Application Note

MCU Control Center 工具开发指南

Ryan Ma, Delaney Woodward and Nima Eskandari

摘要

MCU Control Center 工具是一款适用于 C2000™ 微控制器的数据记录和可视化工具。为便于使用，该工具集成在 Code Composer Studio™ (CCS) 环境和 SysConfig 工具中。本应用手册确定了可与该工具集一起用于实现数据记录的常见硬件设置，描述了各种特性和用例场景，阐述了如何向现有工程添加支持，并重点介绍了可用的 SDK 示例。本文档中介绍的特性包括：

- 应用程序记录 - 将通用数据日志记录到 PC
- 传输桥 - 使用快速串行接口 (FSI) 外设日志进行数据记录，支持 200MBPS 传输速率
- 通信记录 - 使用 FSI 或通信外设应用程序调试实现增强的数据记录功能
- 快速实时记录 - 专为实时应用设计的极低侵入性数据记录

内容

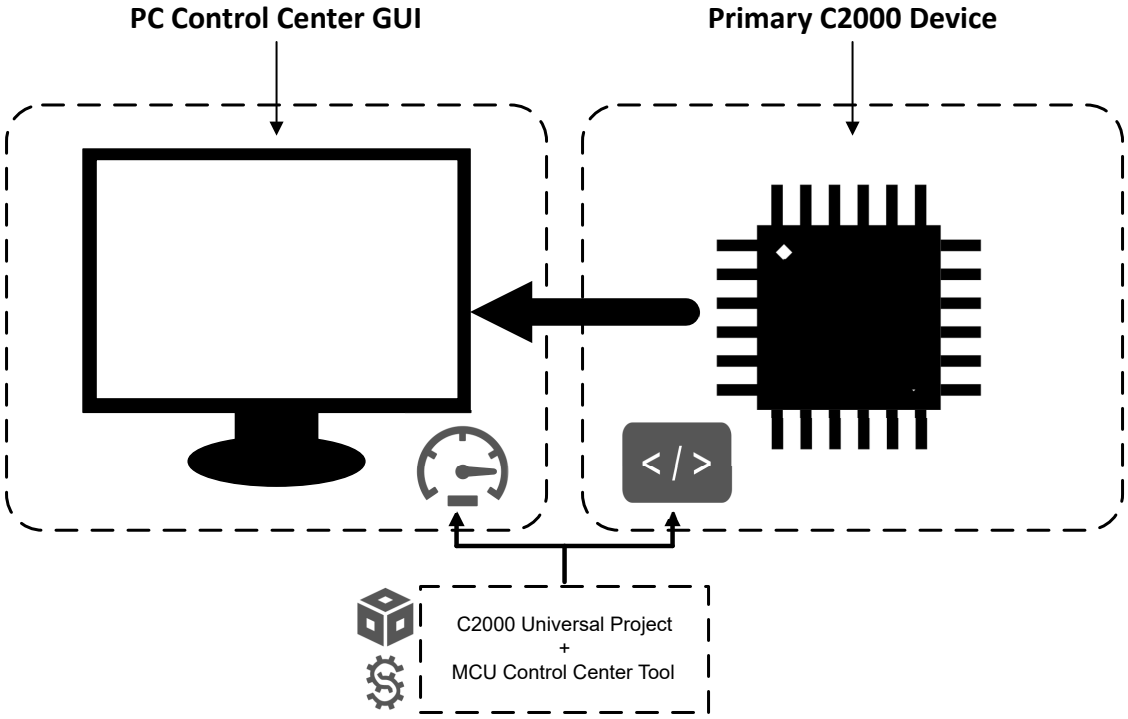
插图清单

表格清单

商标

1 简介

MCU Control Center 是一款基于 SysConfig GUI 的工具，可帮助通过通信外设从器件导出数据，并在 PC 端实现数据可视化。MCU Control Center 会生成应用程序代码，将通信外设配置、数据打包与格式配置以及 PC GUI 创建代码抽象出来。MCU Control Center 的目的是让开发人员可以通过 SysConfig 工具，轻松地将数据记录和可视化功能整合到现有工程中。通过该工具生成的 GUI 可在 CCS 内打开。



