Application Note

使用 C2000™ 实时微控制器的 MATLAB® 和 Simulink® 基于模型的设计

Aditya Dholakia, Jayakarthigeyan Prabakar and Aditya Padmanabha

摘要

鉴于工业和汽车控制应用变得愈发复杂，使用 MathWorks® 的 Embedded Coder® 等代码生成工具将高层控制算法开发与特定于器件的低层驱动程序开发分离开来非常有用。随着代码生成工具的出现，对 C2000™ 等实时微控制器所生成代码的易用性、效率和性能进行评估变得至关重要。本应用手册详细介绍了如何使用适用于 C2000 的 MATLAB C2000 Microcontroller Blockset，其中涵盖了从入门到最佳实践和性能评估的各个方面。为了展示整个用例，本文选择了 eCompressor 参考设计 TIDM-02012 进行评估。

内容

插图清单

商标

1 引言

本应用手册重点介绍了如何通过在 MATLAB® 中生成基于模型的设计代码来实现性能优化。为帮助了解性能优化，本文选择了使用 MATLAB 基于模型的设计代码生成工具的 TIDM-02012 eCompressor 参考设计示例。本应用手册将讨论此参考设计的实现、优化和性能评估。

1.1 入门

启用基于模型的设计所需的软件工具集包括 TI 的 Code Composer Studio IDE、C2000Ware 软件开发套件 (SDK) 以及 MATLAB 工具（如 Simulink®、Embedded Coder、C2000 Microcontroller Blockset 和 Motor Control Blockset）。

下面给出了这些工具的下载和安装链接。

德州仪器 (TI) 工具：

- Code Composer Studio：CCSTUDIO IDE
- C2000Ware SDK：C2000WARE 软件开发套件 (SDK)
- C2000Ware MotorControl SDK：C2000WARE-MOTORCONTROL-SDK 软件开发套件 (SDK)

MathWorks 工具：

- MATLAB：下载 MATLAB、Simulink、Stateflow 和其他 MathWorks 产品 安装 Simulink、Embedded Coder、MATLAB Coder、Simulink Coder、C2000 Microcontroller Blockset 和 Motor Control Blockset。
- C2000 Microcontroller Blockset：C2000 Microcontroller Blockset - MATLAB

与 MATLAB 集成的 C2000 Microcontroller Blockset 的安装指南可以在该 Blockset 的安装页面中找到。点击此处，可查看 C2000 Microcontroller Blockset 的培训视频。

备注

由于 Embedded Coder、C2000 Microcontroller Blockset 以及 C2000 的编译器会在每个版本中进一步更新，以优化代码生成，因此建议使用可用的最新工具来尽可能获得最佳的代码生成效率和性能。

2 eCompressor 基于模型的设计

TIDM-02012 基于模型的设计示例是 eCompressor 硬件，其目标应用包括空调、供暖和牵引驱动器。随着汽车行业向电动汽车过渡，汽车的供暖和制冷系统使用永磁同步电机 (PMSM) 来驱动 eCompressor。C2000 电机控制 SDK 提供的参考设计演示了在不使用位置传感器的情况下如何使用磁场定向控制 (FOC) 来控制 eCompressor 电机。本应用手册的重点不在于了解 eCompressor 硬件的功能，而是了解为给定目标应用实现基于模型的设计的流程。

使用手动编写代码开发应用的传统方式当前面临着诸多挑战，包括编码专业知识要求、编译终端应用所用的时间和可调试性。这些挑战可以通过按照基于模型的设计工作流程编译应用来解决。基于模型的设计让您即使对微控制器的编码知识有限，也能轻松进行开，同时还能缩短开发时间，并且采用了图形化方法，让您能够直观地查看应用代码和流程。模型可以随时进行仿真，从而即时查看 Simulink 中的系统行为。

有关混合动力汽车/电动汽车 HVAC eCompressor 高压电机控制参考设计的详细信息，请参阅 TIDM-02012 产品页面和*混合动力汽车和电动汽车 HVAC eCompressor 高压电机控制参考设计*

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

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

保存所有警告和说明以供将来参考。

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

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

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

c. EVM 准备就绪后，根据需要将 EVM 通电。

警告

EVM 通电后，请勿触摸 EVM 或其电路，它们可能存在高压，会造成电击危险。

3. 人身安全
   a. 穿戴人员防护装备（例如乳胶手套或具有侧护板的安全眼镜）或将 EVM 放置于带有联锁装置的透明塑料箱中，避免意外接触。

安全使用限制条件：

勿将 EVM 作为整体或部分生产单元使用。

2.2 方框图

图 2-1 显示了 eCompressor 电机控制参考设计基于模型的设计。

名为 *TMS320F280039C* 的 Simulink 子系统包含 C2000 器件驱动器块和控制块。该子系统的 MCU 源代码可以使用 C2000 Microcontroller Blockset 和 Embedded Coder 自动生成。名为*逆变器和电机 – 受控体模型* 的块包含逆变器和电机的仿真受控体模型，其配置参数与实际参考设计硬件中的参数相似。您需要根据具体应用的硬件规格更新模型。

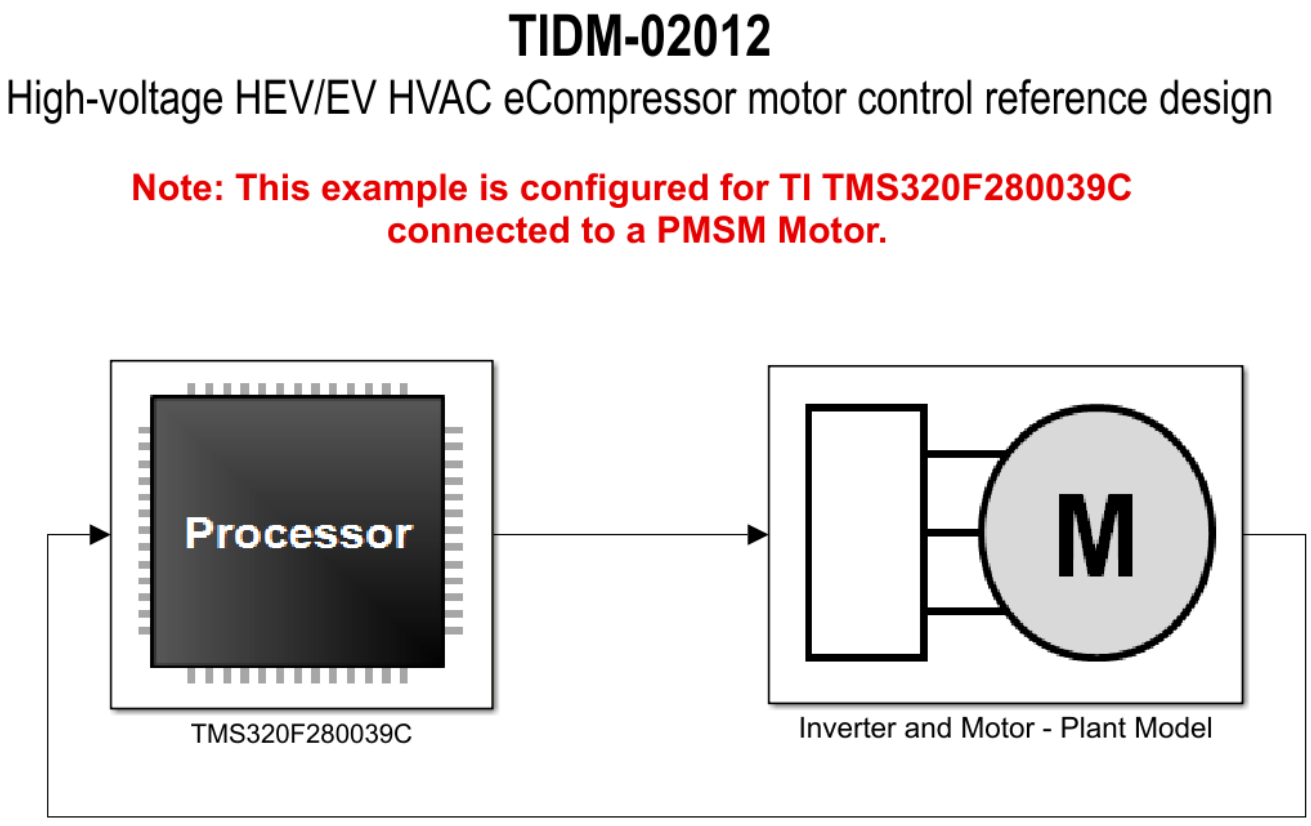

图 2-1. eCompressor 基于模型的设计

进入 *TMS320F280039C* 块后，完整控制循环算法的实现如图 2-2 所示。

备注

处理器 TMS320F280039C 模型仅包含那些通过 Embedded Coder 生成代码的块。受控体模型仅用于仿真不会生成应用代码的逆变器和电机。

该操作主要分为 3 个部分。

- 传感和通信
- 控制环路
- 通过 PWM 进行的占空比控制

电机相电流 I_a、I_b 和 I_c 的检测是使用模型中传感器驱动器块子系统内提供的 ADC 完成的，如图 2-2 所示。中断模块包含针对 2 个中断的配置：ADC 和 SCI。ADC 中断会以 *15kHz* 的更快速率执行来实现控制操作，而 SCI 中断会以更低的速率每 *0.1 秒*运行一次，以读取电机的参考速度和开/关控制变化。中断配置位于 *TMS320F280039C* 块内的*传感器驱动器块* 和 *SCI 中断块* 中。

