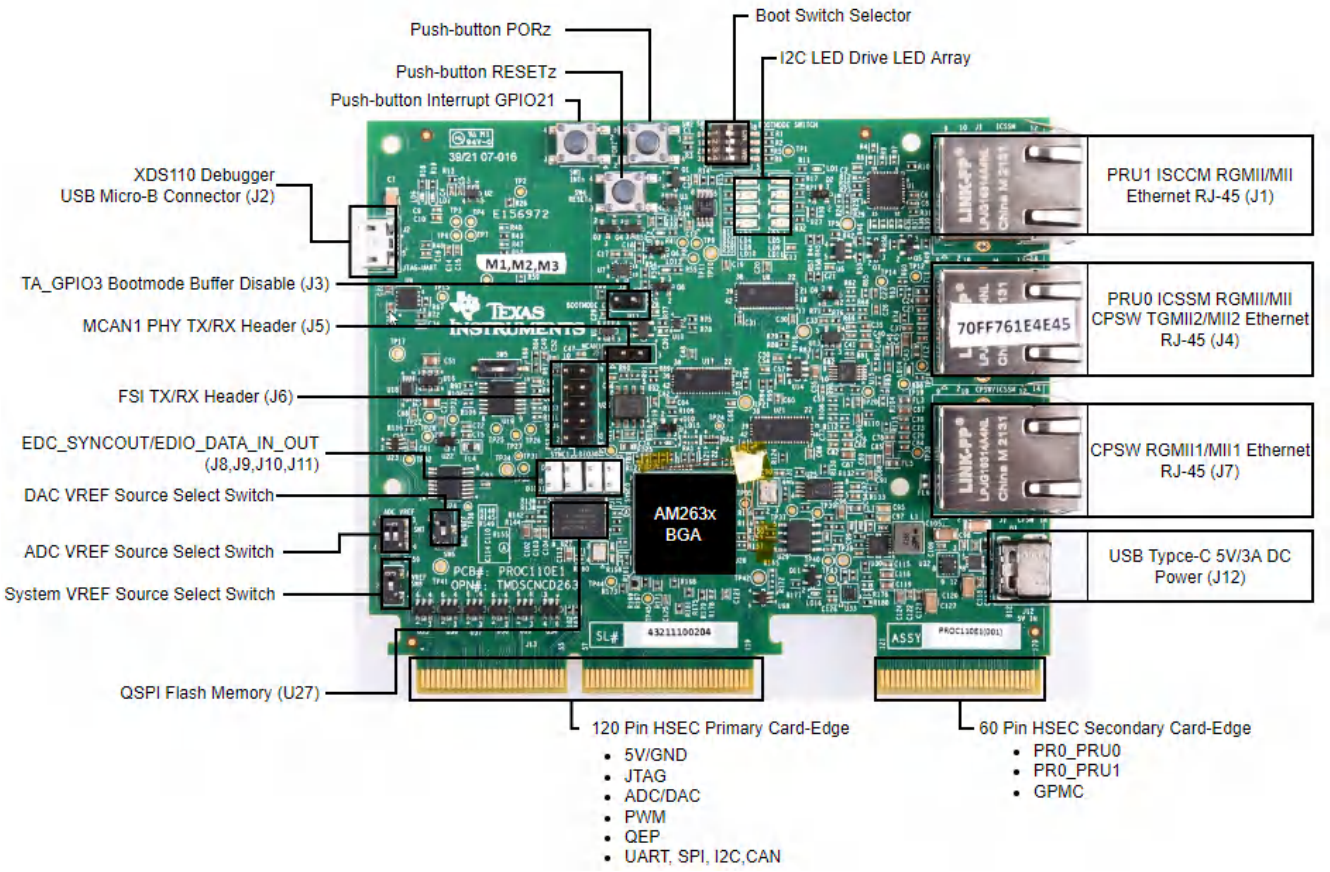

图 3-1. AM263x 控制卡的用户界面

3.1.1 Code Composer Studio 工程

要将 TIDM-02014 的软件工程导入 CCS，请点击“Project”→“Import CCS Projects”，然后浏览至 *TractionDemo_am263x-cc-rev2_r5fss0-0_nortos_ti-arm-clang2* 文件夹并点击“Select Folder”。选择名为 TractionDemo_am263x-cc-rev2_r5fss0-0_nortos_ti-arm-clang2 的工程，然后点击“Finish”。现在，该工程显示在 CCS 的 Project Explorer 窗格中。用户指南提供了有关*将工程导入 CCS* 的更多详细信息。

libraries/foc 文件夹包含典型的 FOC 模块，包括 Park 和 Clarke 变换、PID 函数和估算器。这些模块独立于特定器件和电路板。

Motor 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。用户可以添加自己的用于系统控制、通信等功能的代码。这些模块专用于此参考设计工程，但与器件和电路板无关。

特定于电路板、特定于电机和特定于器件的文件位于根文件夹和 Motor 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。如果要将工程迁移到您自己的电路板或其他器件，只需根据器件或电路板的引脚分配和功能更改 trinv.syscfg、Motor_param.h 和 Resolver_param.h 文件。

3.1.2 软件结构

图 3-15 中显示了工程的总体结构。器件外设配置基于 AM263x Driverlib，部分使用 SysConfig 生成，使代码可跨硬件和器件移植。要将参考设计软件移植到不同的电路板或器件，用户只需更改 trinv_hal.c、trinv_hal.syscfg 和 trinv_hal.h 文件以及 trinv_settings.h 中的参数。

??? 图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、一个允许用户通过调试窗口更新电机控制参数的主循环。ISR 由 ADC 转换结束 (EOC) 触发。在主 ISR 中运行的函数在 trinv.h 头文件中定义。此外，在该设计中，通过旋转变压器接口来检测准确的电机位置。读取解析器信号的 ADC 值并执行相应位置、速度计算的函数在作为独立处理内核的控制律加速器 (CLA) 中运行。该函数在 trinv_cla_tasks_cpu1.cla 文件中定义。

3.2 创建实时调试接口

对于 AM263x，通过 CCS 与 AM263x 之间的 UART 连接启用实时调试。通过实时调试，全局变量可以添加到表达式窗口，并可在程序持续运行期间读取/写入。通过所列文件中的调试程序建立连接。

- Serial_Cmd_Monitor.c
- Serial_Cmd_Monitor.h
- Serial_Cmd_HAL.c
- Serial_Cmd_HAL.h

虽然这里列出了四个文件，但应用程序只需要两个函数。一个是初始化中调用的“SerialCmd_init()”，另一个是 BareMetal 后台循环或 RTOS 低优先级任务中调用的“SerialCmd_read()”。此部分重点介绍如何创建 UART 连接和如何在 CCS 中启动实时调试。

3.2.1 确认 CCS 特性

如果 CCS 版本早于 11.1，建议检查以下 CCS 驱动程序文件。Cortex_R5 的配置需要类似于图 3-2。如果缺少任一行，必须如图 3-2 所示添加行。至于行的内容，需要在目标配置文件中更新 COM 端口和波特率，此文件包含在下一步中。

- ccs\ccs_base\common\targetdb\drivers\gti_uart_driver.xml

```
<isa Type="Cortex_R5" ProcID="0x75803400">
    <driver file="../../../DebugServer/drivers/XPCOMToGTIAdapter.dvr">
        <property Type="stringfield" Value="COM14" id="COM Port" />
        <property Type="stringfield" Value="9600" id="Baud Rate" />
        <property Type="hiddenfield" Value="Little Endian" id="Endianness" />
        <property Type="hiddenfield" Value="32" id="Word Size Page 0" />
        <property Type="hiddenfield" Value="8" id="Minimum Addressable Size Page 0" />
        <property Type="hiddenfield" Value="@ti.com/UARTMonitor;1" id="XPCOM Class ID" />
        <property Type="hiddenfield" Value="Flash DLL Delegate" id="TargetAccess" />
        <connectionType Type="UARTConnection"/>
    </driver>
</isa>
```

图 3-2. R5F 的 CCS GTI UART 驱动程序

3.2.2 创建目标配置文件

AM263x controlCARD 在一个 USB 端口中提供 JTAG 和 UART 端口。有关硬件连接的详细信息，请参阅 *AM263x 控制卡用户指南*。控制卡在一个 USB Micro-B 连接器中同时提供 JTAG 和 UART 调试端口。必须为调试端口创建目标配置文件。图 3-3 至图 3-10 以屏幕截图形式提供分步指南。简而言之，创建一个目标配置文件，然后为其配置 JTAG 和 UART。图 3-10 中的 UART COM 端口需要与 JTAG 探头应用/用户 UART 的 PC 设备管理器 COM 端口匹配。图 3-10 中的波特率需要与下一步中配置的 SoC UART 波特率一致。