备注

在本文中，通用异步接收器/发送器的首字母缩写词 UART 和串行通信接口的首字母缩写词 SCI 通常可以互换使用。SCI 指大多数使用 UART 协议的 C2000 器件上存在的一种外设。

图 1-1. MCU Control Center 简要示意图

2 硬件设置方案

根据可供用户使用的具体 C2000 器件和硬件，可使用四种常见的硬件设置将数据记录功能集成到工程中。第一步是确定记录（主）器件上有哪些外设。若要在 MCU 和 PC 之间创建通信链路，硬件设置中的某个位置必须包含 USB 外设，因为这是连接 PC 的常用协议。

所有 C2000 器件都有一个 SCI 或 UART 外设可使用标准 UART 协议记录数据。在记录器件上使用此协议的优点在于：德州仪器 (TI) 销售的所有 controlCARD 和 LaunchPAD™ 都在电路板上刷写了一个 UART 转 USB 桥接器件，可以将 UART 消息转换为 PC 所需的 USB 格式。

使用此协议的缺点在于：SCI 和 UART 外设的最大传输速度相对较低，这会影响时间关键型应用中的性能。在这种情况下，可以利用快速串行接口 (FSI) 外设来高速记录数据。部分 C2000 器件具有 FSI 外设，可用于此目的。在 C2000 实时微控制器外设用户指南中查找 C2000 器件，以确定可用的外设和最大速度能力。请注意，MCU Control Center 需要使用 Sysconfig GUI，只有第 3 代和更高版本的器件提供了该 GUI，因此这些器件是唯一可与此工具配合使用的器件。

备注

虽然 F2838x 有 UART 模块，但该模块只能通过器件上的 CM 内核访问。Sysconfig 不支持 CM 内核，因此 MCU Control Center 不能与 F2838x 上的 UART 外设配合使用。

2.1 设置 #1

仅当数据记录（主）器件具有 USB 外设时，才能执行第一种硬件设置。在此设置中，无需桥接器件，因为器件可以使用 USB 协议将数据直接发送到 PC。总体而言，这是最简单的硬件设置，因为除了主器件之外不需要额外的 MCU。

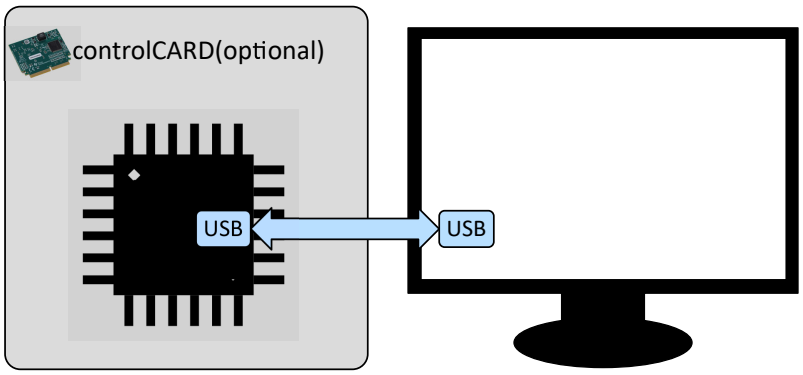


图 2-1. 硬件设置 #1

2.2 设置 #2

第二种硬件设置是一种非常常见的应用场景，可在所有第 3 代和更新的 C2000 器件上执行。此设置使用 SCI 或 UART 外设从器件导出数据，并使用桥接器件将这些消息转换为 PC 可以理解的 USB 协议。若使用 LaunchPAD 或 controlCARD，则主器件和桥接器件均已存在并正确接线。板载桥接器件通常已刷写一个程序，可执行必要的协议转换（在大多数情况下，这是 XDS110 程序）。因此仅需配置主器件上的 SCI/UART 代码，该代码可通过 MCU Control Center 工具自动配置。有关更多详细信息，请参阅后续章节。