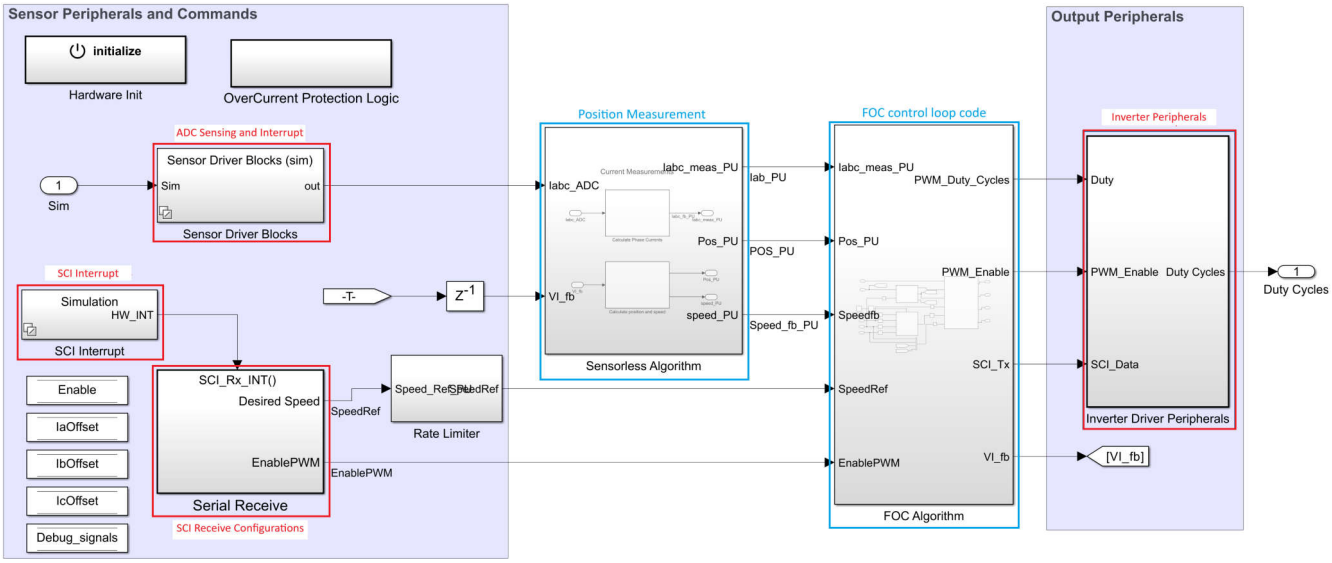

图 2-2. TMS320F280039C 模型块

除了要在器件上执行的代码外，为了确保正确实现控制，还需要满足一些其他要求，例如启用/禁用电机、配置电机的速度以及从实际系统中读回参数，比如速度、直流和正交轴电流（I_d 和 I_q）、电机相电流（I_a 和 I_b）以及计算得出的位置。要读取这些值，板载 SCI 模块用于将选定的数据发送到控制主机。用户可以使用控制主机模型完全控制电机的运行。如图 2-2 所示，接收器的 SCI 模块在*串行* 接收块内实现。

控制主机上的 SCI 模块在 TIDM_02012_control_host.six 文件内的*串行通信*块中实现，如图 2-3 所示。

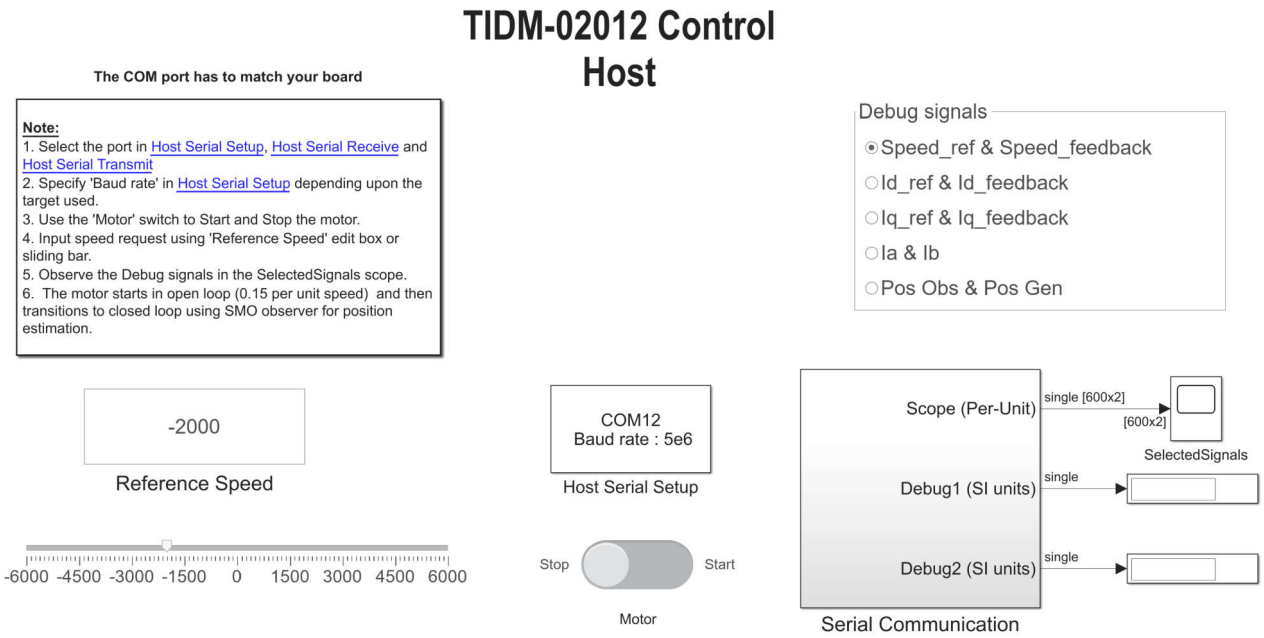

图 2-3. TIDM-02012 控制主机模块

图 2-4 中显示了模型工作原理的图形表示。模型在硬件上运行，而主机系统运行控制主机应用程序。

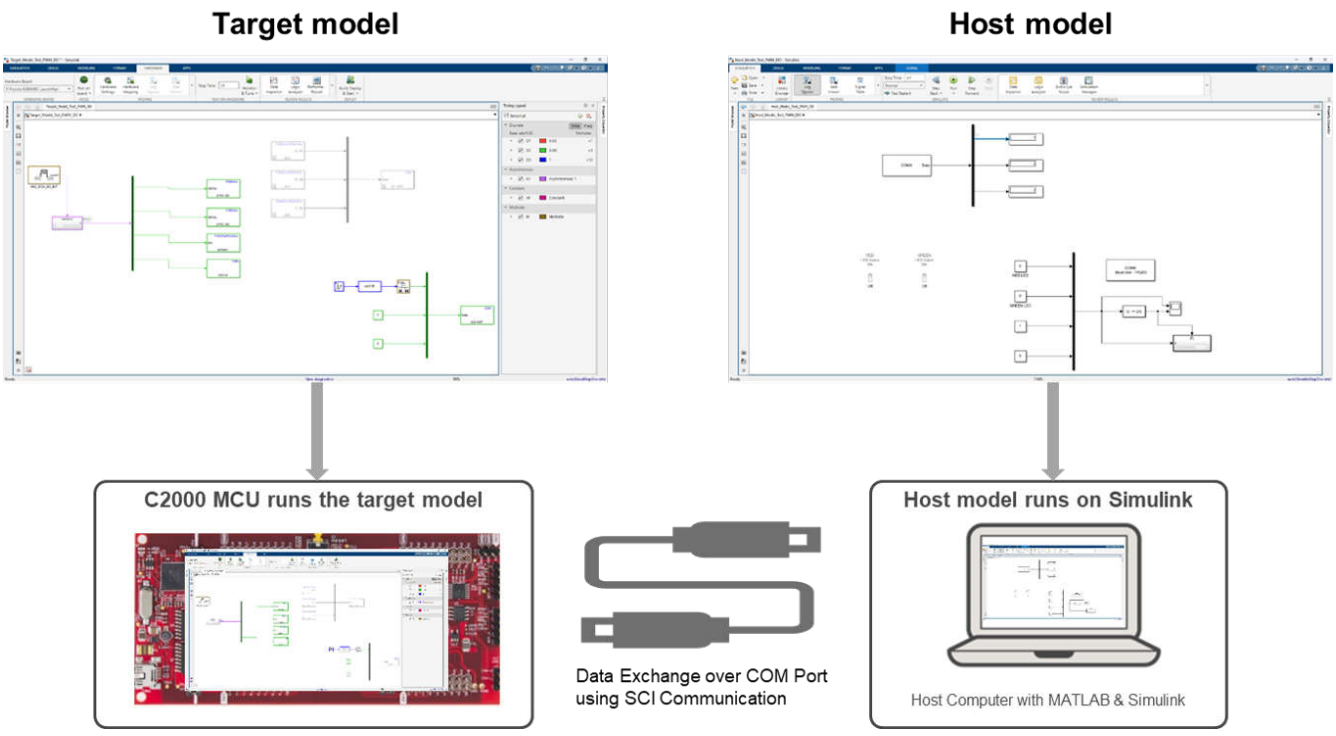

图 2-4. 系统概述

2.3 硬件、软件和测试要求

本节重点介绍了在 TIDM-02012 eCompressor 硬件上运行基于模型的设计的要求。有关指导原则的详细信息，请参阅*混合动力汽车和电动汽车 HVAC eCompressor 高压电机控制参考设计*中的*硬件*、*软件*、*测试要求和测试结果*部分。

2.3.1 硬件设置

1. 将 USB 电缆连接到 TMDSCNCD280039C controlCARD，进行 JTAG 连接。
2. 将 eCompressor 电机电线连接到端子 J13U、J13V 和 J13W。
3. 连接测量设备（万用表、示波器探头和其他测量设备），以根据需要探测或分析各种信号和参数。向端子 J5 施加 12V 辅助直流电源。
4. 在端子 J2 和 J3 上施加直流总线电源。设计的最大输入为 $800V_{DC}$。

备注

如果在测试时外部仿真器出现连接问题，请在 JTAG 信号和 USB 电缆上添加铁氧体磁珠。此外，使连接线路尽可能短。

警告

两个电源域的接地平面可以相同或不同，具体取决于硬件配置。在将任何测试设备与电路板连接之前，应满足适当的隔离要求，以确保您和您的设备的安全。在为电路板供电之前，请检查 GND 连接。如果测量设备连接至电路板，则需要隔离器。

2.3.2 软件设置

下载并安装 MATLAB、Code Composer Studio™ (CCS) 和 C2000Ware。有关这些软件版本的详细信息，请参阅节 1.1。

此参考设计基于模型的设计作为 C2000 MotorControl SDK 的一部分提供。完成下载和安装后，您可以转至 *C2000Ware_MotorControl_SDK_X_XX\solutions\tidm_02012_ecompressor\matlab*，找到此设计的文件夹。MATLAB 文件夹最初只包含 .slx 文件。编译工程后，也可以在同一文件夹位置找到生成的文件。

该示例包含两个主文件：*TIDM_02012_F280039_MBD.slx* 和 *TIDM_02012_control_host.slx*。TIDM_02012_F280039_MBD.slx 文件包含主控制环路实现，而 TIDM_02012_control_host.slx 包含通用接收器/发送器 (UART) 代码，用于发送与所需速度和控制参数相关的数据。

2.3.3 测试过程

要在硬件上运行算法，请通过在 TIDM_02012_F280039C_MBD 示例中选择 *Edit motor & inverter parameters* 选项来配置逆变器和电机对应的硬件参数。这会打开 tidm_2012_param_init_script.m，其中包含电机和逆变器参数初始化，例如 PWM 频率、数据类型、电机型号和电气参数。它还包含 C2000 器件初始化参数，例如器件型号、频率、PWM、ADC 配置。