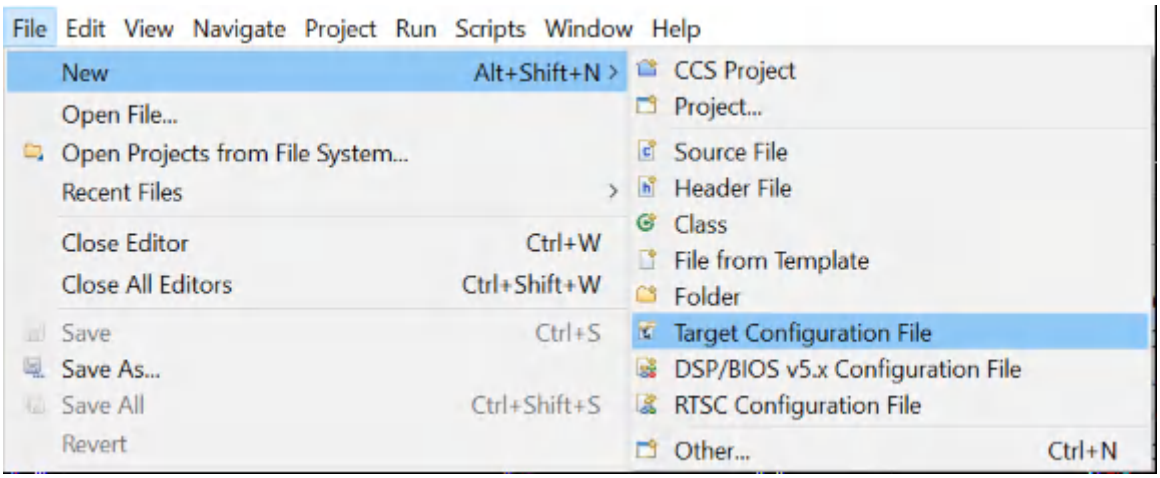

图 3-3. 创建新目标配置文件

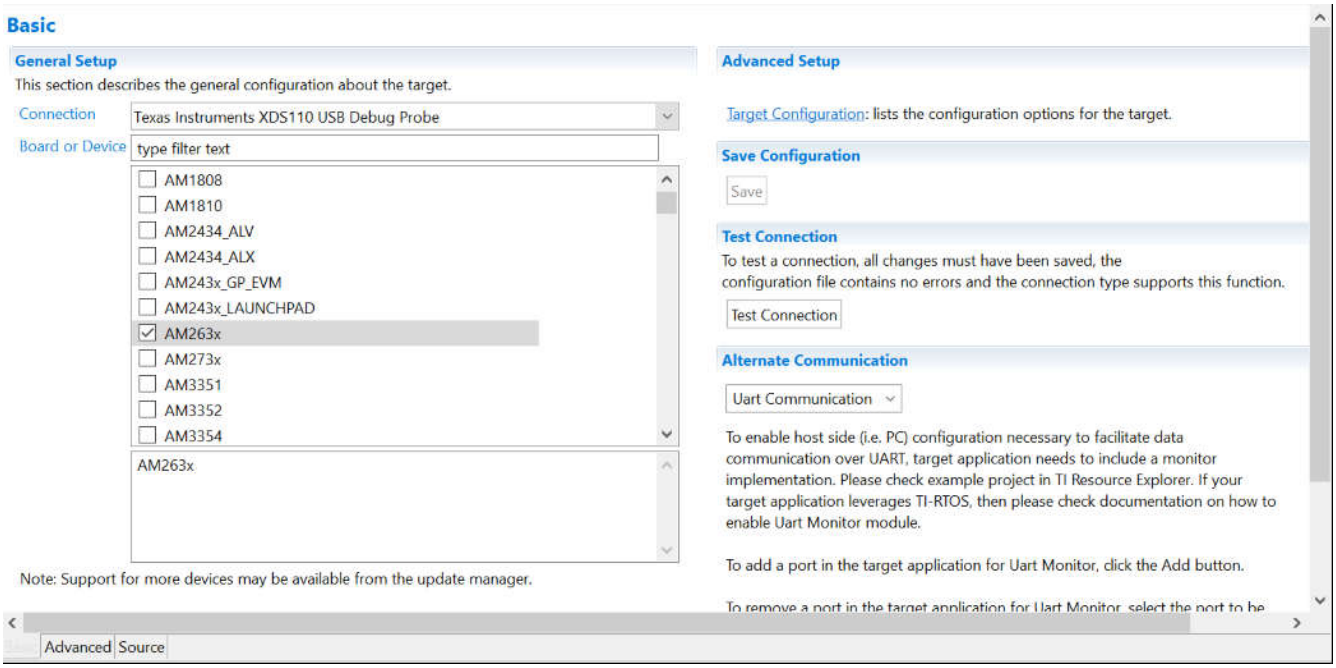

图 3-4. 选择 JTAG 连接和设备

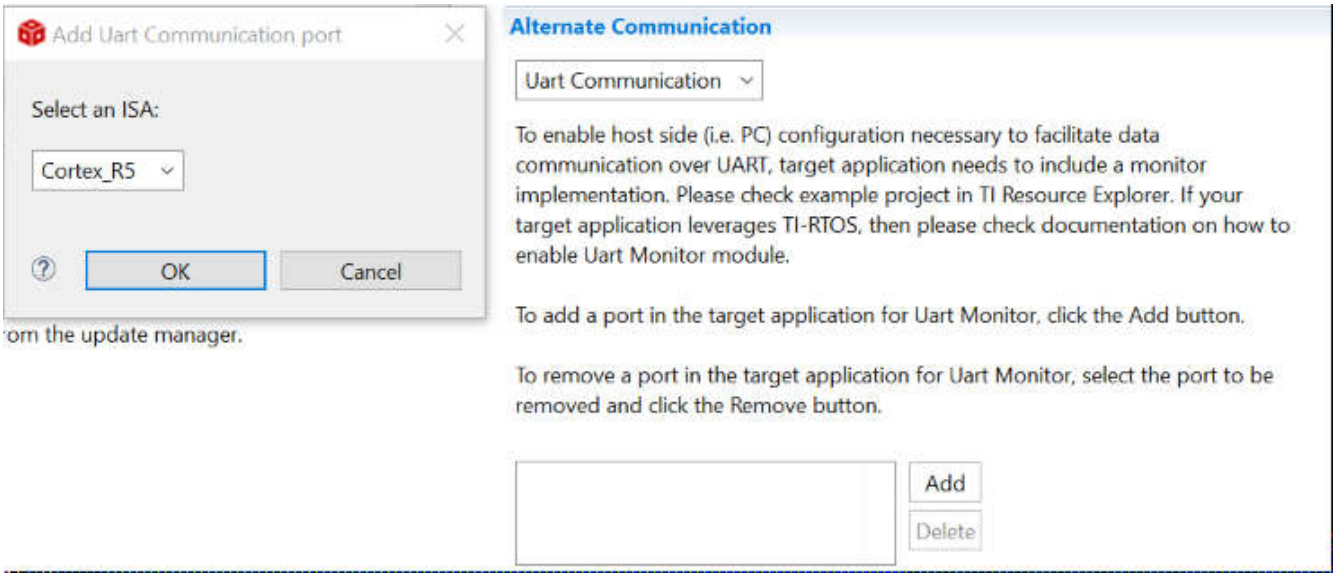

图 3-5. 添加 UART 通信端口

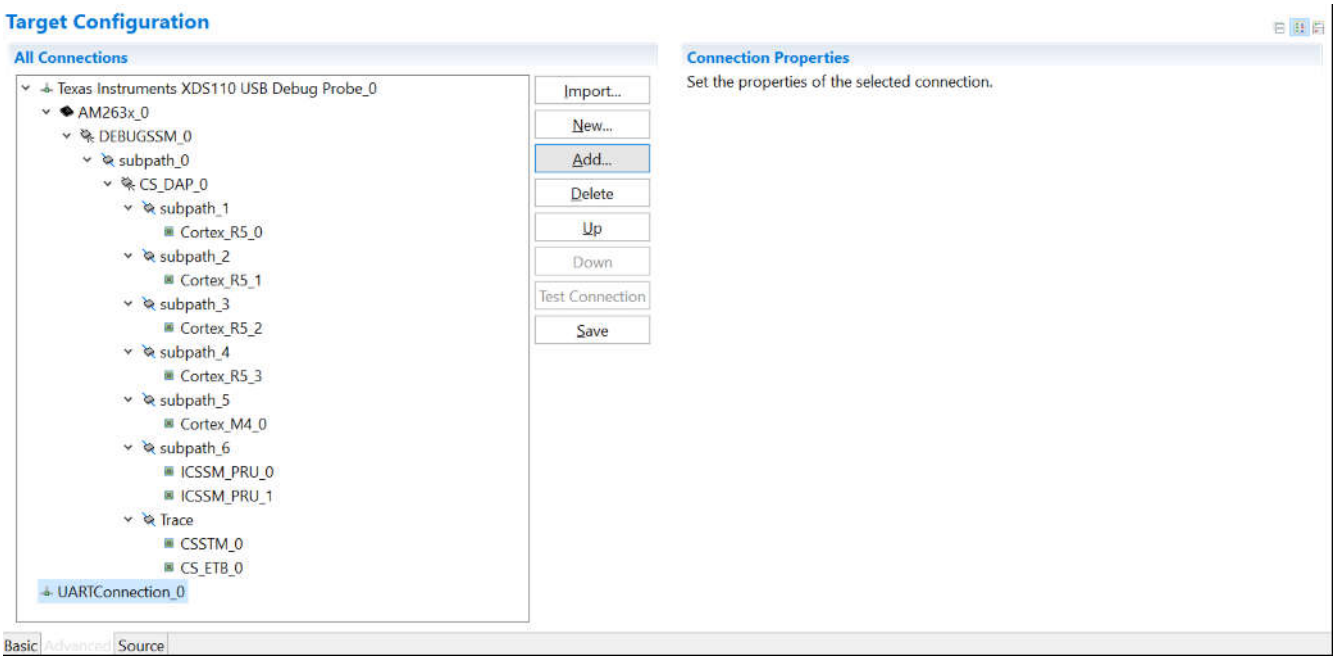

图 3-6. 打开高级目标配置

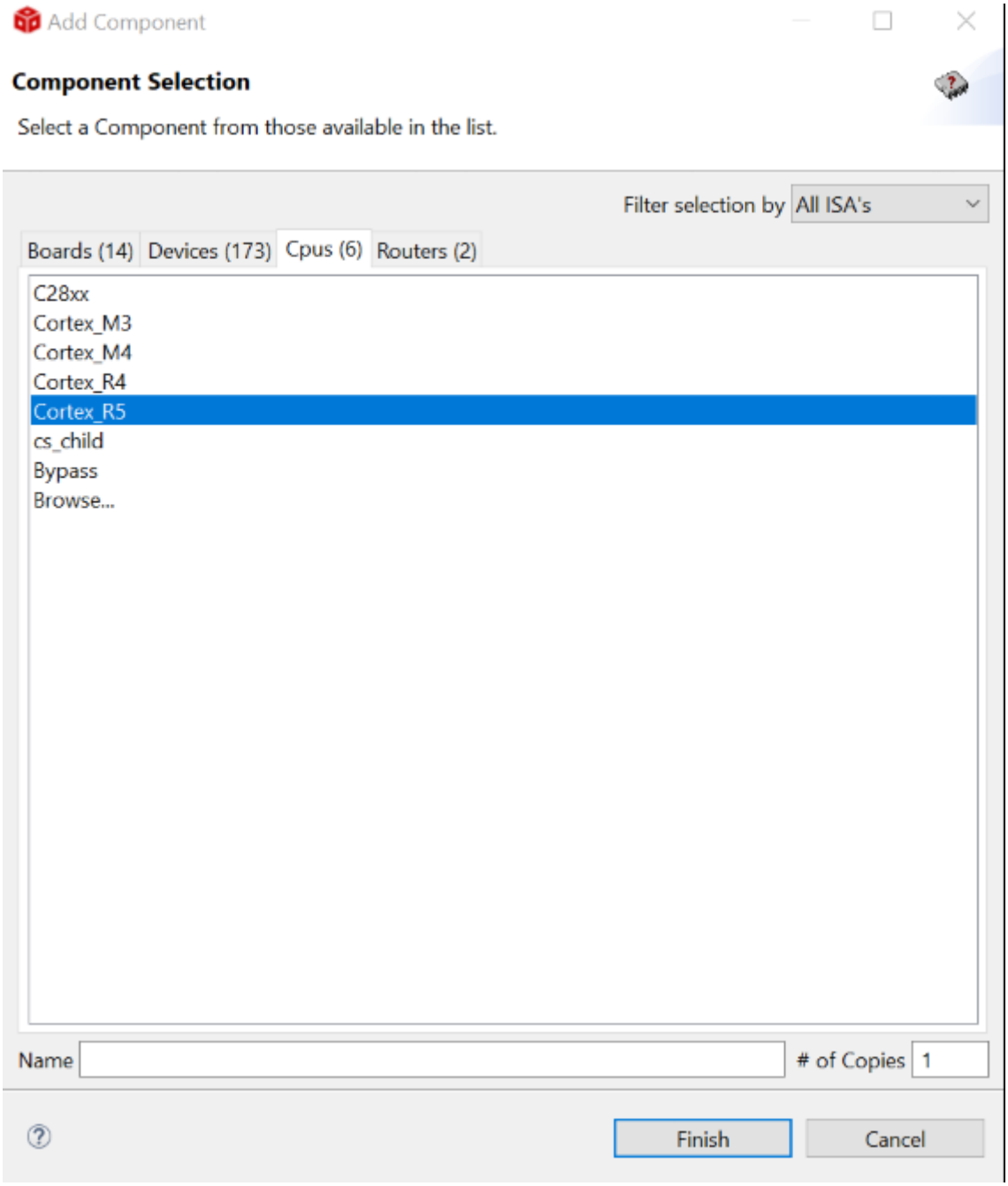

图 3-7. 添加元件

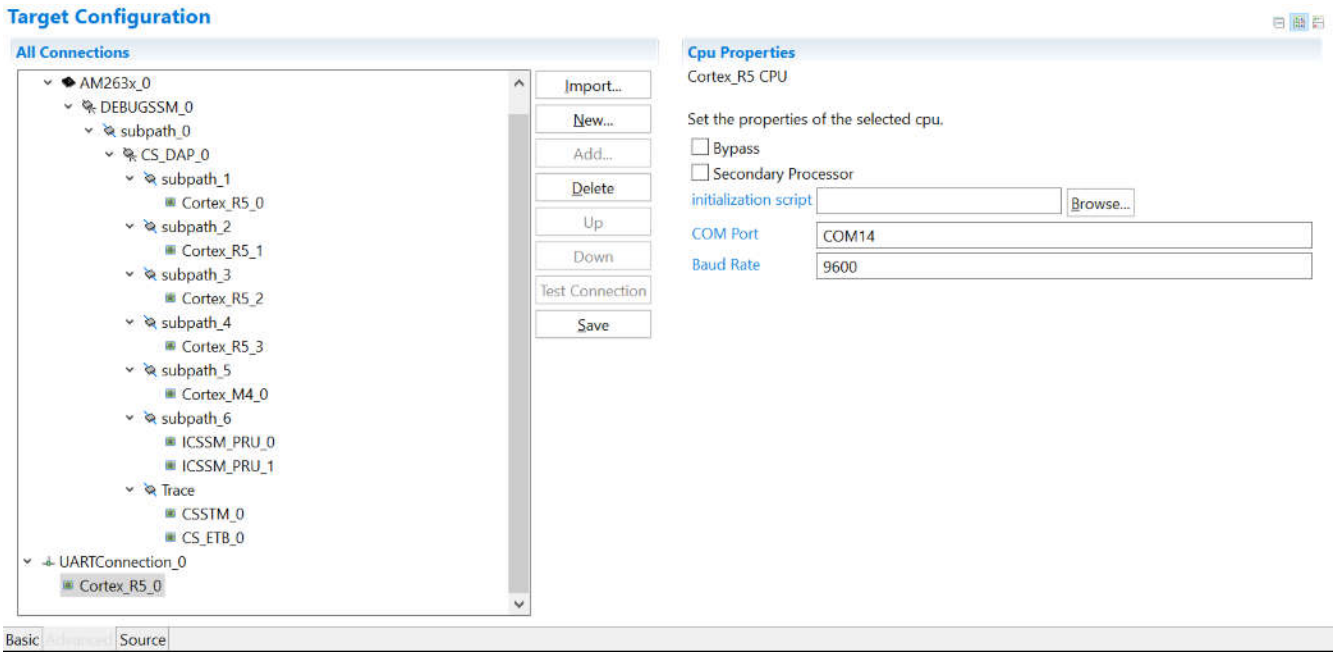

图 3-8. 选择 CPU 属性

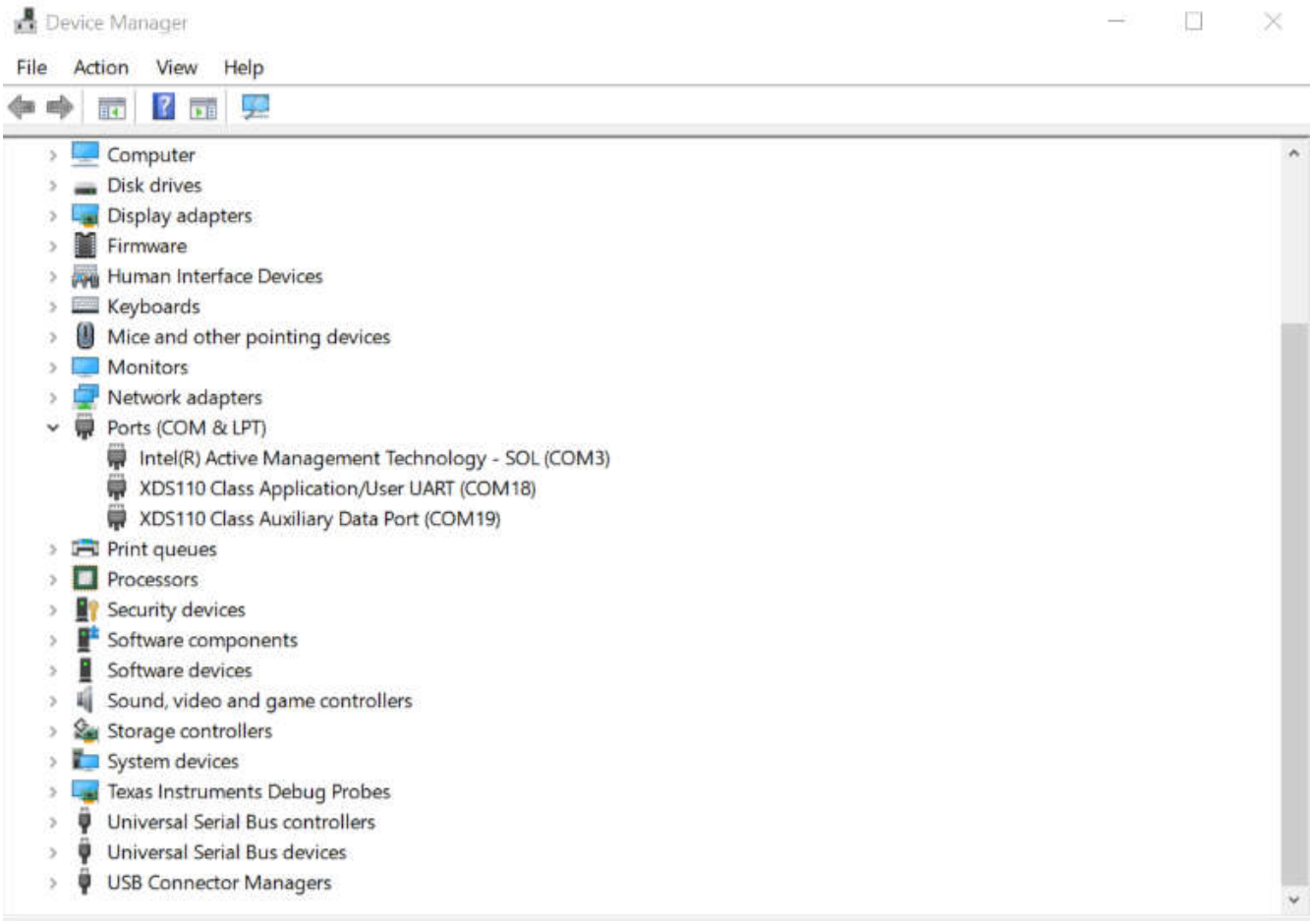

图 3-9. 查找 XDS110 UART COM 端口