请注意，若用户没有 LaunchPAD 或 controlCARD，只有一个独立器件（例如 C2000 器件已焊接到定制板上），则需要一个额外的桥接器件。由于嵌入式开发中经常会使用这类桥接器件，市场上有多种低成本硬件方案可供选择（既有来自自德州仪器 (TI) 的，也有来自第三方供应商的）。

连接独立（无 USB）器件的另一种选择是购买一台带 USB 模块的 C2000 器件，使用 MCU Control Center 生成并刷写桥应用程序。

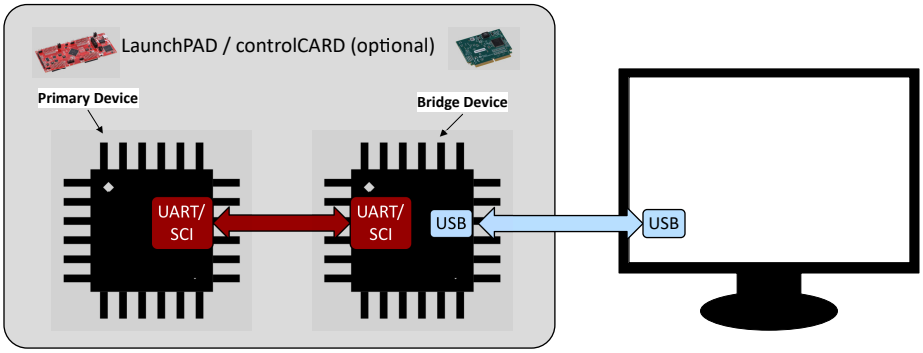


图 2-2. 硬件设置 #2

2.3 设置 #3

仅当数据记录（主）器件有 FSI 外设，而且桥接器件同时具有 FSI 和 USB 外设时，才能采用第三种硬件设置。此设置使用独立器件上的 FSI 外设从器件导出数据，并使用桥接器件将这些消息转换为 PC 可以理解的 USB 协议。如果为桥接器件使用了 controlCARD，则 USB 信号已正确连接至可插入 PC 的 USB 连接器。主器件上的 FSI 代码和桥接器件上的 FSI 转 USB 代码都需要配置，两者都可以通过 MCU Control Center 工具在两个单独的 Sysconfig 工程中自动配置。有关更多详细信息，请参阅后续章节。

请注意，如果用户没有 controlCARD，只有一个具有 FSI 和 USB 外设并可用作桥接器件的独立器件，则需要使用 USB 连接器电路连接到 USB 端口。

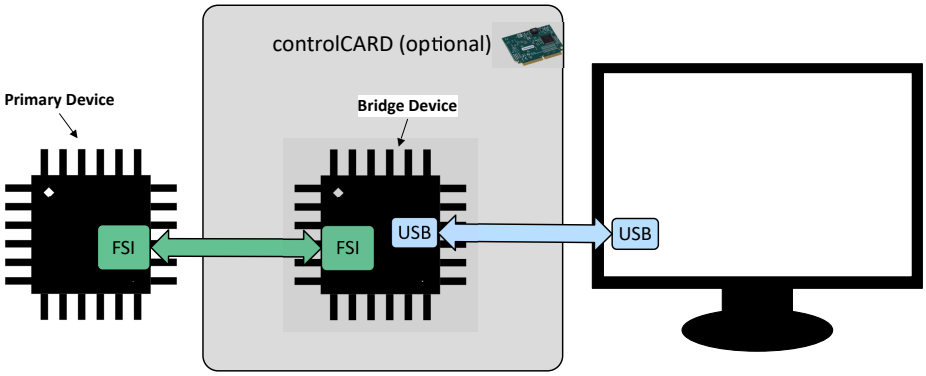


图 2-3. 硬件设置 #3

2.4 设置 #4

仅当数据记录（主）器件和第一个桥接器件均具有 FSI 外设时，才能执行第四种硬件设置。此设置采用 FSI 外设从独立器件导出数据，通过第一个桥接器件将这些消息转换为 UART 协议，再通过第二个桥接器件将消息从 UART 协议转换为 PC 可以理解的 USB 协议。