用户需要根据具体应用来更新参数。

在硬件上运行该算法之前，可以使用 Simulink 中的仿真来测试控制环路算法。要运行仿真，在配置了电机和逆变器的所有参数后，点击 *Simulink 窗口中的“Simulation”选项卡，然后点击“Run”*。可以通过在示例中任意放置示波器或数据显示块来检查仿真的执行情况。通过从 Simulink 的 *Apps* 选项卡中打开 Simulink Data Inspector，可查看预设参数。您可以自由添加验证算法所需的更多参数。

请记住，在运行仿真时不会生成任何代码。仅能对控制算法的运行情况进行验证。在此之前不需要连接 C2000 器件 F280039C。

验证仿真后，要在硬件上部署代码，请选择 *Hardware* 选项卡并选择 *Build, Deploy and Start*，如图 2-5 所示。确保已连接硬件，以便能够部署代码。代码部署在硬件上后立即开始运行。

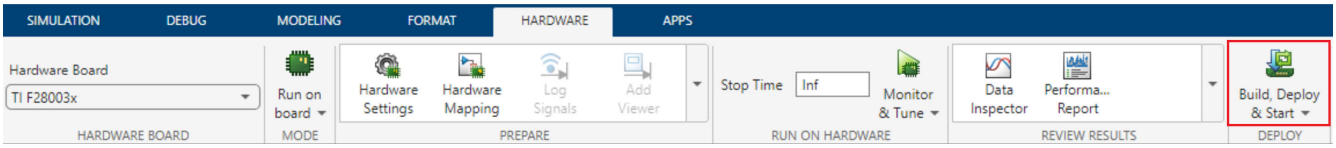

图 2-5. 编译、部署并开始

若要控制启动/停止、电机转速等电机运行，并观察速度、电流等运行时参数，必须使用单独的 TIDM_02012_control_host 模型示例文件。打开该文件并为 UART 通信选择正确的 COM 端口。确保在 *Host Serial Setup*、*Host Serial Receive* 和 *Host Serial Transmit* 中启用了相同的 COM 端口，如图 2-6 中高亮所示。

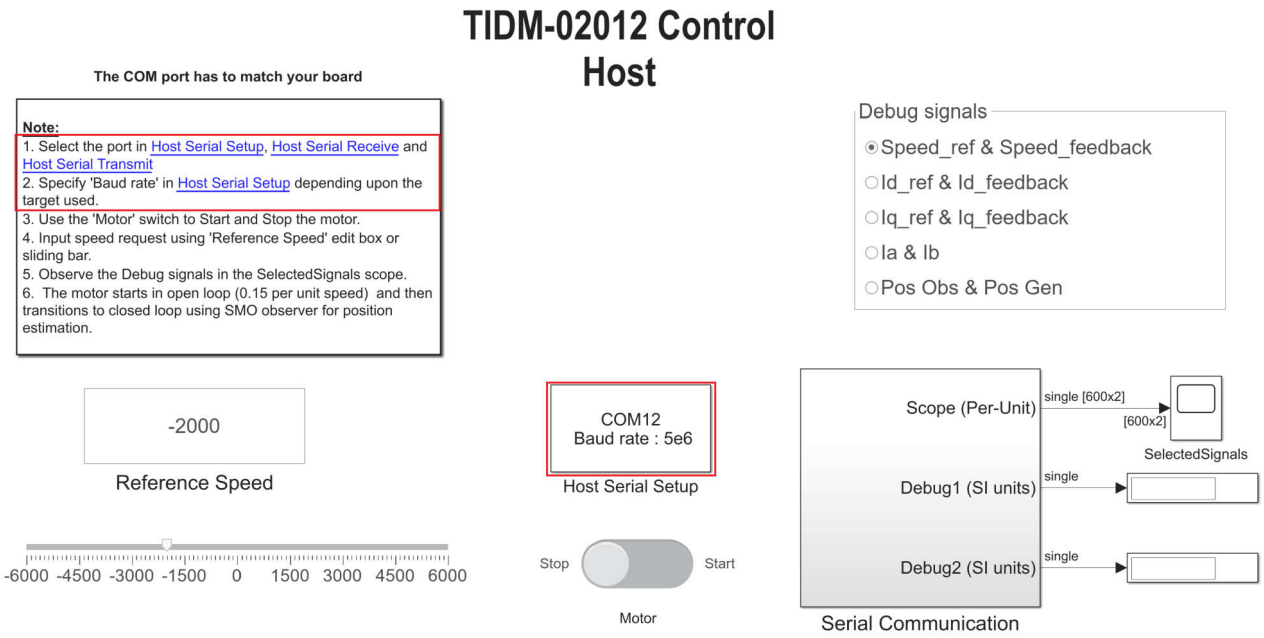

图 2-6. 主机串行配置

与器件通信的波特率要配置为与硬件上所部署控制算法模型中配置的相同，示例中的默认值为 5e6。要检查主示例，请打开模型资源管理器 (Ctrl + E)，转至 *Hardware Implementation*，展开 *Target hardware resources* 下拉列表，然后在“SCIA”下检查已配置的波特率，如图 2-7 所示。

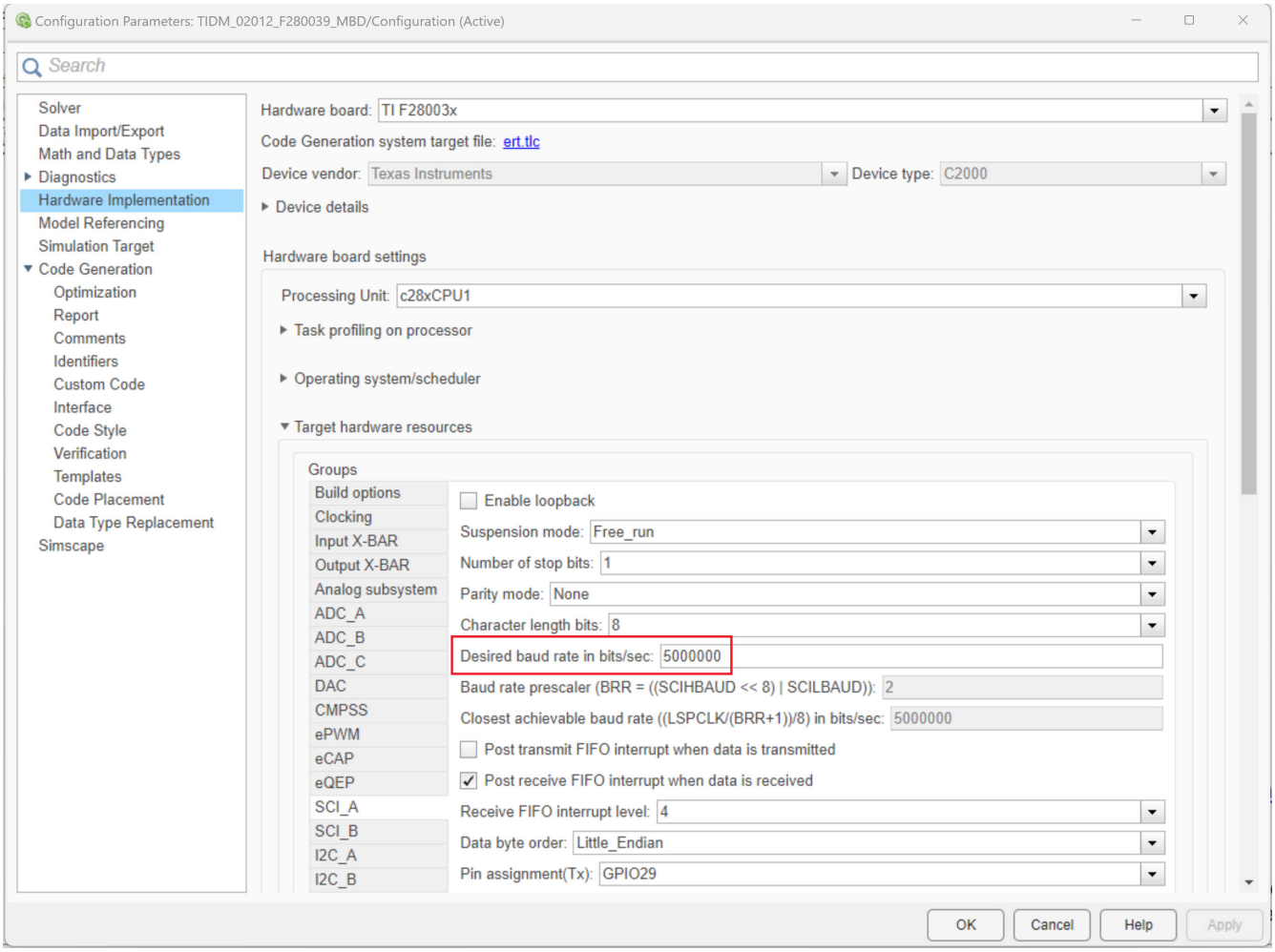

图 2-7. 波特率配置

通过点击“Simulation”选项卡中的 *Run*，运行控制主机示例。将“Simulation Time”保持为“Inf”，以实现连续执行。现在将持续读取从这些选项中选择的参数并在电机启动时将其绘制在示波器上。此外，还可以通过移动标记或在框中提供输入来控制电机转速。

3 MathWorks® Simulink 配置设置

使用基于模型的设计时，一个关键挑战是为 C2000 MCU 生成优化的代码。在使用 C2000 Microcontroller Blockset 生成代码时，使用的优化分为两个阶段：Simulink 工具优化和 C2000 特定优化。若要为应用生成最优的代码，需要配置上述每个优化。需要注意的是，虽然我们在上一节中讨论了 TIDM-02012 eCompressor 应用，但此处讨论的优化设置是通用的，可用于为所有应用高效生成代码。

3.1 Simulink 工具优化

C2000 Microcontroller Blockset 与 Embedded Coder 负责为 C2000 器件生成代码。为了确保生成的代码针对 C2000 进行优化，需要特别选择工具中的配置。Simulink 的“model configuration”选项卡中提供了与基于模型的代码生成相对应的所有配置。

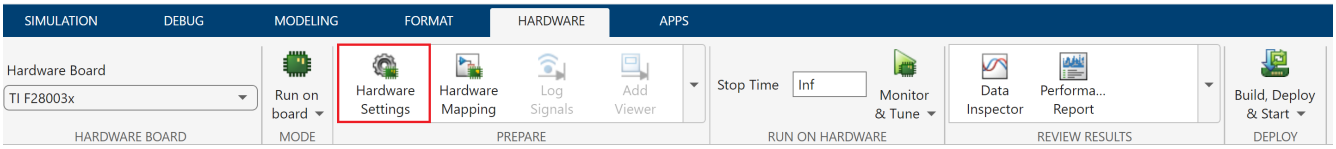

图 3-1. 硬件设置

通过在 Simulink 窗口的 *Hardware* 选项卡下选择 *Hardware settings* 选项，打开模型配置参数。代码优化配置可在 *Optimization* 选项卡的 *Code Generation* 下拉列表中找到，如图 3-2 中所示。