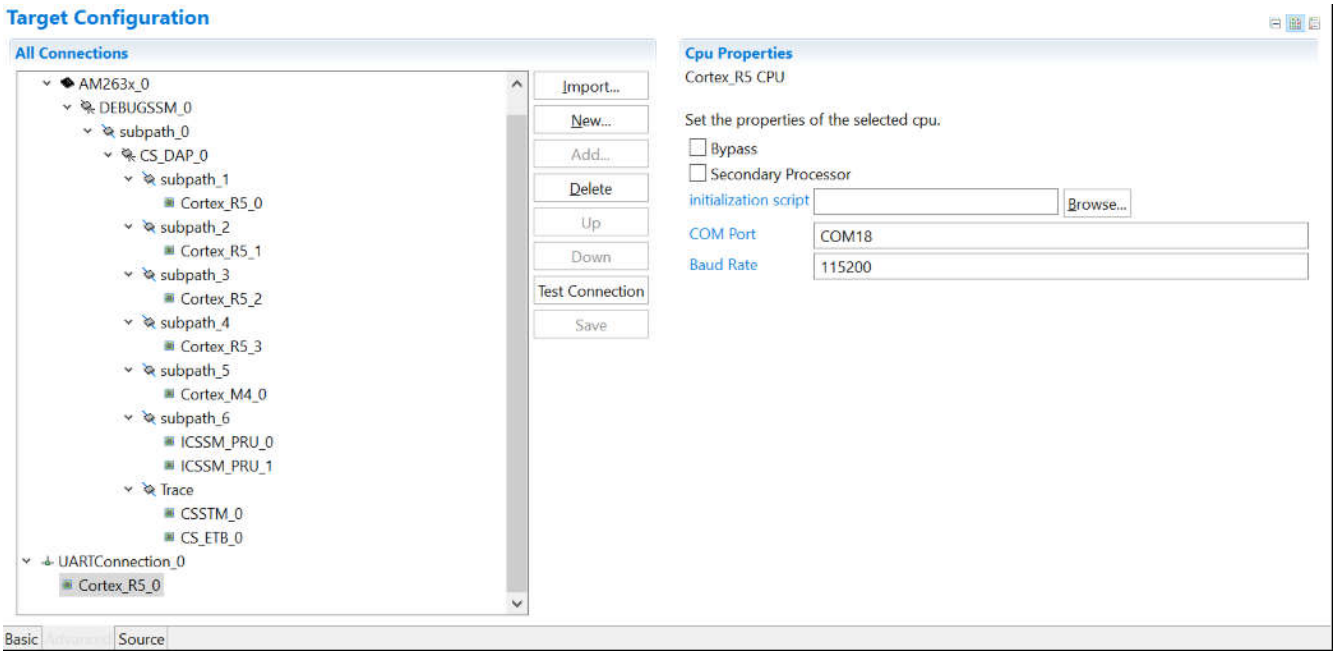

图 3-10. 在高级目标配置中更新 CPU 属性

3.2.3 添加串行命令监视器软件

使用 UART0 作为调试接口的方法有多种，分别是调试日志和串行命令监视器。调试日志是一个内置工具，位于 SDK 的驱动程序移植层。正如串行命令监视器，它的函数必须在中断回调之外。它是一个实现字符串输入和输出的方便工具。但是，输入和输出只通过 UART 控制台。CCS 中没有表达式窗口和图形之类的内置 GUI。建议在图 3-11 所示的“Debug Log”中禁用 UART0，并为图 3-12 所示的串行命令监视器配置 UART0 实例。Sysconfig 中的 UART 名称“CONFIG_UART_CONSOLE”与“Serial_Cmd_HAL.c”中的 handle 名称匹配，因此不必修改初始化和后台循环所需的两个函数。它们可以如图 3-13 所示直接插入。