若将 C2000 器件的 LaunchPAD 或 controlCARD 与 FSI 外设配合使用，则两个桥接器件均已存在并正确接线。板载的第二个桥接器件通常已刷写一个程序，可在 UART 与 USB 间执行必要的协议转换（在大多数情况下，这个程序为 XDS110 调试程序）。因此仅需配置主器件上的 FSI 代码以及第一个桥接器件上的 FSI 转 UART 桥接代码。两种器件的配置都可以通过 MCU Control Center 工具自动完成（在两个单独的 Sysconfig 工程中）。本文后续章节将提供更多详细信息。

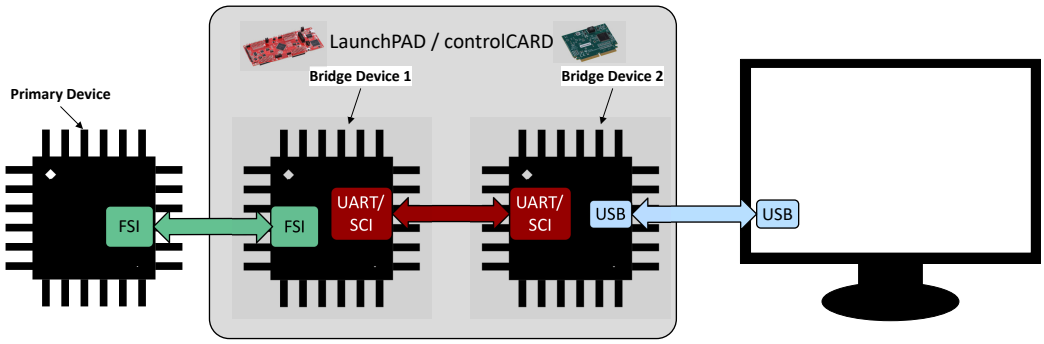


图 2-4. 硬件设置 #4

3 软件层

本节介绍 MCU Control Center 工具的不同特性实现的软件抽象层。添加和配置 MCU Control Center 模块时，Sysconfig 会自动生成这些文件。

表 3-1. SysConfig 生成的文件

记录层	说明	涉及的 C 头文件
用户应用程序	该层提供了最高的抽象级别。其中描述了要在用户应用程序代码中针对每个特性调用的顶层 API 函数。（应用程序记录、通信记录器、桥和快速实时记录）	logger/coms_logger.h export/export_log.h logger/rt_log.h bridge/bridge.h
封装	该层先对日志进行封装，再将其从器件发出。可用的封装层为 JSON 和 START/END。	export/export_package.h
缓存器	在发出日志消息之前，该层为其提供缓冲器。这样就可以灵活地在两个不同的代码区域中采集和发送数据。	export/export_buffer.h
出口	该层可以通过将数据写入相应外设的缓冲器，从而通过通信外设导出数据。	export/export.h

4 GUI 创建

GUI 层利用 PC 上的 GUI Composer 并可在 CCS 环境中运行。有关 GUI Composer 的更多信息，请参阅 GUI Composer 文档。在将 MCU Control Center Sysconfig 模块添加到工程中后，将生成以下文件以创建 GUI Composer 应用程序。