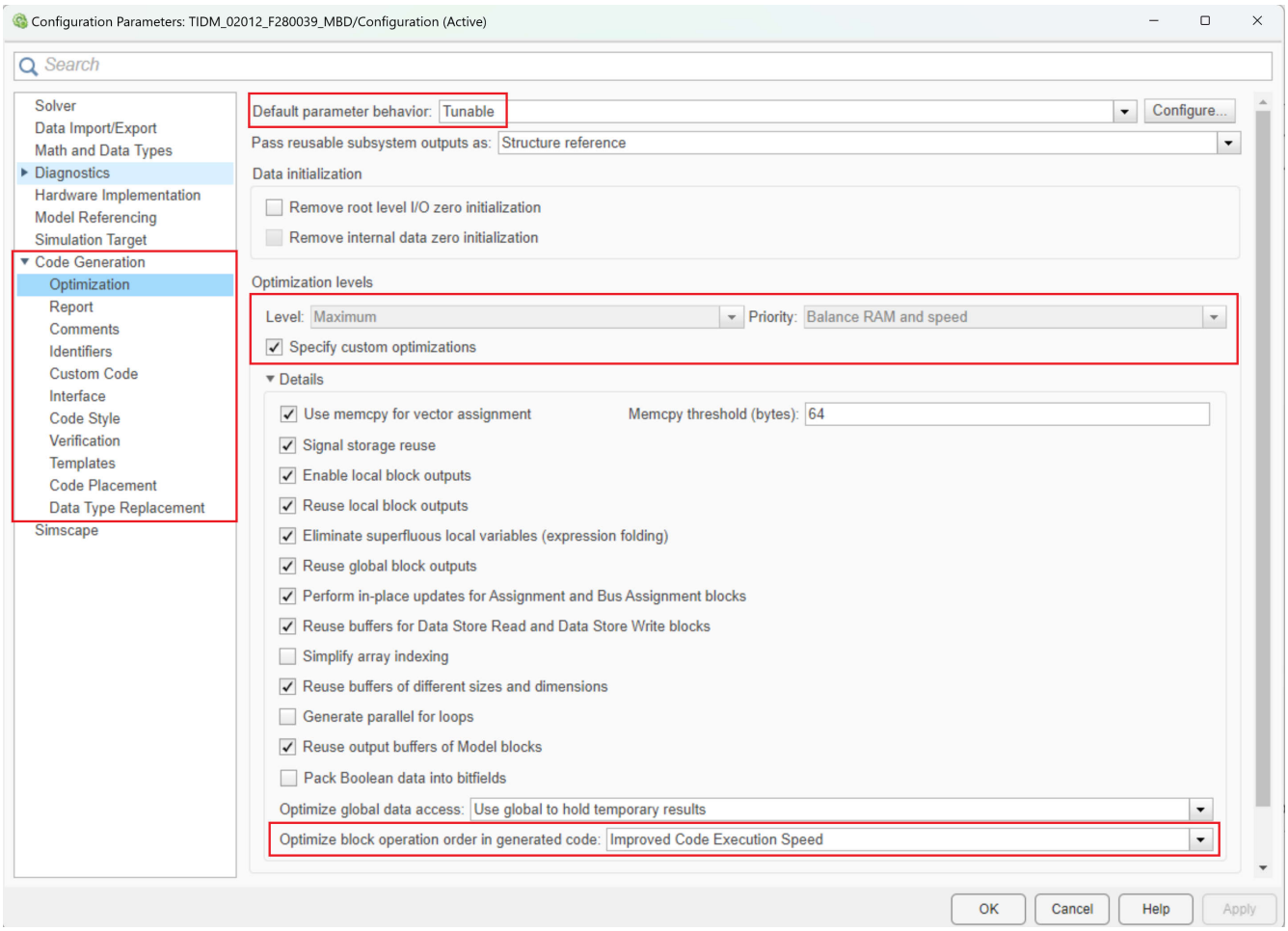

图 3-2. 配置参数

为确保针对 C2000 生成最佳代码而要进行的配置如表 3-1 所示。

表 3-1. MATLAB 优化设置

Simulink 选项卡	具体设置	默认配置	优化配置
Code Generation	Build configuration	Faster Builds	Faster Runs
	Prioritized objectives	Unspecified	Execution efficiency
Optimization	Priority	Balance RAM and speed	Speed
	Specify custom optimization	On	Off
	Default parameter behavior	Tunable	Inlined
	Efficient Map of Float to Int	Off	On
Interface	Support absolute time	On	Off
	Support complex	On	Off
	Support non finite	On	Off
Math and Data Types	Life span	auto	1

通过在 *Model configuration* 部分中选择适当的选项，可以手动配置上述所有配置，如图 3-2 所示。或者，可以将包含上述配置的 MATLAB .m 脚本与应用示例集成，以确保调用优化的配置。

名为 *TIDM_02012_F280039C_MBD_optimconfigs.m* 的 TIDM_2012_F280039_MBD 示例中提供了与该应用集成的脚本。

备注

示例 *TIDM_02012_F380039C_MBD_optimconfigs* 随附的脚本可按原样用于其他应用示例，方法是仅更改模型名称并在生成代码之前运行，以确保在 Simulink 工具中配置所有最佳设置。

代码生成：

1. 编译配置：以更快速度运行

   为“build configuration”选择“faster runs”可以确保 C2000 工具链配置更新为使用编译器优化 -O2，而不是默认的 -O0。
2. 目标优先事项：执行效率

   在目标优先事项中将执行效率保持为最高优先级，可确保从 MATLAB 中生成代码时侧重于执行速度。目标优先事项中还提供了 RAM、ROM 效率等其他参数，可根据要求调用这些参数，但可能会影响性能。默认配置为“none”。

优化：

1. 优化优先事项：速度最大

   用户可以配置优化优先事项，以实现最大速度或最小内存 (RAM)，或者在 RAM 最小和速度最大之间实现平衡。要生成在执行速度方面性能更优的代码，应将优化优先事项设置为实现最大速度。默认行为是优化时在 RAM 与速度之间保持平衡。

   将 **Configuration Parameters > Code Generation > Optimization > Optimization levels > Priority** 更改为 **Maximize Execution Speed**。这会应用代码生成设置来实现最大执行速度。

2. 优化自定义：关闭

   如果未选中“optimization customization”，则 *Details* 选项卡下会自动填充默认配置，使块操作针对提高代码执行速度进行了优化。

   在 **Configuration Parameters > Code Generation > Optimization > Optimization levels** 中，取消选中 **Specify custom optimizations**。这会禁用 **Details** 部分中的参数，因此您无法单独选择或清除这些参数。
3. 默认参数行为：内联

   通过将 **Configuration Parameters > Code Generation > Optimization > Default parameter behavior** 更改为 **Inlined** 来优化默认参数行为，从而提升性能。

   此外，在 **Configuration Parameters > Code Generation > Optimization > Advanced parameters** 中，选中 **Inline invariant signals** 复选框。这在生成的代码中使用模型参数的数值，而不是它们的符号名称。
4. 浮点到整型的有效映射：开启

   从浮点到整型的数据类型映射为待启用。此配置会移除处理 NaN 值的浮点到整型转换结果的代码。

   在 **Configuration Parameters > Code Generation > Optimization > Advanced parameters** 中，选中 **Remove code from floating-point to integer conversions with saturation that maps NaN to zero** 复选框。这会移除处理 NaN 值的浮点到整型转换结果的代码。

接口：

在“interface”选项卡下，可以禁用对所有不打算使用的数据类型配置的支持。为了生成高效的代码，可以移除对绝对时间、非有限数和复数的配置支持。*Support* 选项卡中启用了针对所选配置的额外检查，这可能会在运行时应用代码中增加额外数量的周期。

在 **Configuration Parameters > Interface > Software environment** 中，取消选中 **Support: absolute time, non-finite numbers and complex numbers**。这会导致支持依赖于绝对时间、非有限数和复数的块。取消选择 *Support* 选项卡下的未使用配置可确保更好的运行时代码效率。

应用生命周期：

设置 **Configuration Parameters > Math and Data Types > Advanced parameters** 中的 **Application lifespan (days)** 字段。这指定了依赖经过时间的应用程序在计时器溢出发生前可以执行多长时间（以天为单位）。

3.1.1 最佳代码生成

表中展示的配置可以通过在 Simulink 的 *Hardware Settings* 中配置每个参数来手动配置。为了避免手动操作，MATLAB 还允许通过 MATLAB 脚本在 Simulink 中配置设置。

优化的代码生成配置也可以通过脚本轻松解析，方法是在现有模型中包含脚本，或者直接在运行应用代码之前运行一次脚本来配置所有设置。脚本中的参数“mdl”需要反映所用模型的名称，才能正确配置设置。如果配置了这些设置，则可以通过在 Simulink 窗口中手动检查 *Hardware Settings* 中的配置来验证这些设置。

```
%% Load the model
mdl = 'TIDM_02012_F280039_MBD';%Model Name
load_system(mdl);

%% Set Build Configurations and Prioritized Objectives in Code Generation tab
set_param(mdl,'BuildConfiguration','Faster Runs') ; %Build Configurations
set_param(mdl,'ObjectivePriorities','Execution efficiency');  %Prioritized Objectives
%% Set Level, Priority in Optimization levels and enable some advanced Parameters in Optimization
tab
set_param(mdl,'OptimizationPriority','speed');%Optimization Priority

set_param(mdl,"OptimizationCustomize","off");%Customize Optimizations Checkbox
%% TODO: Check -- updated Code Config > Optimization > default parameter behaviour to inlined
instead of tunable to make the below config work
set_param(mdl,"InlineInvariantSignals","on");%Inline Invariant Signals
set_param(mdl,"EfficientMapNaN2IntZero","on");% Removes code from float to int with saturation
mapping NaN to zero.
%%  Remove support for non-finite, complex and absolute time Interface tab
set_param(mdl,"SupportAbsoluteTime","off");%Remove Absolute time support
set_param(mdl,"SupportComplex","off");%Remove Complex Number support
```

```
set_param(mdl,"SupportNonFinite","off");%Remove Non-Finite Number support
%% Application Lifetime setting
set_param(mdl,"LifeSpan","1");%Remove Absolute time support
```