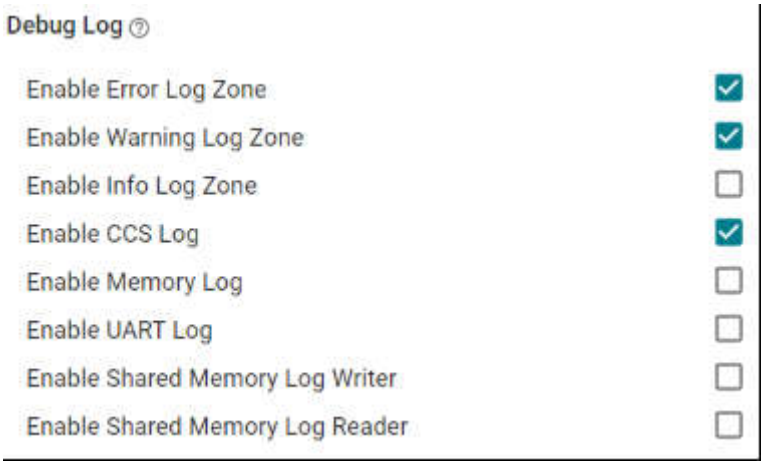

图 3-11. 在调试日志中禁用 UART 日志

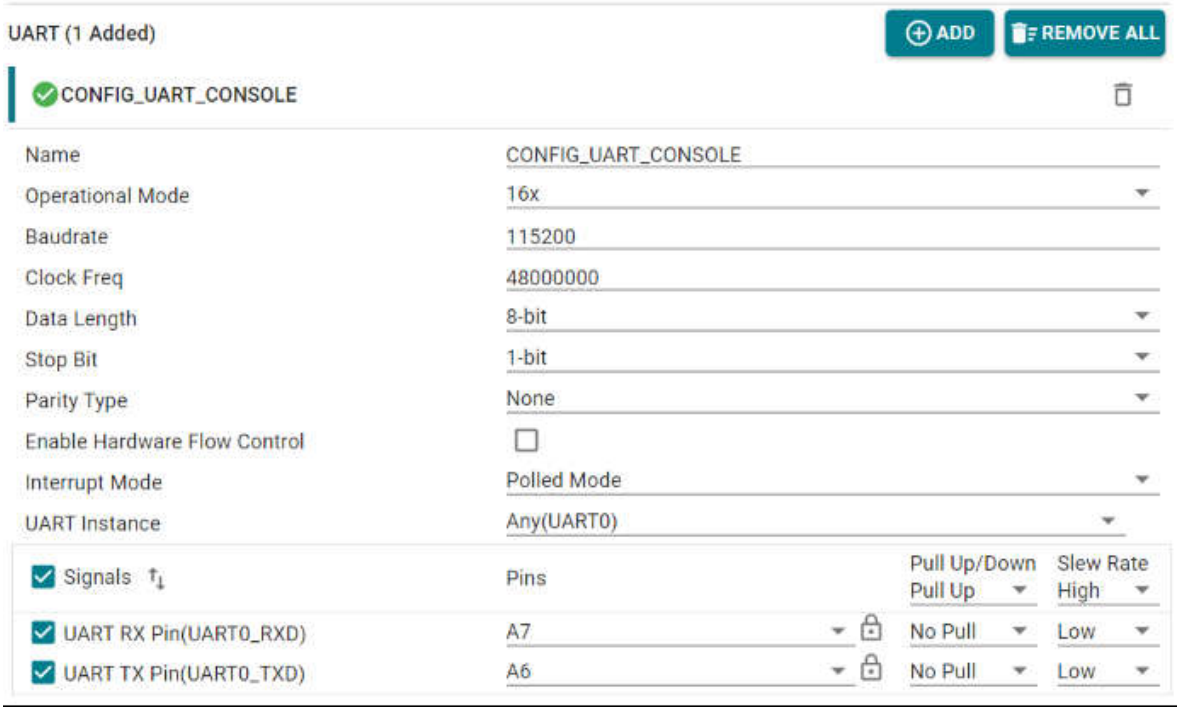

图 3-12. 配置 UART0 实例

```
void trinv_main(void *args)
{
    /* Open drivers to open the UART driver for console */
    Drivers_open();
    Board_driversOpen();

    SerialCmd_init();

    trinv_init();
    DebugP_log("trinv init done \r\n");
    FOC_init();
    DebugP_log("foc init done \r\n");
    FOC_cal();
    DebugP_log("foc cal done \r\n");
    FOC_run();
    DebugP_log("foc run done \r\n");

    while (gFlag){

        SerialCmd_read();

        gLoopTicker+=1;
        if (gLoopTicker>=100)
        {
            gLoopTicker=0;
        }
    }
    Board_driversClose();
    Drivers_close();
}
```

图 3-13. 添加串行监视器函数调用

3.2.4 启动实时调试

在构建程序之后，调试窗口应该会打开，其中包含在节 3.2.2 中创建的目标配置文件。如果创建的目标配置文件尚未打开，可以按照图 3-14 并查看“Target Configuration”窗口的“User Defined”文件夹找到它。创建的目标配置文件应该在“User Defined”文件夹下。右键点击该文件后，显示一个菜单，里面有一个“Launch Selected Configuration”选项，如图 3-15 所示。然后，调试窗口将显示。在许多 CCS 教程中可以找到连接目标、加载映像和通过 JTAG 运行的步骤。在连接到 UART 之前，处理器必须连续运行。由于 UART 连接以程序的连续运行为基础，停止串行命令监视器程序运行的断点、暂停、终止或任何其他事件将打断 UART 连接并冻结 CCS。有时，习惯在这些特性可用时使用它们。建立在使用 UART 连接时断开与目标的 JTAG 连接，如图 3-16 所示。当处理器运行时，可直接选择 UART 连接 → Run → Load → Load Symbols，建立 UART 连接，如图 3-17 所示。

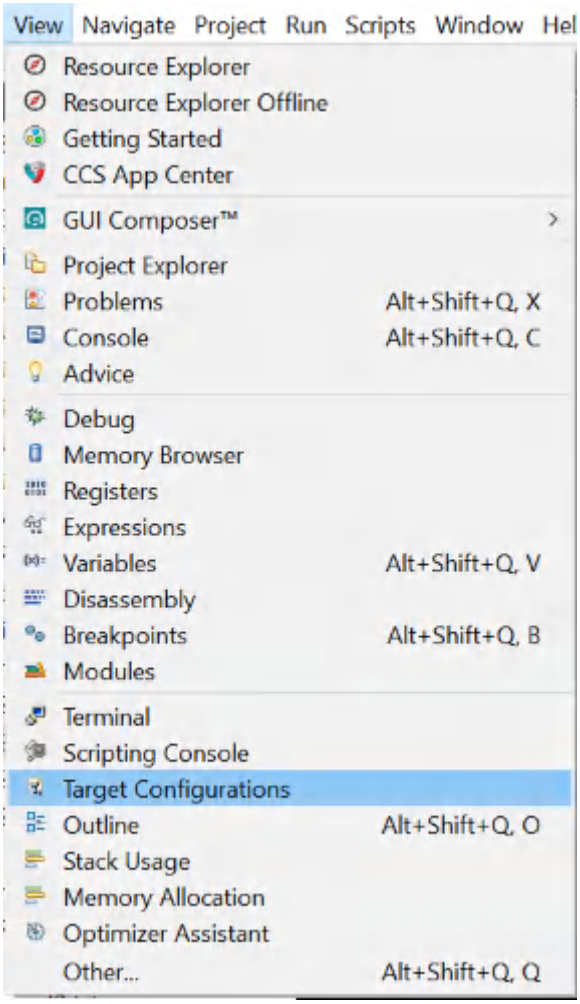

图 3-14. 找到目标配置文件

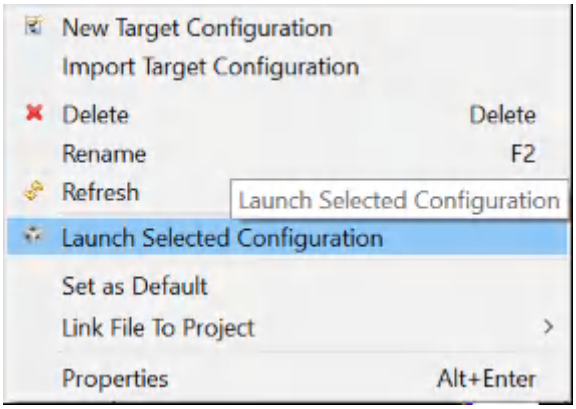

图 3-15. 启动所选配置

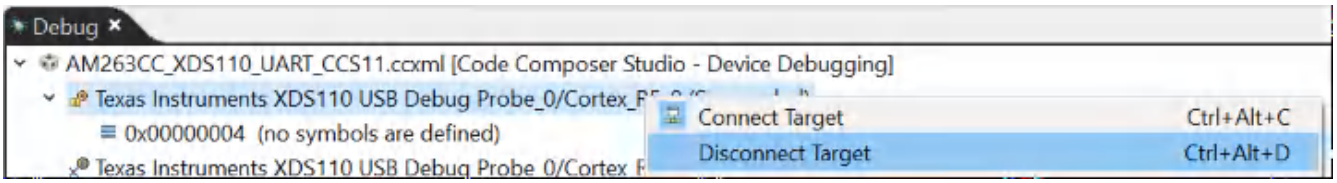

图 3-16. 断开 JTAG 连接

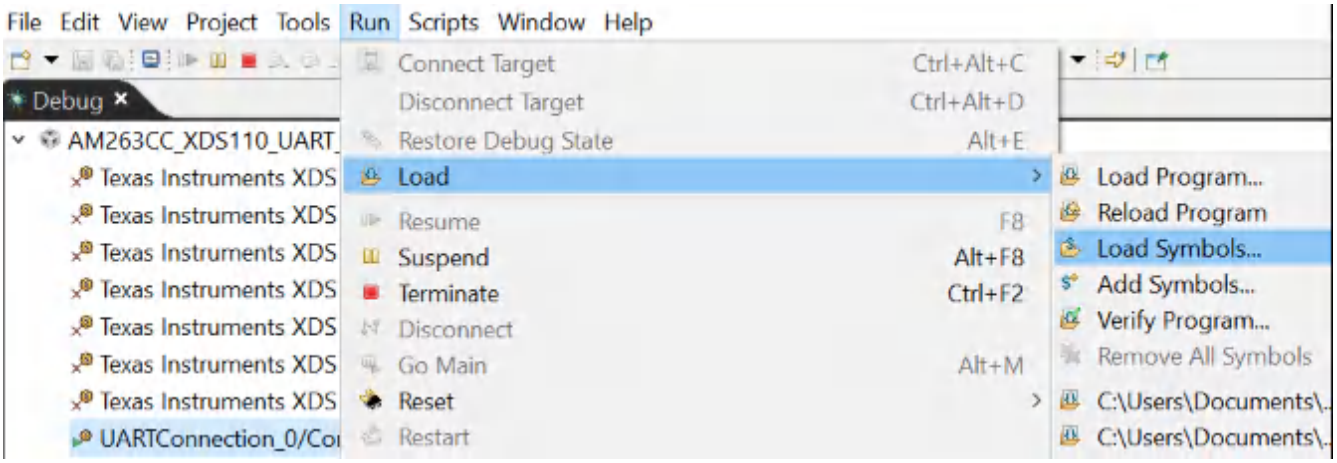

图 3-17. 建立 UART 连接

3.3 运行代码

电机控制软件可以针对不同的测试模式进行配置，以实现增量软件测试。例如，如果既未定义 CLOSED_CURRENT_LOOP 符号，也未定义 CLOSED_SPEED_LOOP 符号，则 Vd 和 Vq 基准值直接被设置为默认测试值。该模式可用于测试 PWM 设置和开环逆变器运行。同样，如果仅定义 CLOSED_CURRENT_LOOP，而不定义 CLOSED_SPEED_LOOP，则可以直接设置 id、iq 基准。在这种情况下，可以通过设置斜坡发生器频率来设置电机转速。trinv_setting.h 文件中提供了用户电机类型和相应的参数。可以通过更改 USER_MOTOR 宏的值来更改默认设置。如果用户打算使用预定义电机列表之外的电机，则用户可以复制使用的模板。

3.3.1 工程设置

将工程导入 CCS，并选择适当的构建配置。右键点击“Project Explorer”内的工程并选择“Rebuild Project”。确认“Console”窗格显示工程构建时没有任何错误。

成功完成构建后，选中 tidm_02014 工程，转至“Run”→“Debug”，或点击工具栏上的“Debug”按钮。工程默认情况下将使用工程中的 AM263x.ccxml 文件启动调试会话。AM263x.ccxml 配置为使用 TMDSCNCD263 控制卡电路板上的德州仪器 (TI) XDS110 USB 调试探针。

点击“Debug”后，CCS 将自动连接至目标，将输出文件载入器件内并更改为“CCS Debug”视图。程序应该在 main() 的开始处暂停。

点击“Expressions”窗口工具栏中的“Continuous Refresh”按钮，告知 CCS 以 CCS 调试首选项中定义的速率持续更新数据。

3.3.2 运行应用程序

转至“Run”→“Resume”或点击工具栏中的“Resume”按钮来运行代码。该工程现在可以运行并且变量显示在“Expressions”窗口中。检查以下各项，确认应用和硬件设置能够正常工作：

- 栅极驱动器板上的绿色电源 LED 必须亮起。如果栅极驱动器在无故障的情况下初始化，则红色 nFault LED 均不会亮起。可以通过 tripFlagDMC.fault.UCC5880_status 变量检查栅极驱动器初始化状态。
- 类似地，tripFlagDMC 结构中的其他变量显示其他故障的状态。如果未设置故障标志，那么要运行测试电机，可以将 runMotor 设置为 RUN_MOTOR。您的变量需要类似于图中所示内容。
- 如果未检测到故障，则 motor1.isrCount 必须持续递增。
- 检查电机逆变器板的校准偏移。电机相电流检测值的偏移值必须大约等于 ADC 满量程电流的一半。
- 也可以使用示波器探测电机驱动器的 PWM 输出。