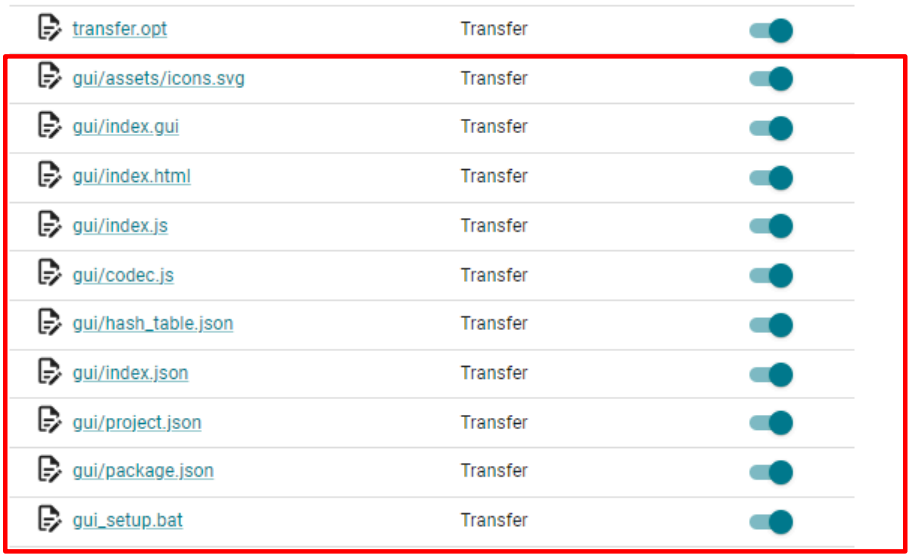


图 4-1. SysCfg 生成的 GUI 层文件

为了针对 CCS 生成自定义 GUI 应用程序，需要执行以下构建后步骤。导航至特定工程的 *Properties -> General -> Variables*，添加一个名为 *GUI_SUPPORT* 的变量。