备注

在上面的代码块中，将模型名称替换为使用中的应用模型名称，并在执行应用模型的代码生成之前运行脚本，以包含所有优化的配置。

3.2 C2000 专用优化

为了确保生成的代码在 C2000 MCU 上以最佳方式执行，除了 Simulink 专用优化外，还需要正确配置编译器设置，这些设置是特定于 C2000 MCU 的硬件核质。如节 3.1 中所述，如果编译配置设置正确配置为 *Faster Runs*，则在硬件上运行代码时将调用优化级别 (-O2)。此外，如果要手动更改优化级别，您可以在“hardware settings”的“Code generation”选项卡中转至 *Toolchain details* 部分，以通过将编译配置选择为 *Specify* 来配置编译器设置。编译器优化设置位于 C 编译器设置中，如图 3-3 所示。

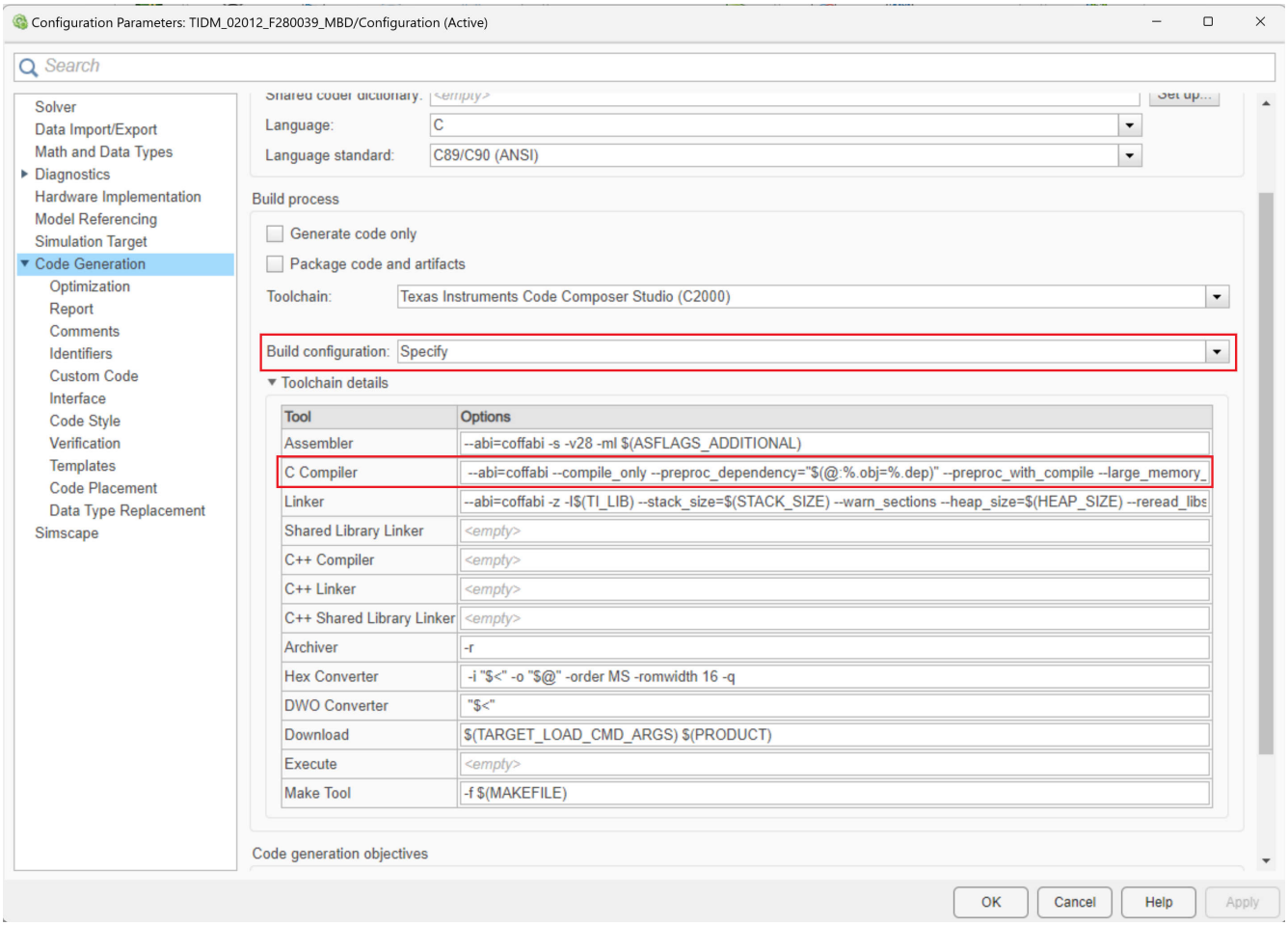

图 3-3. 自定义编译器优化配置

3.2.1 通过 Simulink 使用 TMU

C2000 器件还包含三角函数加速器 (TMU) 等加速器来更快地执行浮点三角运算。为确保生成的代码使用 TMU 指令，“Hardware Settings”中的“Hardware Implementation”选项卡包含用于启用 TMU 的选项。这确保在执行三角运算时均会调用 TMU 指令。默认情况下，除非手动取消选中，否则会对具有 TMU 加速器的器件启用 TMU。

虽然 TMU 可以通过调用 C2000 编译器用户指南中所述的内在函数来进行专门调用，但通过在 Simulink 窗口中启用 TMU 函数，仍然会使用传统的三角运算，而不是 TMU 内在函数。TMU 通过在编译配置中使用适当的编译器标志来启用，并确保正确调用硬件加速器以执行三角运算。

备注

即使启用了 TMU，正弦/余弦 LUT 等查询表块仍会使用基于 LUT 的方法。需要使用 Simulink 库中的正弦/余弦块（而不是 LUT 块）来生成优化的三角运算。

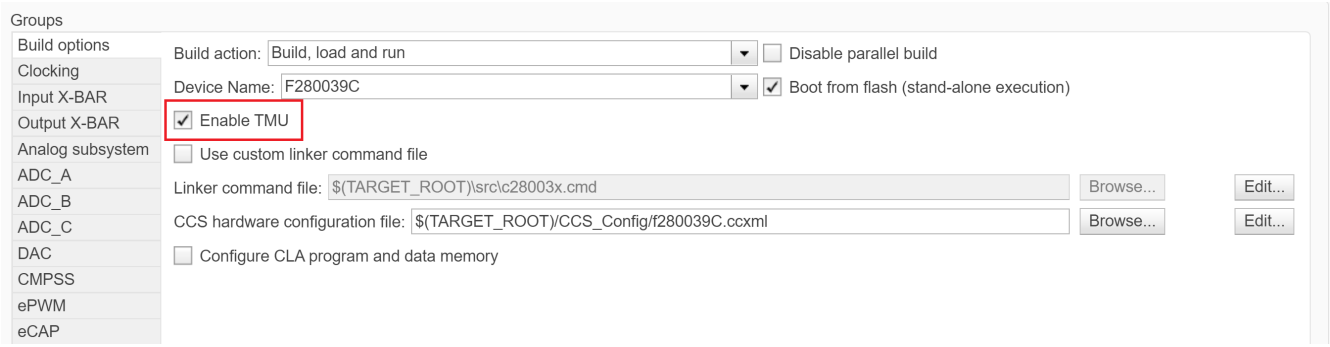

图 3-4. TMU 配置

3.2.2 通过 Simulink 使用软件库

为了调用其他软件库以加快计算速度，嵌入式编码器允许导入代码替换库 (CRL)，在适用的情况下使用库函数代替传统代码生成。C28x IQMath 库便是用于更快地执行定点算术计算的库之一。如果应用使用 Q 值，建议使用 IQMath CRL。要调用 IQMath CRL，请转至“Hardware settings”中 *Code Generation* 选项卡下的 *Interface* 部分。如图所示，选择 *TI C28x* 库，该库包含 IQ Math、FastIntDiv 和 CLA 等的代码替换库。

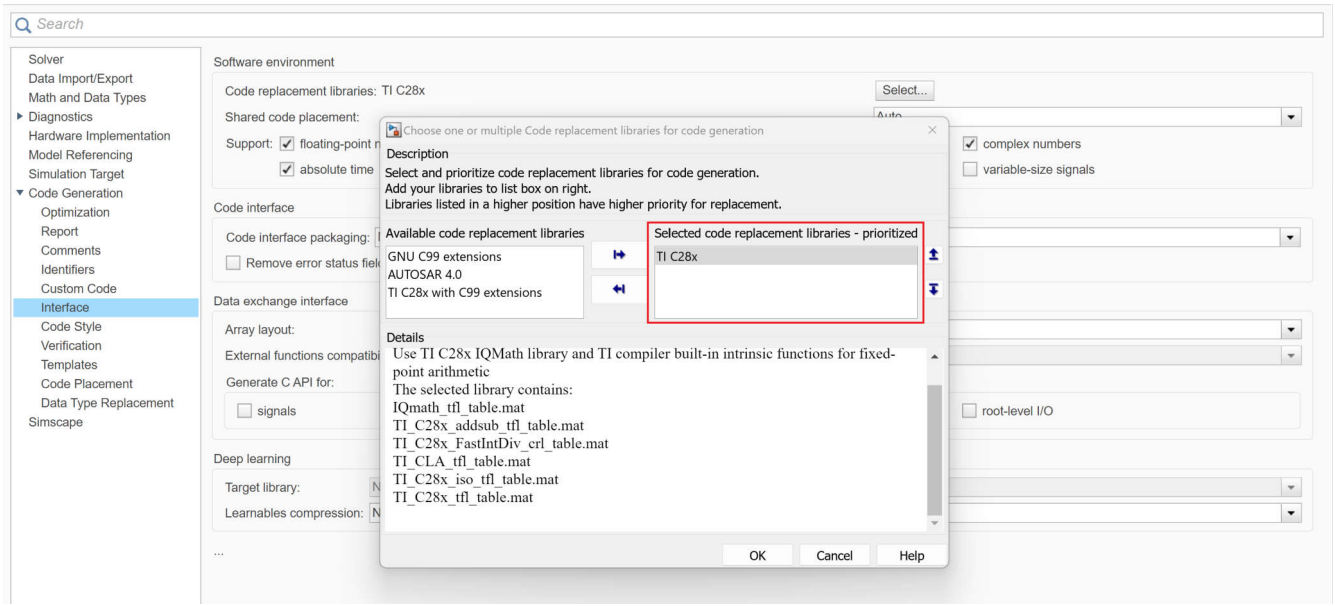

图 3-5. 代码替换库配置

备注

2018 年之前的 MATLAB 版本不支持 TI C28x 代码替换库。

备注

需要注意的是，硬件加速器三角函数加速器 (TMU) 是一种用于更快地执行浮点运算的加速器，而 IQ Math 库是一个用于更快地执行定点计算的软件库。

3.2.3 从 RAM 运行代码

任何应用代码都可以在 RAM 或闪存中存储并进行运行时执行。将代码存储在闪存存储器可允许存储更大的应用程序，因为闪存的存储容量通常大于 RAM，但从闪存执行代码会在执行读写操作时调用额外的等待状态。由于从 RAM 执行代码更快，因此理想的解决方案是将代码存储在闪存存储器中并从 RAM 执行代码。