您可以首先点击工具栏上的“Suspend”按钮或选择“Target”→“Suspend”来暂停 CPU。要从头开始重新运行应用，请点击“CPU Reset”工具栏按钮或点击“Run”→“Reset”→“CPU Reset”，然后点击“Restart”按钮或“Run”→“Restart”来重置控制器。您可以点击“Terminate”按钮，或点击“Run”→“Terminate”来关闭 CCS 调试会话。程序将暂停，并断开 CCS 与控制器的连接。

请注意，无需每次更改代码时都终止调试会话。您可以转到“Run”→“Load”→“Load Program...”（如果是同一个文件，请选择“Reload Program...”）。如果 CCS 检测到您已重建可执行文件，还会自动进行提示，询问您是否需要重新加载该可执行文件。

3.4 从 ADC 采样并通过 CCS 读取样本

按照前面部分的配置，ADC 在 PWM0 零计数点对信号采样，并在 PWM0 每个周期的选定转换结束点创建 ADC INT。此部分介绍如何寄存 INT、读取样本并在图形中绘制它们。

3.4.1 寄存和启用中断

INT 配置后，必须用以下行寄存和启用中断。第 1 行使用默认值初始化硬件中断参数。第 2 行用上一部分中配置的数字更新中断数。在“cslr_intr_r5fss0_core0.h”中可找到有关宏定义的更多详情。如果使用其他集群或内核，在 SDK 中可找到相似的文件。第 3 行将回调函数分配给硬件中断。它只接收函数地址。第 4 行使用更新的硬件中断参数构造硬件中断。如果有错误，第 5 行会通过 JTAG 将故障消息发送给 CCS 控制台。如果中断状态不清除，它会在第 6 行清除该状态之后开始运行。为了保持中断持续运行，必须在每次执行回调函数时调用第 6 行。

```
1. HwiP_Params_init(&hwiPrms);
2. hwiPrms.intNum = CSLR_R5FSS0_CORE0_CONTROLSS_INTRXBAR0_OUT_0;
3. hwiPrms.callback = &FOCrun_ISR;
4. status = HwiP_construct(&gAdcHwiObject, &hwiPrms);
5. DebugP_assert(SystemP_SUCCESS == status);
6. ADC_clearInterruptStatus(CONFIG_ADC4_BASE_ADDR,ADC_INT_NUMBER1);
```

3.4.2 添加日志代码，以固定速率读取图中样本

由于 SoC 和 CCS 之间的通信用时是不确定的，所以有必要以给定的采样率从中断记录值。这对于通过图形窗口观察数据很关键。一个简单的日志已经实现。16 个指针可用于 16 个全局变量。硬件中断开始之前，指针需要分配给全局变量或 NULL。需要调用以下命令行。在指针分配后调用第 1 行。每次中断完成所有计算后调用第 2 行。第 2 行函数中实现了一个名为 gLogScaler 的量程。它表示两个日志记录点之间跳过的中断数。通过设置全局变量，记录数据的频率可以低于或等于中断频率。

1. LoopLog_init();
2. LoopLog_run();

3.4.3 在表达式和图形窗口中读取 ADC 样本

以下第 1 行至第 8 行读取三相电流、旋转变压器 sin/cos 和直流母线电压的 ADC 样本。SDK API 包装到一个文件中的 Macros。*按住 Ctrl 键简单地左键点击* 几次变量名称，有助于找到定义它的位置。第 9 行和第 10 行提供了示例。ADC_readResult 要读取 ADC 结果，ADC_readPPBResult 要在后处理块之后读取 ADC 结果。有关后处理块的详细信息，请参阅技术参考手册。

1. motor1.I_abc_A[0] = (float32_t)IFBU_PPB;
2. motor1.I_abc_A[1] = (float32_t)IFBV_PPB;
3. motor1.I_abc_A[2] = (float32_t)IFBW_PPB;
4. resolver1.sin_samples[0] = (float32_t)R_SIN1;
5. resolver1.sin_samples[1] = (float32_t)R_SIN2;
6. resolver1.cos_samples[0] = (float32_t)R_COS1;
7. resolver1.cos_samples[1] = (float32_t)R_COS2;
8. motor1.dcBus_V = (float32_t)VDC_EVT;
9. ADC_readResult(CSL_CONTROLSS_ADC1_RESULT_U_BASE, ADC_SOC_NUMBER0)
10. ADC_readPPBResult(CSL_CONTROLSS_ADC1_RESULT_U_BASE, ADC_PPB_NUMBER1)

为显示在图形窗口中读取和绘制 ADC 的示例，日志指针连接到了三相电流，并调用日志函数。通过在表达式窗口中右键点击 gLog_CH[7] 并选择图形，空载时的 A 相电流将绘制到图形窗口中，如图 3-18 所示。在这种情况下，A 相电流指向 gLog_CH[7]，如以下列表中所示。它可以分配给任何日志通道。要将 gLog_CH 添加到表达式窗口，只需右键点击它并添加到监视表达式。如需更多详情，请查看 CCS 教程。

1. gLog_ptr[7] = &motor1.I_abc_A[0];
2. gLog_ptr[8] = &motor1.I_abc_A[1];
3. gLog_ptr[9] = &motor1.I_abc_A[2];

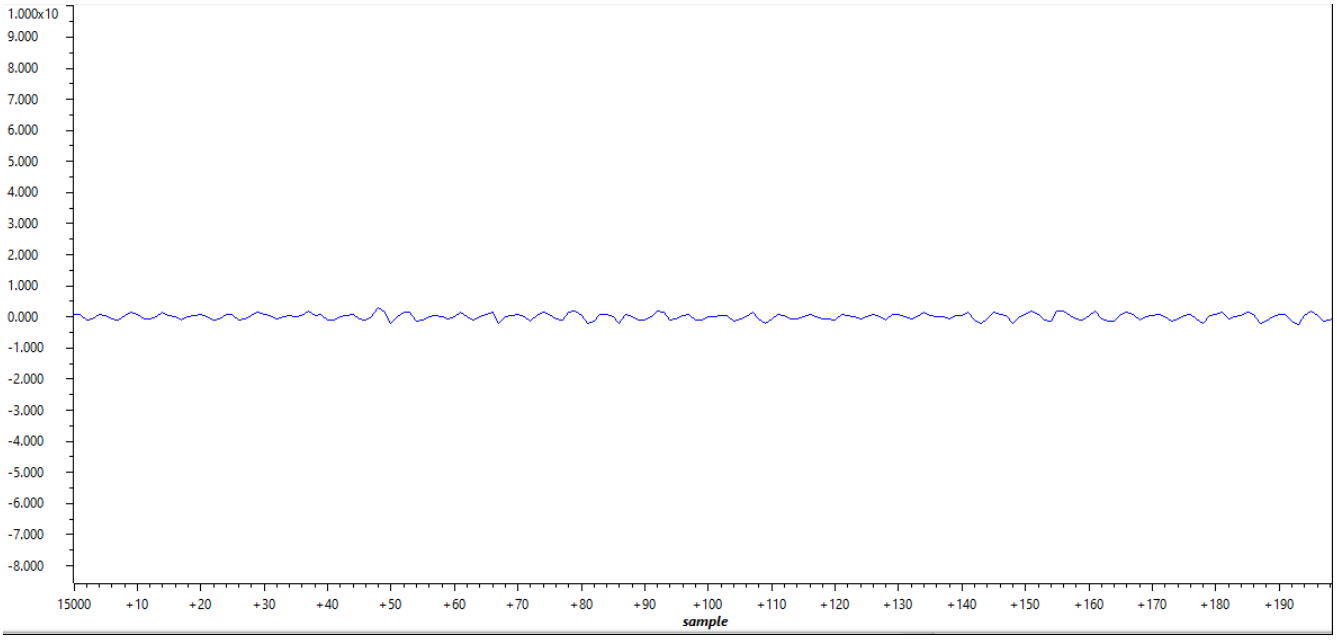

图 3-18. 绘制的空载 A 相电流

在低电压下启动系统期间，建议校准直流母线电压反馈的失调电压，因为测量范围较大，可能会存在 10V - 20V 的失调电压。

3.5 生成空间矢量 PWM 和在开环中驱动电机

空间矢量 PWM 是一种经典的电机控制方法。此部分仅显示高级函数，而不像教科书那样详细。高级函数可轻松替换为 C28 库或其他控制器库中的相似函数。关键在于，减少使用本地变量并将全局变量分配给 TCM，以获得确定性的执行时间。

3.5.1 设置 SVPWM 发生器输入

SVPWM 发生器的输入为 Vd 和 Vq。需要调用以下行来赋值。Motor1 是存储在 TCM 中的结构。在程序文件中可找到有关其定义的更多详情。*按住 Ctrl 键简单地左键点击* 几次变量名称，有助于追踪它的定义位置。代码逻辑与 TIDM-02014 的 C28 程序相同。Vd 和 Vq 是实际值，而不是标幺值。

```
1.  motor1.Vout_dq_V[0] = VdTesting;
2.  motor1.Vout_dq_V[1] = VqTesting;
```

以下行生成电机转速和电机角度。第 1 行至第 4 行设置斜坡控制器 rc1 和斜坡发生器 rg1。SpdRef 是 0 和 1 之间的标幺值。在第 5 行和第 6 行中，生成的 omega 和 theta 分配给 motor1。第 7 行将 theta 限制在 0 到 TWO_PI 的范围内。文件中定义了 TWO_PI 值。按住 Ctrl 键简单地左键点击几次变量名称，有助于追踪它的定义位置。值得注意的是，在启动硬件中断之前，需要相应地初始化 rc1、rg1 和 motor1。

```
1.  rc1.TargetValue = SpdRef;
2.  rampControl(&rc1);
3.  rg1.Freq = rc1.SetpointValue;
4.  rampGen(&rg1);
5.  motor1.omega_e = rg1.Freq * BASE_FREQ * TWO_PI;
6.  motor1.theta_e = rg1.Out * TWO_PI;
7.  theta_limiter(&(motor1.theta_e));
```

接下来的几行将输入送到 SVPWM 发生器，并保持标幺值输出。第 1 行将输入限制在一定范围内。第 2 行是 Park 逆变换。在 CMSIS DSP 库和其他地方可找到相似的函数。角度信息已经包含在 motor1 的结构中。第 3 行是 SVPWM 发生器。逻辑与 TIDM-02014 的 C28 程序相同。以前的 C28 库中还有其他实现方式。值得注意的是，此版本中有实际值到标幺值的转换。第 4 行将标幺值输出限制在一定范围内。

1. dq_limiter_run(&motor1);
2. ipark_run(&motor1);
3. SVGEN_run(&motor1, &pwm1);
4. PWM_clamp(&pwm1);

SVPWM 生成后，标幺值输出通过下面第 1 行中的函数传递到 EPWM 计数器比较。第 2 行提供有关设置 EPWM0 计数器比较的详情。EPWM_setCounterCompareValue 是设置计数器比较值的 SDK API 的名称。对于向上/向下模式或中心线模式，此处计算该值。

1. TRINV_HAL_setPwmOutput(&pwm1);
2. EPWM_setCounterCompareValue(CONFIG_EPWM0_BASE_ADDR, EPWM_COUNTER_COMPARE_A, (uint16_t)((pwm->inv_half_prd * pwm->Vabc_pu[0]) +pwm->inv_half_prd));

3.5.2 在图形窗口中读取 SVPWM 占空比

就像节 3.4.3 中在图形窗口绘制 ADC 样本一样，以下标幺值输出需要连接到日志指针。

1. gLog_ptr[4] = &pwm1.Vabc_pu[0];
2. gLog_ptr[5] = &pwm1.Vabc_pu[1];
3. gLog_ptr[6] = &pwm1.Vabc_pu[2];