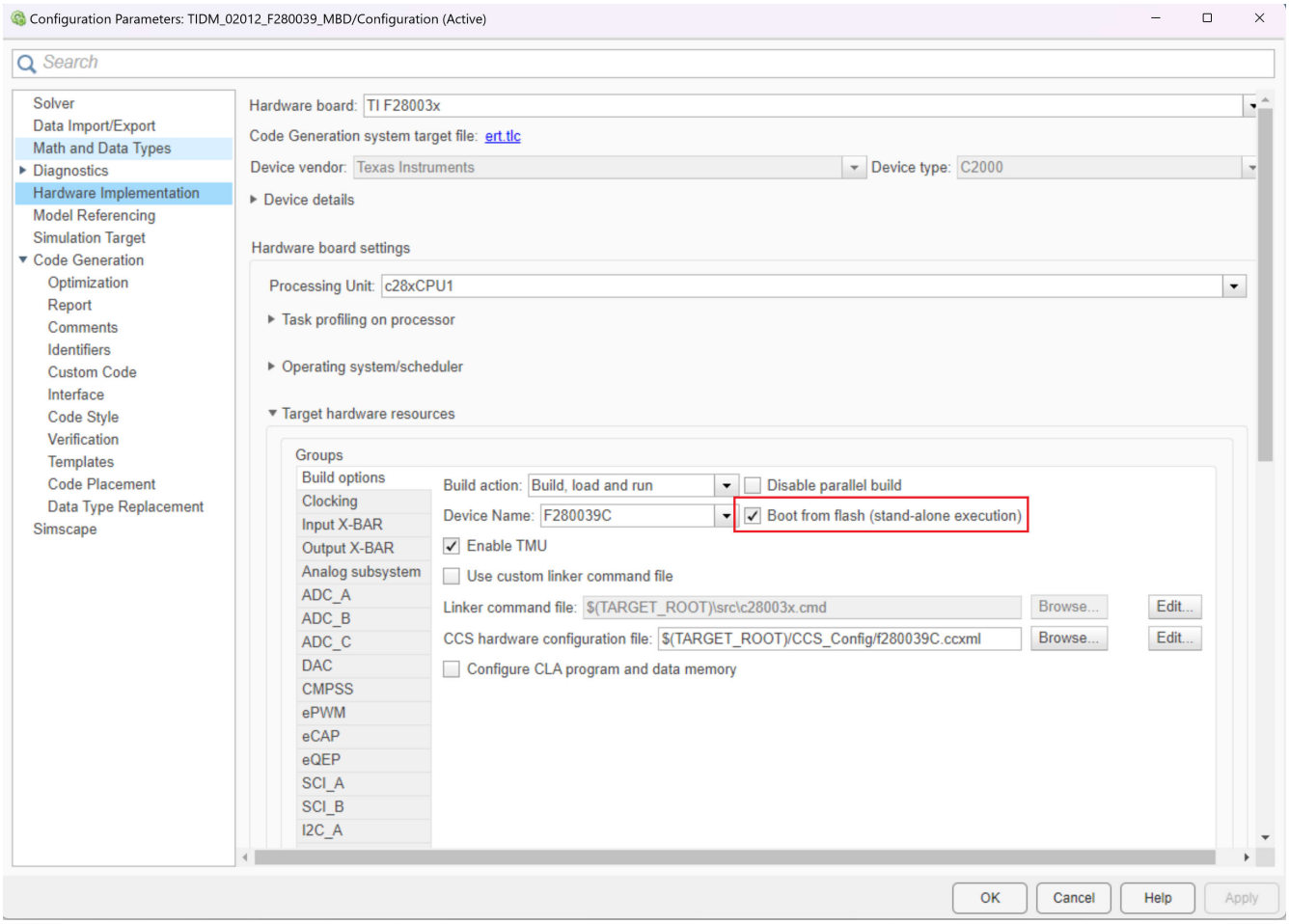

图 3-6. 从闪存引导

如果按图 3-6 所示选择了从闪存引导的选项，则代码将存储在闪存中。要从 RAM 复制并运行，请右键点击子系统块并选择 *Block Parameters (Subsystem)*。选择 *Code Generation* 选项卡并从下拉菜单中将 *Memory section for execution functions:* 配置为 *code_ramfuncs*，如图 3-7 所示。需要注意的是，如果子系统大于器件上的可用内存，则会显示编译时间错误，表明内存不足。

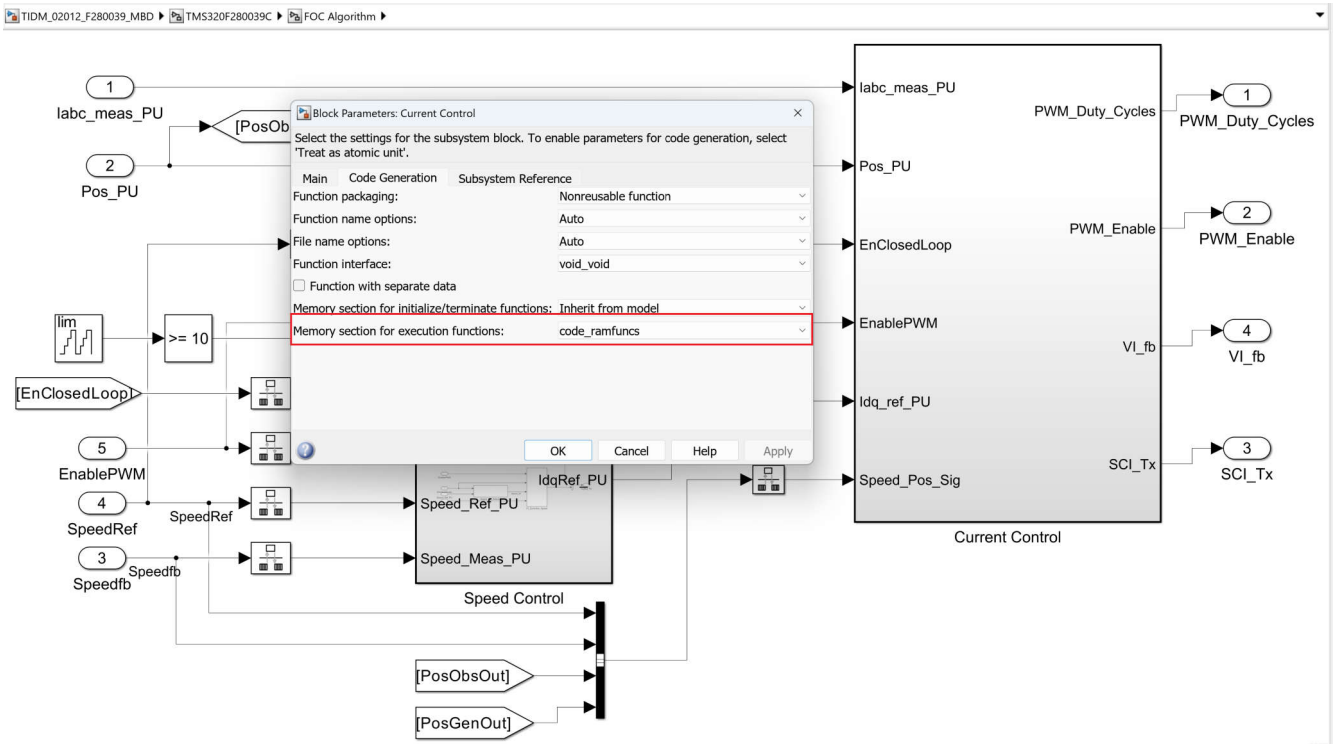

图 3-7. 从 RAM 运行代码

3.3 性能比较

启用节 3 中所述的优化后，验证 TIDM-02012 eCompressor 模型相对于默认配置下的性能提升 。FOC 执行速度最快的环路的整体性能提升大约 57%，而对于速度环路，执行性能提升大约 38%。

性能比较中展示的数值使用处理器在环 (PIL) 技术进行分析，节 4.1 对此进行了介绍。默认配置仍使用 C2000 硬件加速器，即 TMU。优化配置列的不同之处在于通过节 3.1.1 中所示的脚本启用了在 MATLAB 中优化的配置设置。

表 3-2. 性能比较

功能 (*执行速率*)	默认配置执行时间 (*ns*)	优化配置执行时间 (*ns*)	相对于默认配置的性能提升	默认配置平均 CPU 利用率	优化配置平均 CPU 利用率
电流控制环路 (66.67 µ s)	15261	6286	57.45%	22.89%	9.43%
速度控制环路 (0.67ms)	2961	1840	37.87%	0.44%	0.28%
后台辅助控制环路 (0.1s)	260	200	23.07%	0.0003%	0.0002%

4 MathWorks 使用 Simulink 进行性能分析

Embedded Coder 工具允许使用 Simulink 灵活地对部分应用代码、应用代码中的多个块或完整应用代码进行性能分析。对 MATLAB 生成的代码进行性能分析的方法有多种，包括 Simulink 上的处理器在环方法、C2000 计时器、基于通用输入/输出 (GPIO) 的性能分析以及 CCS 时钟分析器工具。虽然性能分析数据方面没有明显的区别，但由于对应用代码进行性能分析的方法不同，可能存在细微的差异。

4.1 处理器在环 (PIL) 方法

Simulink 提供处理器在环 (PIL) 仿真工具，该工具允许使用 SIL/PIL 管理器工具验证代码并对其进行性能分析。要使用 PIL 工具进行性能分析，需要配置硬件设置。打开“hardware settings”(Ctrl + E)，转到 *Code Generation* 下的 *Verification* 选项卡，并使用图 4-1 中所示的设置启用 *Measure task execution time*，然后在 *Advanced Parameters* 下的 *Create Block* 部分中选择 *PIL*。