图 3-19 中呈现了 A 相占空比的绘制图形。

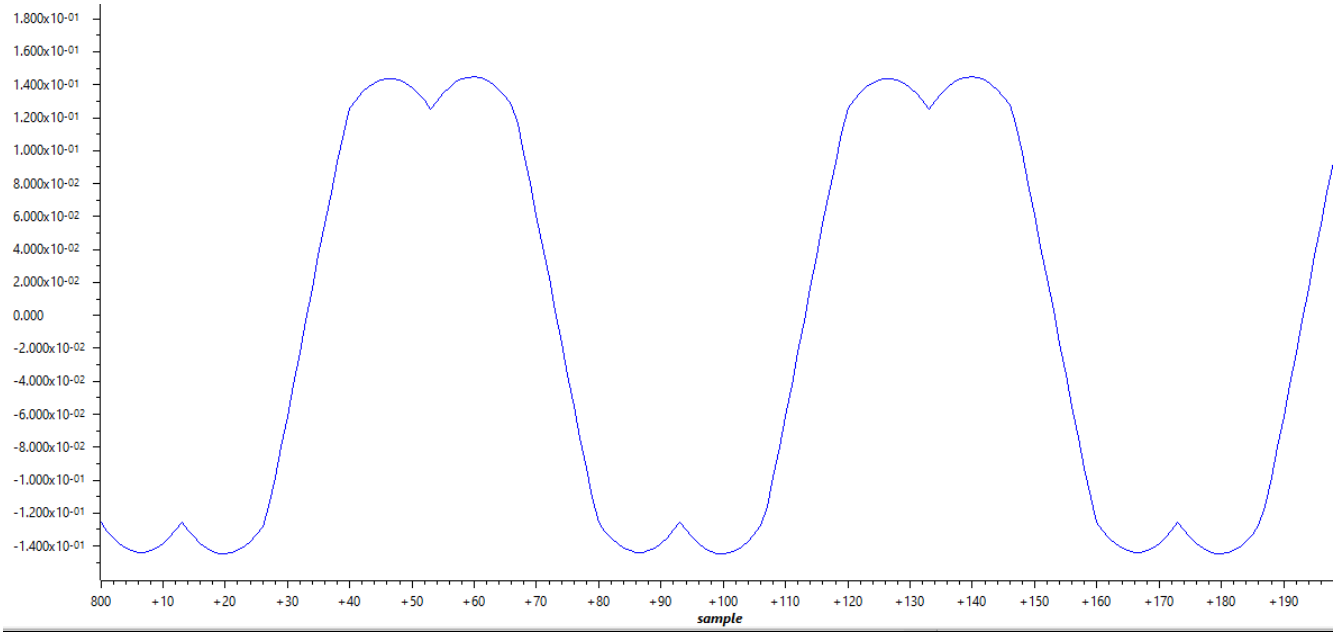

图 3-19. A 相占空比

3.5.3 逆变器上电并在开环中旋转电机

逆变器上电后，便可以在开环中旋转电机。建议起始速度为低速。标幺值全局变量 SpdRef 可设置在 0.01 左右。几个标志控制所列示例程序的执行。“runMotor”是“gTFlag_MockTheta”和“gTFlag_SpdDemo”的栅极。如果“runMotor”为“FALSE”，不会发送速度命令。“gTFlag_MockTheta”将在开环和电流闭环中使用仿真角度和速度。“gTFlag_SpdDemo”将提供速度命令，以演示速度闭环。“gTFlag_MockTheta”和“gTFlag_SpdDemo”不应该同时为“TRUE”。必须使用“runMotor”停止电机，然后在“gTFlag_MockTheta”和“gTFlag_SpdDemo”之间切换。当“gTFlag_MockVdq”为“TRUE”时，在 SVPWM 生成前来自程序的 Vd 和 Vq 将通过从表达式窗口中手动输入进行覆盖。当“gTFlag_MockId”或“gTFlag_MockIq”为“TRUE”时，电流环路控制器输入端的电流值将通过从表达式窗口手动输入进行替换。

- runMotor
- gTFlag_MockTheta
- gTFlag_MockVdq
- gTFlag_MockId
- gTFlag_MockIq
- gTFlag_SpdDemo

在此部分中，将“gTFlag_MockTheta”和“gTFlag_MockVdq”设置为“TRUE”后，“runMotor”可更改为“TRUE”。正确选择“SpdRef”、“VdTesting”和“VqTesting”后，值得注意的是，Vd 和 Vq 是实际值，而不是标幺值。如节 3.4.3 所述，可读取 A 相电流。图 3-20 中绘制了它。在接通低频交流电流后，电机应该开始旋转。否则，建议检查电机、逆变器和控制卡。逆变器硬件详情请参阅 TIDM-02014。用户指南中应提供控制卡详情。

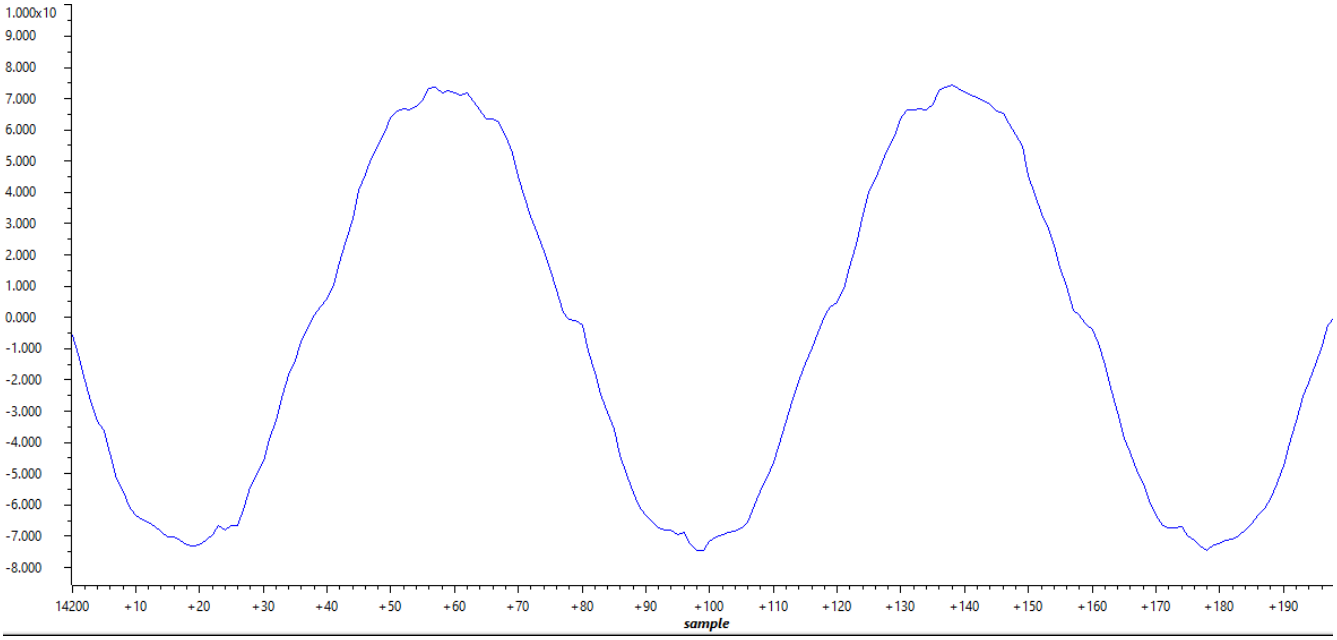

图 3-20. A 相电流开环

3.6 以模拟速度闭合电流环路

此部分要以模拟速度闭合电流环路。模拟速度由节 3.5.1 中提出的斜坡创建。手动分配 Id 和 Iq 基准。

3.6.1 添加变换和读取开环中的 Id-Iq

要闭合电流环路，需要添加以下变换。第 1 行是 Clark 变换，第 2 行是 Park 变换。在 CMSIS DSP 库和其他地方可找到相似的函数。角度信息已经包含在 motor1 的结构中。这里的实现与 TIDM-02014 的 C28 程序相似。

1. clarke_run(&motor1);
2. park_run(&motor1);

理想情况下，Id 和 Iq 在开环稳态运行时接近恒定值。大多数情况下，在表达式窗口中读取它们并不难。如果需要图形视图，可按照节 3.4.3 将它们添加到图形视图。以下是日志指针的设置。第 1 行是 Id，第 2 行是 Iq。绘制的 Id 和 Iq 分别位于图 3-21 和图 3-22 中。

1. gLog_ptr[10] = &motor1.I_dq_A[0];
2. gLog_ptr[11] = &motor1.I_dq_A[1];