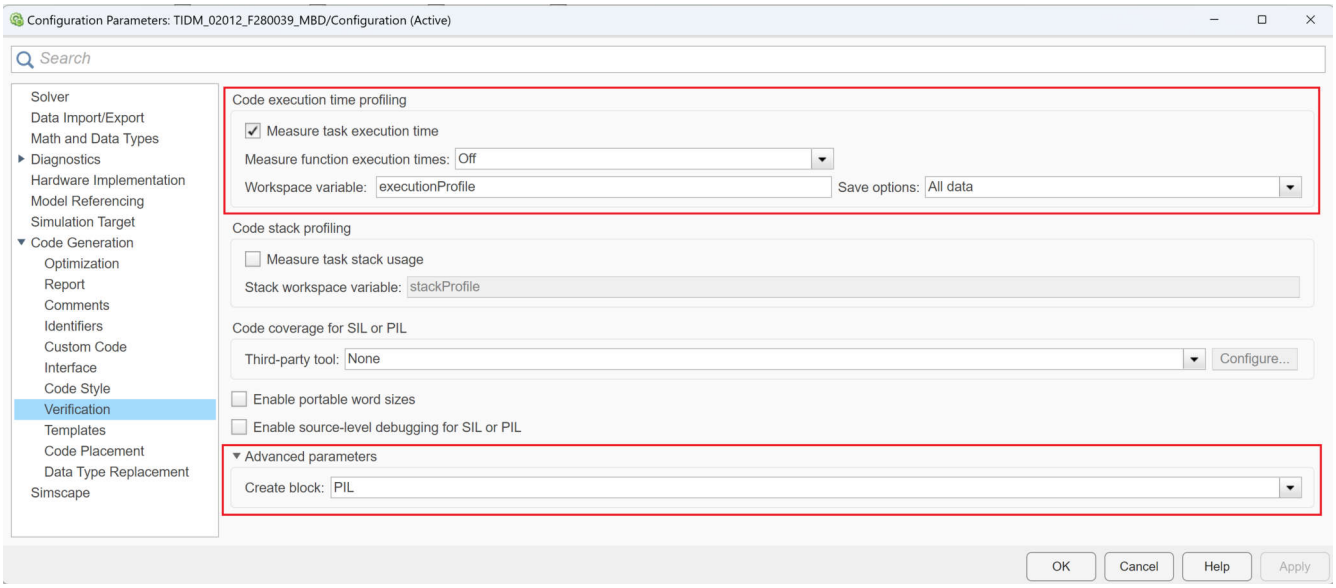

图 4-1. 处理器在环性能分析

在 *Hardware Implementation* 选项卡下 *Target hardware resources* 的 PIL 部分中，按图 4-2 所示配置 COM 端口。根据器件 UART 端口，在 MATLAB 中验证串行端口连接。

要从子系统/系统创建 PIL 块，请右键点击要选择用于性能分析的块，然后在 *C/C++ Code* 中选择 *Deploy subsystem to hardware*。这会为所选的块生成代码并为该块生成 PIL 块。将生成的 PIL 块替换为实际块。从 *APPS* 选项卡打开 SIL/PIL 管理器。选择 *SIL/PIL Simulation Only* 作为模式，选择 *Model blocks in SIL/PIL mode* 作为待测系统，并在运行自动验证之前提供仿真的参考停止时间，如图 4-2 所示。成功完成执行后，可以在 MATLAB *Results* 部分下提供的报告中查看代码执行参数。该报告包含 CPU 利用率和以纳秒为单位的执行时间（平均值、最小值和最大值），可用于根据使用的器件计算周期计数。

虽然 PIL 仅用于对单个代码块进行性能分析，但可以使用基于 C2000 计时器和基于 GPIO 的分析方法来一次对多个代码块进行性能分析。基于 PIL 的性能分析优势在于性能分析数据能够清晰地反映执行速率。如果多个循环以不同的速率运行，PIL 会根据执行速率将每个块的执行时间分开。基于 PIL 的方法在测量中具有固定的小开销，因此建议使用基于 PIL 的性能分析方法对块进行性能分析，该方法使用更大的代码大小来尽可能地降低开销的影响。

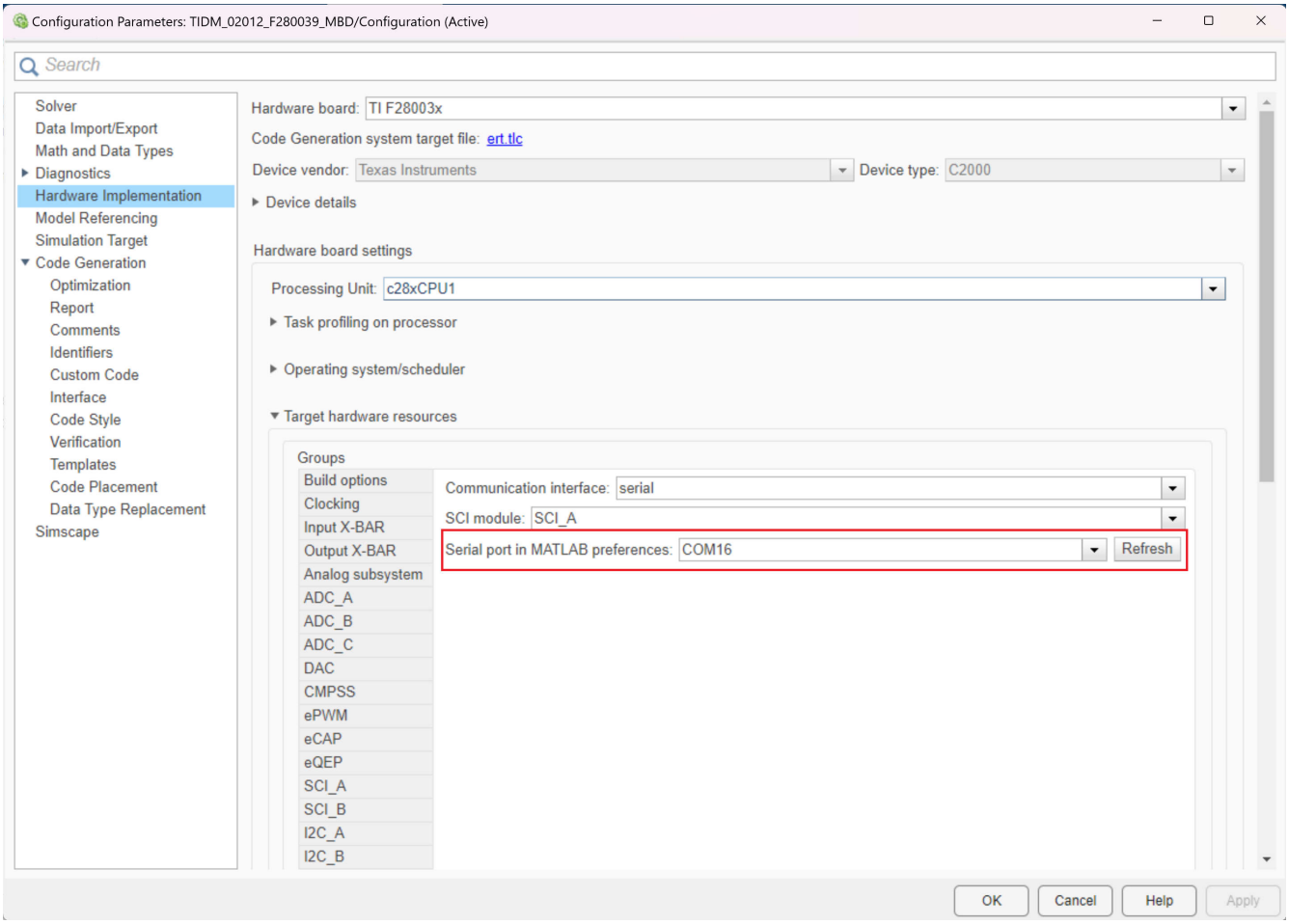

图 4-2. PIL COM 端口配置

图 4-3. PIL 配置设置

有关 PIL 仿真的更多信息，请参阅 PIL 仿真文章。

4.2 基于 C2000 计时器的性能分析

C2000 计时器外设也可用于获取执行时间数据。计时器代码需要与现有模型集成以进行性能分析，因此该方法属于侵入式性能分析方法。通过单独测量计时器执行时间，可以消除基于计时器的方法所产生的开销。在运行应用程序的代码时，可以移除计时器代码，以节省额外的计时器配置和运行周期。下面讨论了使用 C2000 计时器外设来对某个部分或完整应用代码进行性能分析的步骤。当前示例使用计时器 2 作为性能分析工具，但如果应用程序已经在使用该计时器，则建议选择未使用的计时器。

在模型的顶层，在 Simulink Coder > “Custom Code” 下，添加 Simulink 库中可用的*系统初始化* 块。此块包含计时器初始化代码。

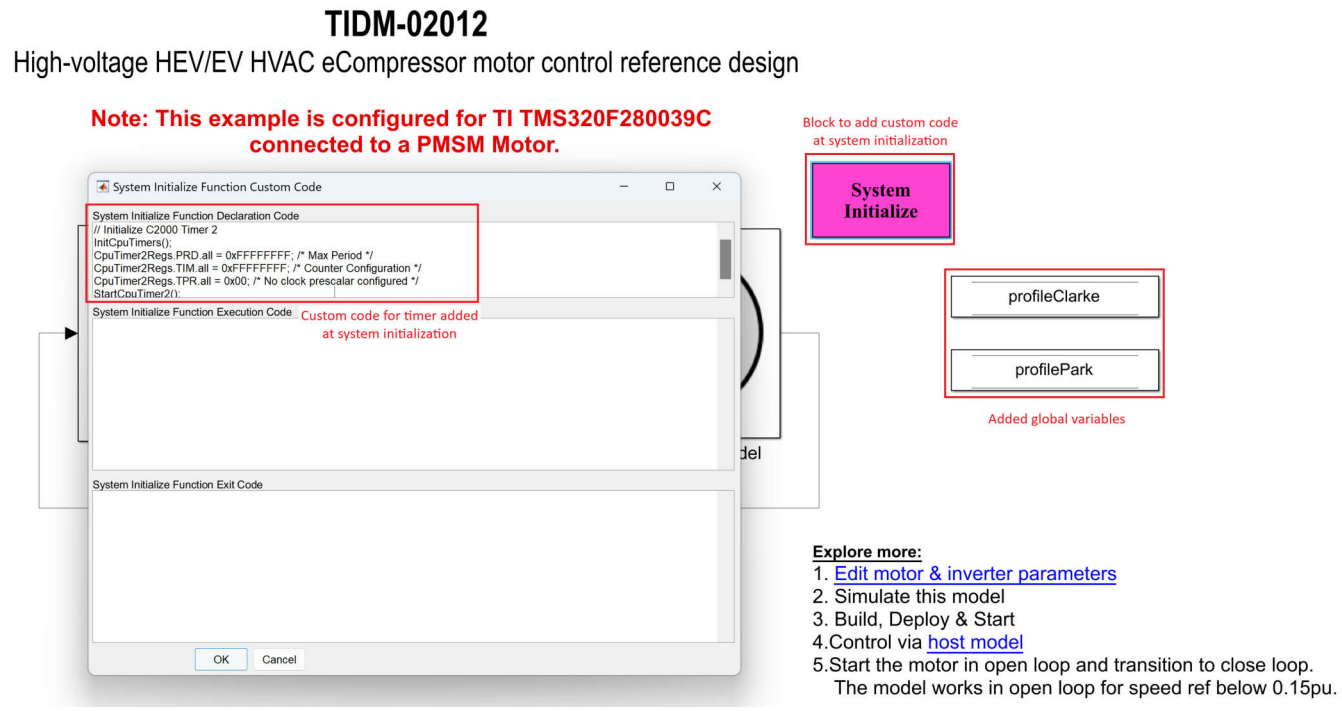

图 4-4. 计时器初始化代码

```
// Initialize C2000 Timer 2 InitCpuTimers(); CpuTimer2Regs.PRD.all = 0xFFFFFFFF; /* Max Period */
CpuTimer2Regs.TIM.all = 0xFFFFFFFF; /* Counter Configuration */ CpuTimer2Regs.TPR.all = 0x00; /* No
clock prescalar configured */ StartCpuTimer2();
```