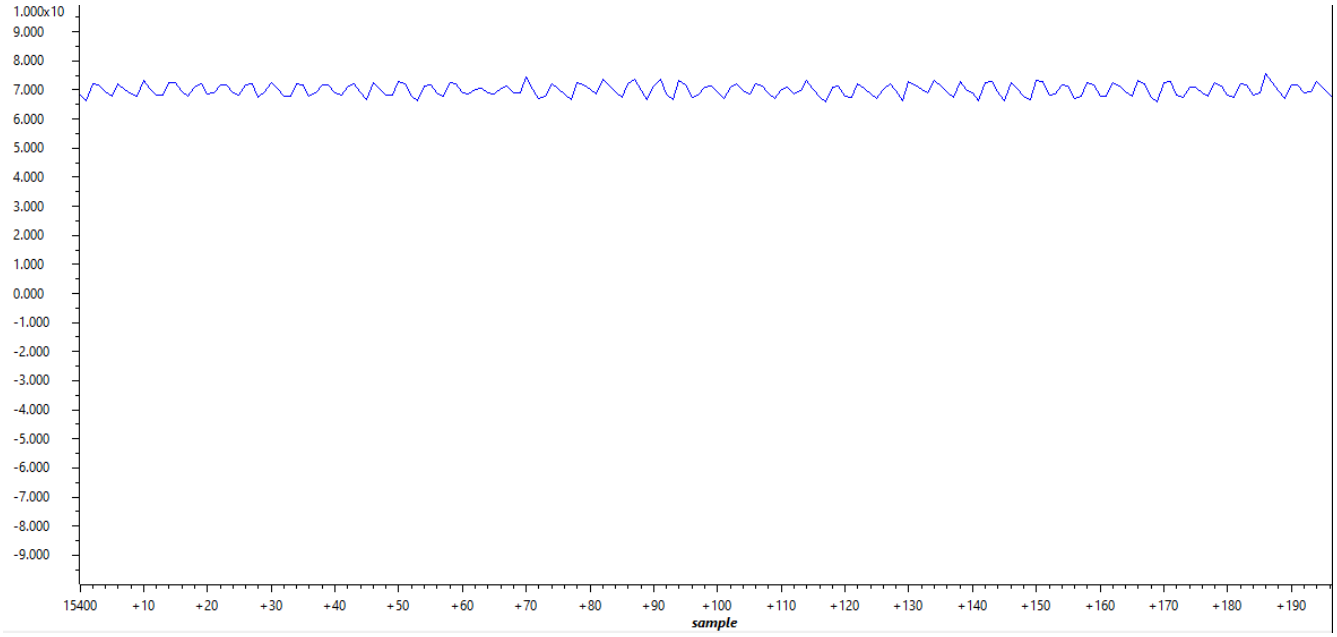

图 3-21. 开环 Id

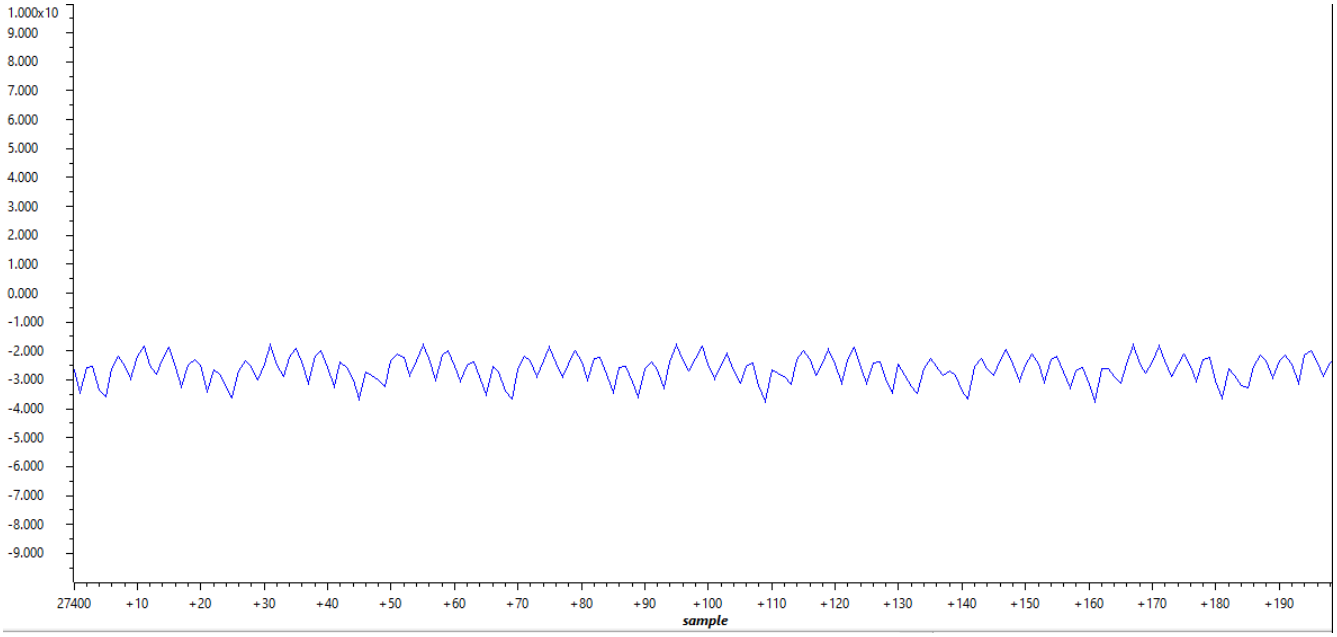

图 3-22. 开环 Iq

3.6.2 添加控制器，以闭合电流环路

必须添加 PI 控制器，以闭合电流环路。这里 PI 控制器的实现与 TIDM-02014 的 C28 程序相似。CMSIS DSP 库和其他地方有不同的实现。这里无意介绍 PI 控制器的详情。主要信息是确保控制器位于 TCM 中。从 TCM 运行对于确定性的执行时间至关重要。否则，在高速缓存和 OCRAM 之间交换内容或在 OCRAM 中直接运行将占用大

量的时间。第 1 行中提供了属性设置示例。第 2 行和第 3 行中显示了一个函数调用的另一个示例。属性设置必须将链接命令文件和存储器保护单元配置匹配起来。更多详情请参阅技术参考手册。

```
1.  __attribute__ ((section(".tcmb_code"))) static inline float32_t PI_run_series(PI_Obj * pi)
2.  motor1.pi_id.fbackValue = motor1.I_dq_A[0];
3.  motor1.Vout_dq_V[0] = PI_run_series(&(motor1.pi_id));
```

3.6.3 读取 Id-Iq，以闭合电流环路

理想情况下，Id 和 Iq 在闭合电流环路稳态运行期间是恒定值。大多数情况下，在表达式窗口中读取它们并不难。如果需要图形视图，可按照节 3.4.3 将它们添加到图形视图。以下是日志指针的设置。第 1 行是 Id，第 2 行是 Iq。绘制的 Id 和 Iq 分别位于图 3-23 和图 3-24 中。

```
1.  gLog_ptr[10] = &motor1.I_dq_A[0];
2.  gLog_ptr[11] = &motor1.I_dq_A[1];
```

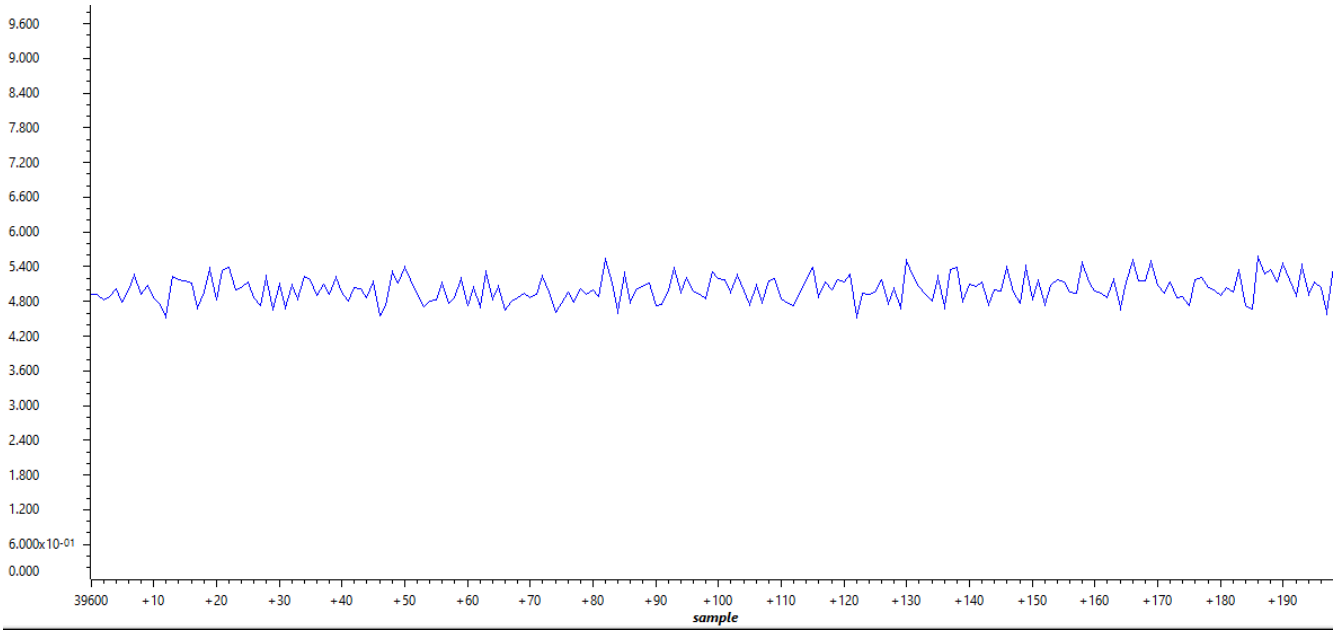

图 3-23. 闭环 Id

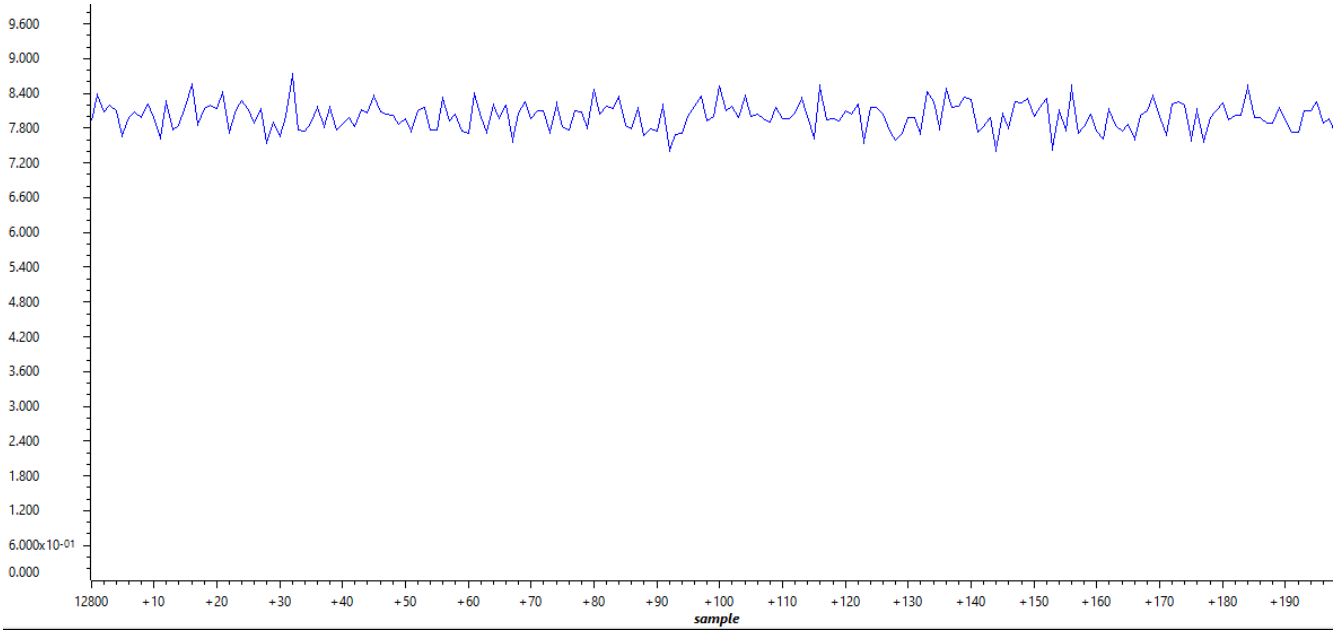

图 3-24. 闭环 Iq

3.7 添加软件旋转变压器数字转换器

软件旋转变压器数字转换器的实现方法是通过 DAC 和 DMA 发送激励信号，并通过 ADC 读取正弦和余弦反馈。图 3-25 总结了配置概况。在本文中，PWM X 是 EPWM0，PWM Y 是 EPWM7。10 kHz 的 EPWM0 触发 A 相电源开关和同步源。20 kHz 的 EPWM7 专用于通过 DAC 触发 EDMA0，以产生旋转变压器激励。ADC 转换开始 (SOC) 是在 EPWM0 计数为零时触发的。ADC4 的转换结束 (EOC) 是包含 FOC 环路的 ADC INT1 的源。此部分将介绍软件旋转变压器的功能和读数。