将*数据存储内存*块与要进行性能分析的块所对应的系统初始化块一起添加到顶层。如图 4-5 所示，对*数据存储内存*块进行适当的命名。从 *APPS* 菜单中打开 Embedded Coder 应用程序。选择“Code Mappings”> “Component Interface”，并导航到 *Data Stores* 选项卡。将使用*数据存储内存*块创建的变量存储类更改为 *ExportedGlobal*。

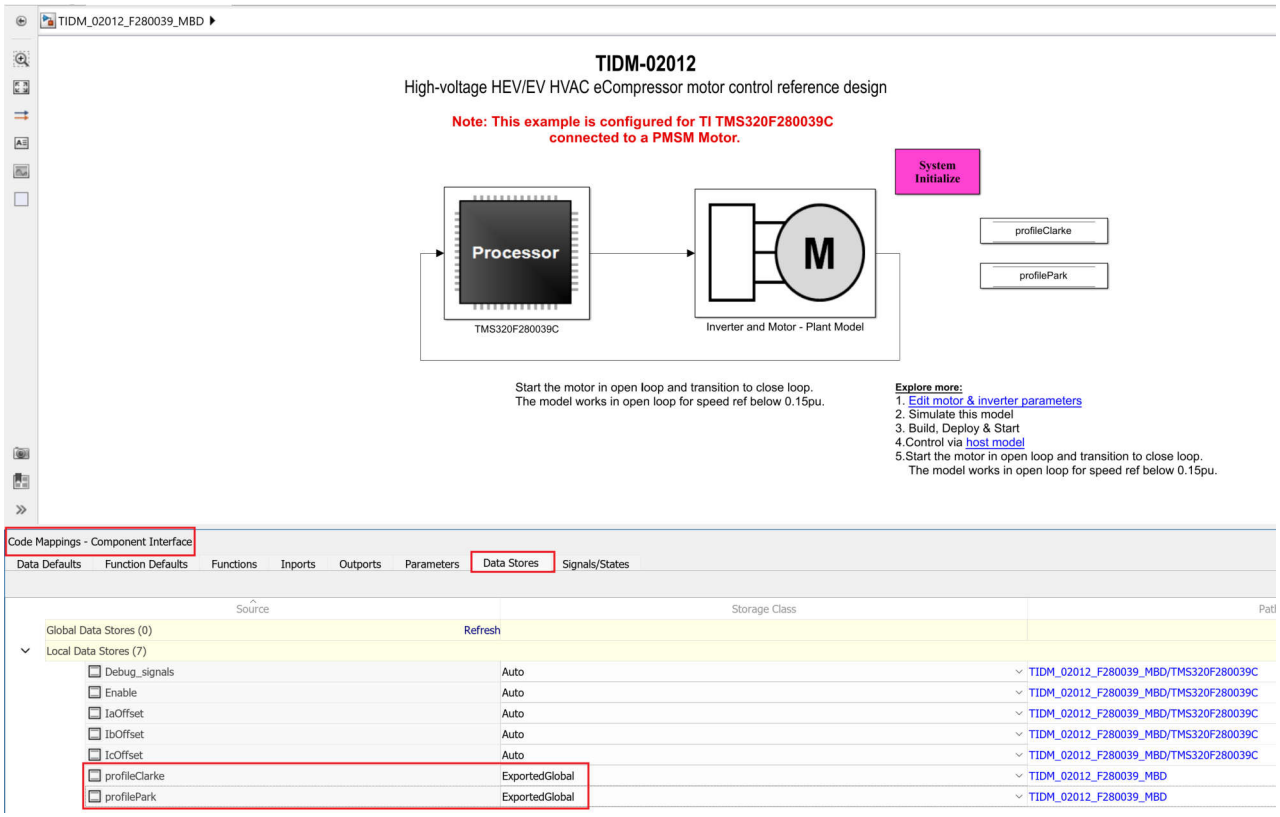

图 4-5. 变量存储类

现在，在要进行性能分析的子系统内，从 Simulink 库添加一个*系统输出*块。如果它是一个独立的块（例如 Park 变换块），那么该块可以成为一个子系统，之后可以添加系统输出块。在*系统输出*块中，在函数声明代码段中添加读取计时器值代码并将其存储在一个临时变量中，然后在函数退出代码段中再次读取计时器值。在退出代码段中计算差值，并将该值存储在之前定义的全局变量中，如代码块中所示。

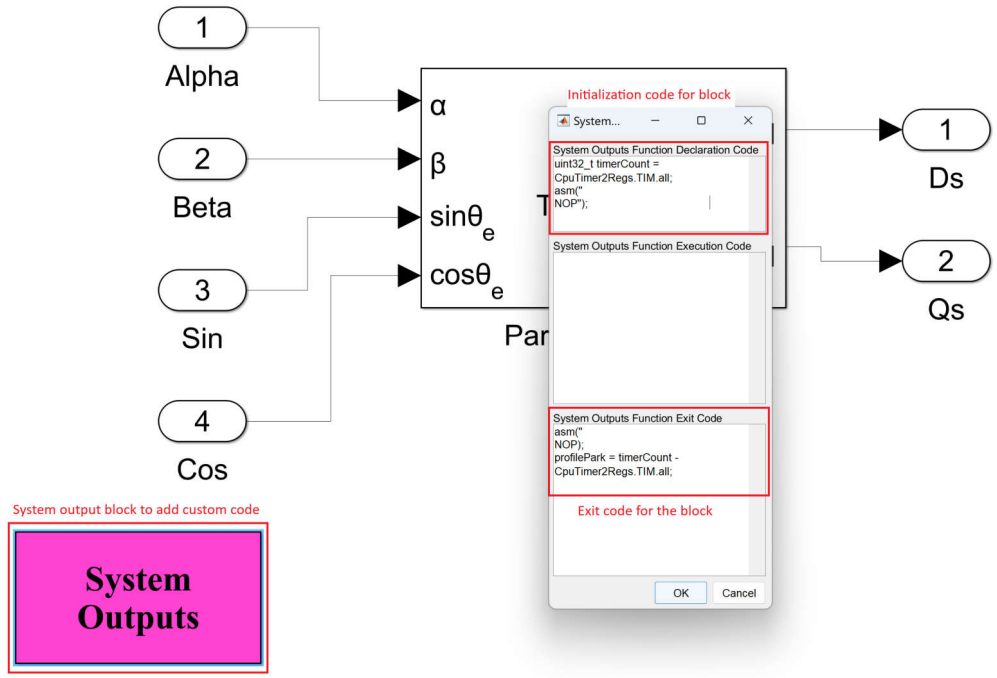

图 4-6. 系统输出计时器代码

要对块性能分析，在创建子系统后，右键点击它并导航至“Block Parameters (subsystem)”。选中“Treat as Atomic Unit”复选框。使用 CCS 构建模型并在器件中加载生成的 .out 文件。在 CCS 的表达式窗口中添加新增的全局变量，以观察和运行代码。打开主机控制模型，选择适当的 COM 端口并设置所需的速度。使用拨动开关启动电机来运行电机。这会向器件发送使能 PWM 信号来运行算法，之后观察窗口中的变量将更新为周期计数。使用表达式窗口中的 *Continuous refresh* 选项，观察变量的变化。

```
/* Code block for System Output Function Declaration Code */ uint32_T timerCount1 =
CpuTimer2Regs.TIM.all; asm(" NOP"); /* Code block for System Output Function Exit Code */ asm("
NOP"); profilePark= timerCount1 - CpuTimer2Regs.TIM.all;
```

要消除计时器测量开销，可以使用下面块中所述的代码在任何块中添加单独的代码块。计时器开销的计算方法是连续读取计时器，两次读取的差值就是开销。执行代码块时，可以量化开销，并从为其他块计算的性能分析数据中减去。需要注意的是，与之前为每个块定义的全局变量相似，也必须添加与计时器开销相对应的变量。

```
/* Code block for calculation of timer overhead */
uint32_T overheadCount = CpuTimer2Regs.TIM.all;
asm(" NOP");

/* Code block for System Output Function Exit Code */
asm(" NOP");
profileOverhead=  overheadCount - CpuTimer2Regs.TIM.all;
```

除了基于 C2000 计时器的性能分析外，如果有示波器，也可以使用基于 GPIO 的性能分析方法。需要注意的是，基于 GPIO 的性能分析需要使用外部示波器来查看波形并测量时序。可以在例程（即需要进行性能分析的例程）的开始和结束时切换 GPIO，而不是进行本节中讨论的计时器读取。在基于 GPIO 的性能分析方法中，开销仅限于写入 GPIO 寄存器所用的时间。

4.3 Code Composer Studio 工具

Code Composer Studio (CCS) IDE 还可以对应用代码的执行周期进行性能分析。通过 CCS 提供的时钟工具可以获取点对点周期计数信息。要使用 CCS 进行性能分析，请在 CCS 环境中导入 MATLAB 生成的工程，方法是打开 CCS 并选择 *Project* 选项卡下的 *Import CCS project*。浏览到 MATLAB 工程文件夹位置并导入工程。导入后，该工程应该会显示在 CCS 窗口的 *Project Explorer* 中。连接硬件并调试项目，以在 C2000 器件上加载 .out 文件。

工程加载完成后，时钟工具便可供使用。有关如何使用 Code Composer Studio 进行性能分析的详细说明，请参阅链接“对 C28x 目标器件进行性能分析”。

5 总结

随着工业和汽车控制变得越来越复杂，针对 C2000 实时控制器使用 MATLAB 的 Embedded Coder 等简化代码工具成为一种明智的选择。不仅可以实现此类复杂时间关键型控制应用的易用性，同时还能满足性能要求。

通过实施如表 3-1 所示的优化配置而非默认配置，足以通过优化的代码生成实现更快的计算性能。要实施优化配置，较简单的方法是在 MATLAB 模型中整合代码脚本。

6 修订历史记录

注：以前版本的页码可能与当前版本的页码不同

Changes from Revision * (07/15/2024) to Revision A (November 2024)	Page
• 更新了文档的标题以包含 Simulink。	1
• 更新了节 2	2
• 更新了节 3.2.1	13

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