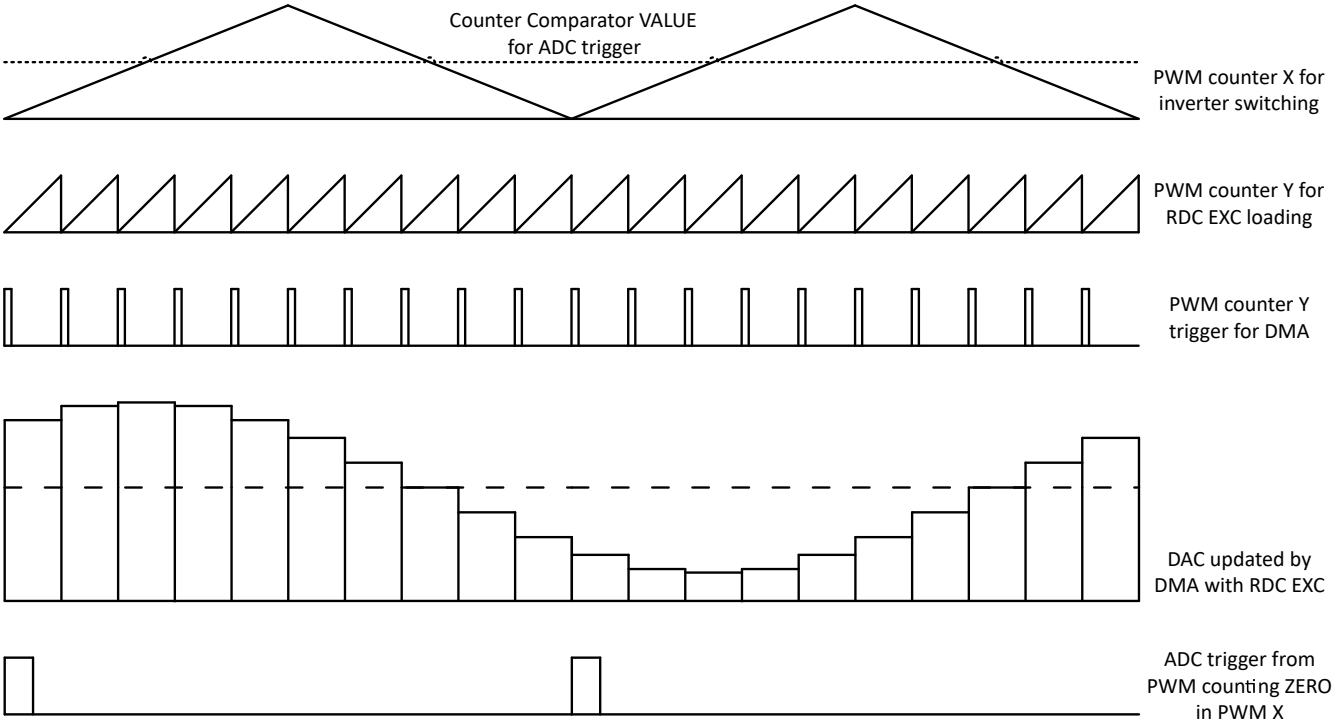

图 3-25. 软件旋转变压器同步和激励

3.7.1 为旋转变压器硬件生成激励

EDMA 配置为每当发生 EPWM7 SOCA 事件时传输数据。为指定关于传输的更多细节，下面第 1 行中定义了函数，以初始化 EDMA，包括数据位置、数据大小和 DAC 值寄存器的位置。为更新详情，下面第 2 行中定义了函数，以更改 EDMA 输入表和 DAC 激励输出的内容。

- uint16_t RDCexc_start(uint16_t *table, uint16_t table_size, EDMA_Handle dma_handle, uint32_t dma_ch, uint32_t dac_base)
- uint16_t RDCexc_update(uint16_t *table, uint16_t table_size, EDMA_Handle dma_handle, uint32_t dma_ch, uint32_t dac_base)

RDCexc_start 需要在控制环路运行前调用，RDCexc_update 可以在控制环路运行期间调用。RDCexc_update 的目的是减少激励相位与采样时间之间的移位。通过在调用 RDCexc_update 之前偏移输入表的指针来调整激励相位。以下行中提供了两个函数的示例。

1. RDCexc_start(gRDCtable_ptr,20,gEdmaHandle[0],DMA_TRIG_XBAR_EDMA_MODULE_0,CONFIG_DAC0_BASE_ADDR);
2. RDCexc_update(gRDCtable_ptr,20,gEdmaHandle[0],DMA_TRIG_XBAR_EDMA_MODULE_0,CONFIG_DAC0_BASE_ADDR);

全局作用域中表的一个元素连接到 gRDCtable_ptr。表中包含多个完整正弦波周期的点。该元素位于表的中间，以便在工作期间向右或向左偏移指针，以调整激励相位。一个周期中有 20 个点。EDMA 输入表的长度为 20。EDMA 句柄、XBAR 信息和 DAC 地址可在配置文件中找到。

3.7.2 添加旋转变压器软件

软件旋转变压器的实现与 TIDM-02014 中的 C28 相似。两个函数如下所列。第 1 行用于初始化旋转变压器结构，第 2 行用于计算角度和速度。初始化结构后，需要根据旋转变压器硬件的特性更新实现细节。有关详细信息，请参见结构定义。

1. static inline void resolver_init(Resolver_t *resolver);
2. static inline void resolver_run(Resolver_t *resolver);

请注意，实现 sin/cos 等数学函数的方法有很多。在此设计中，旋转变压器锁相环输入的 sin/cos 函数从“resolver_run”中移出，因此更容易找到所有数学函数并进行更改。以下第 1 行和第 2 行必须在上面的第 2 行之前调用。这里使用标准 C 库函数举例。

1. resolver1.res_theta0_sin = sinf(resolver1.res_theta0);
2. resolver1.res_theta0_cos = cosf(resolver1.res_theta0);

3.7.3 读取旋转变压器软件输出

在初始化时，软件旋转变压器的输出使用指针传递给全局变量，如下面的第 1 行和第 2 行所示。由于在旋转过程中电机角度总是变化，所以若不以固定采样频率在图形窗口中绘制它，便没有意义。第 3 行和第 4 行将旋转变压器角度和速度的地址传递给两个日志指针。节 3.4.3 中所述的日志函数每当中断指定数量便记录它们。

1. resolver1.resolver_theta = &resolver_theta;
2. resolver1.resolver_omega = &resolver_omega;
3. gLog_ptr[12] = &resolver_theta;
4. gLog_ptr[13] = &resolver_omega;

4 代码迁移的简要指南

此部分首先概述了 AM263x SoC 架构和 SDK 资源。请牢记可用的资源，以下两部分总结了从 AM24 和 C28 迁移代码的一些技巧。

4.1 SDK 资源概览

表 4-1 总结了 SDK 资源概览。在 SDK 安装目录下的“README_FIRST_AM263X.html”中可找到更多详情。默认情况下，目录路径类似于“C:\ti\mcu_plus_sdk_am263x_xx_xx_xx_xx”。“examples/”文件夹是对新手最重要的一个资源。它包括 AM24 匹配连接示例和 C28 控制示例。大多数情况下，在 CCS 中按住 Ctrl 键点击左键，可引导用户进入 API 声明头。由于继承自 C28 和 AM24，控件 API 头包括静态内联的声明和定义，而连接 API 头只显示声明。连接 API 定义必须位于源文件中。头文件和源文件都保存在“source/”文件夹中。如果 CCS 无法获得所需的详细信息，那么这些详细信息很可能保存在“source/”中的某个源文件中。另一个方法是查看“README_FIRST_AM263X.html”的“API Reference”。

表 4-1. SDK 目录结构

文件夹/文件	说明
${SDK_INSTALL_PATH}/	
README_FIRST_AM263X.html	在 Web 浏览器中打开此文件，以获取 SDK 用户指南
makefile	使用“make”构建整个 SDK 的出色 makefile
imports.mak	列出相关工具路径的出色 makefile
docs/	离线 HTML 文档
examples/	跨多个电路板、CPU、NO-RTOS、RTOS 的 AM263X 应用示例
source/	设备驱动程序、中间件库和 API
tools/	工具和实用程序，如 CCS 加载脚本、初始化脚本。
${SDK_INSTALL_PATH}/source/	
board/	电路板外围设备驱动程序
驱动器/	SOC 外围设备驱动程序
industrial_comms/	工业通信协议栈和工业协议 FW HAL (固件和硬件抽象层)
kernel/	这些环境的 NO RTOS 和 RTOS 内核与驱动程序移植层 (DPL)
${SDK_INSTALL_PATH}/examples/	
驱动器/	以 SOC 和电路板为中心的设备驱动程序示例。这些示例基于 NO-RTOS 和 RTOS。
empty/	将模板工程复制到工作区，并根据应用需求进行定制
industrial_comms/	EtherCAT 从属设备示例

4.2 从 C28 迁移代码

C28 与 AM263x 有相似的控制外设。但架构和连接外设完全不同。一般来说，与控制外设相关的程序可以在很少或没有修改的情况下进行迁移，而与 CPU、内存管理和连接外设相关的程序必须针对 AM263x 技术参考手册中的细节进行更新。

众所周知，直接操作寄存器在过去的 C28 程序中得到广泛应用。最近几年，从寄存器操作改为了 API 调用。从寄存器操作改为 API 调用可以简化较复杂的 MCU 的采用。但是，从寄存器用户转为 API 用户需要完成一些工作。对于 C28 和 AM263x，这项工作都是不可避免的。完成这项工作后，不难使用 AM263x 控制子系统，因为来自 ADC 和 PWM 等模块的概念非常相似。表 4-2 中提供了有关控制 API 相似性的一些示例。另外，AM263x SDK 还提供强大的 Sysconfig。它提供直观的系统配置用户接口。终端用户可以直接将他们对控制外设的想法应用到配置中，而无需担心 API 细节。控制环路中广泛使用的 API 已经在框架中提供并在节 3 中介绍。

表 4-2. API 定义相似性示例

API 函数	AM263x 定义	C28 定义
获取 ADC 结果	static inline uint16_t ADC_readResult (uint32_t resultBase, ADC_SOCNumber socNumber)	static inline uint16_t ADC_readResult (uint32_t resultBase, ADC_SOCNumber socNumber)
设置 PWM 占空比	static inline void EPWM_setCounterCompareValue (uint32_t base, EPWM_CounterCompareModule compModule, uint16_t compCount)	static inline void EPWM_setCounterCompareValue (uint32_t base, EPWM_CounterCompareModule compModule, uint16_t compCount)

此外，尽管二者有相似之处，但在 SDK 和名称相似的某些设计上也有些不同。如节 4.1 所示，AM263x 的 SDK 结构与 C28 的 SDK 非常不同。尽管它们都有相似的控制外设和相似的 API，但仍然必须了解 SDK 结构的不同，以便在开发过程中轻松查找细节。对于 XBAR 等某些特性，C28 和 AM263x 都用 XBAR 同步模块之间的操作，但 AM263x 中的 XBAR 远比 C28 中的 XBAR 更加强大。这也带来一项挑战，必须充分理解和正确配置它。C28 中的 XBAR 程序无法直接应用于 AM263x 工程。

4.3 从 AM24 迁移代码

AM24 与 AM263x 有相似的架构和连接外设。但控制外设完全不同。一般来说，与连接相关的程序可以在很少或没有修改的情况下进行迁移，而与控制外设相关的程序必须针对 AM263x 技术参考手册中的细节进行更新。至于 SDK，每个版本都是独特的。但是 AM24 和 AM263x 在驱动程序移植层的 API 几乎是相同的。不同之处请参阅“SOC 特定的设备驱动程序”。

另请注意，AM24 与 AM263x 尽管有相似之处，但在架构和连接方面也存在差异。例如，AM24 为工业以太网、千兆位工业通信子系统、更灵活的外部存储器扩展 DDR4 子系统提供更强大的支持。AM263x 中具备 100 兆位工业以太网和 16 位/32 位并行总线特性。与此类特性相关的程序必须重新设计，才能从 AM24 迁移到 AM263x。

另一点是，AM24 R5F 内核频率最高 800 MHz，而 AM263x R5F 内核频率为 400 MHz。在代码迁移期间，必须料想到并妥当处理执行时间的重大变化。务必确保执行时间保持在要求范围内。但是，对于牵引逆变器，考虑到 400MHz 和 800MHz 内核远高于经典的 MCU，因此大多数情况下，它们的执行时间应该不构成问题。

5 总结

作为电动汽车的核心，牵引逆变器是一个需要高 CPU 吞吐量和灵活控制外设的系统。本文档展示了 AM263x MCU 如何适应牵引逆变器参考设计的架构及其硬件和软件框架。然后，详细讨论了在 TIDM-02014 中使用 AM263x 的软件设置。文中介绍了使用 TMDSCNCD263 控制卡测试 TIDM-02014 参考设计的过程。本文档对缩短学习曲线并加快 AM263x 在牵引系统中的采用的工作做了进一步补充。

6 参考文献

- 德州仪器 (TI)，*经过 ASIL D 等级功能安全认证的高速牵引和双向直流/直流转换参考设计*。
- 德州仪器 (TI)，*AM263x Sitara™ 微控制器* 数据表。
- 德州仪器 (TI)，*AM263x Sitara 处理器技术参考手册*。
- 德州仪器 (TI)，*AM263x 控制卡用户指南*。

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