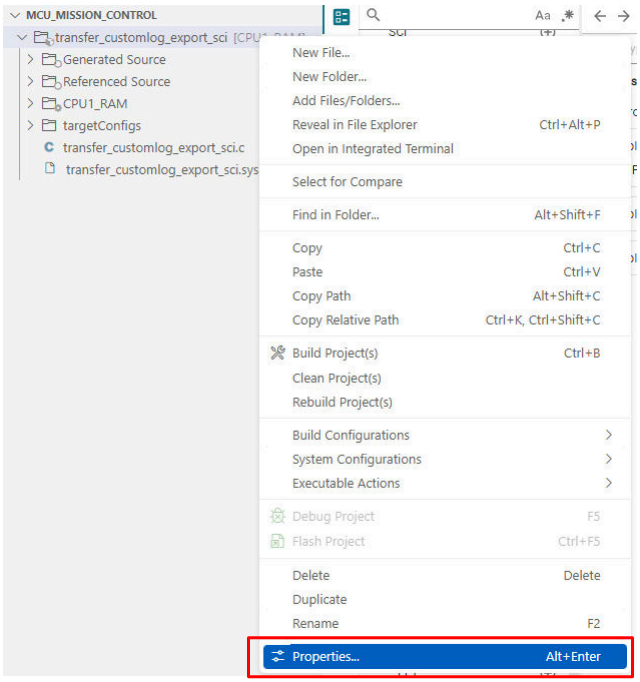


图 4-2. 打开工程属性

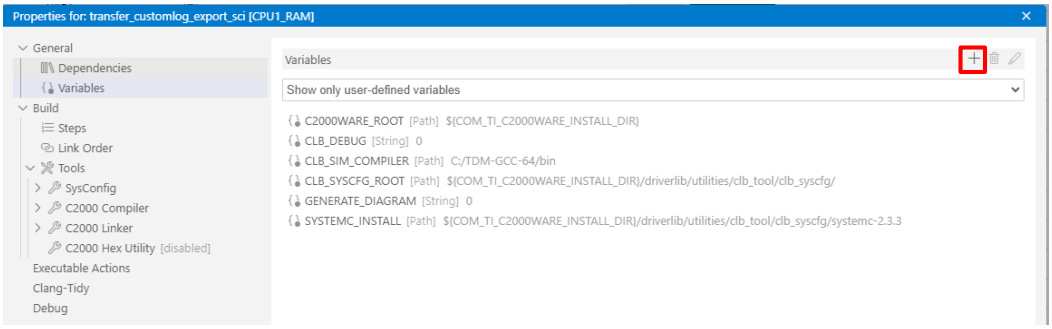


图 4-3. 添加 CCS 变量

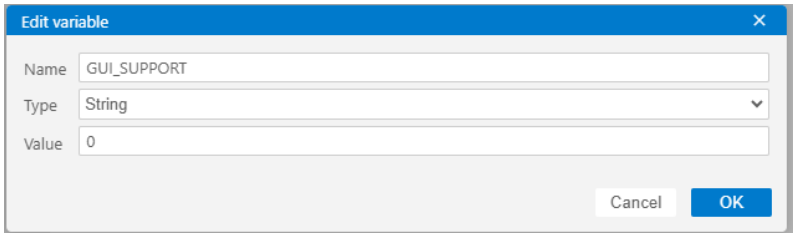


图 4-4. GUI_SUPPORT 变量已添加

此外，在工程属性中，通过转到 *Build* → *Steps*，将以下行添加到构建后步骤：

```
if ${GUI_SUPPORT} == 1 ${BuildDirectory}\syscfg\gui_setup.bat
```

在构建工程之前，请确保 GUI_SUPPORT 变量设置为 1。这会在每次工程构建结束时，将 Sysconfig 自动生成的 GUI 文件复制到 CCS 安装文件夹中的合适位置。放入该文件夹中后，即可直接从 CCS 环境成功启动 GUI 应用程序。构建后，自动生成的 GUI 工程名称与下方 Sysconfig 设置中选择的名称相匹配。

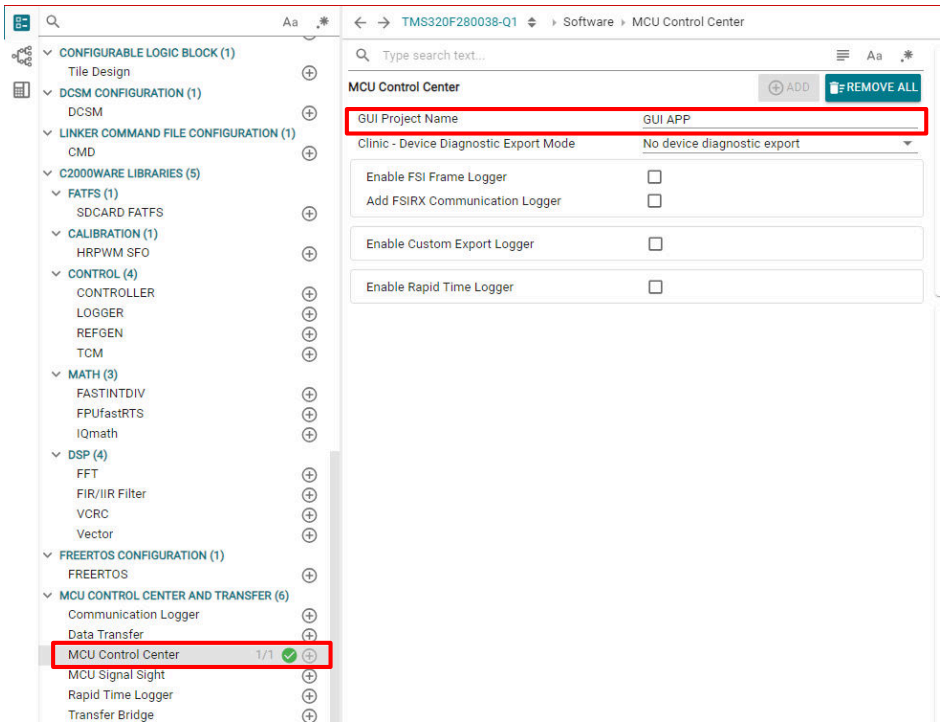


图 4-5. 插件名称

构建工程后，刷新 CCS 视图，可以在 CCS 插件下拉列表中看到新生成的 GUI。转至 *Help* → *Reload Window*。

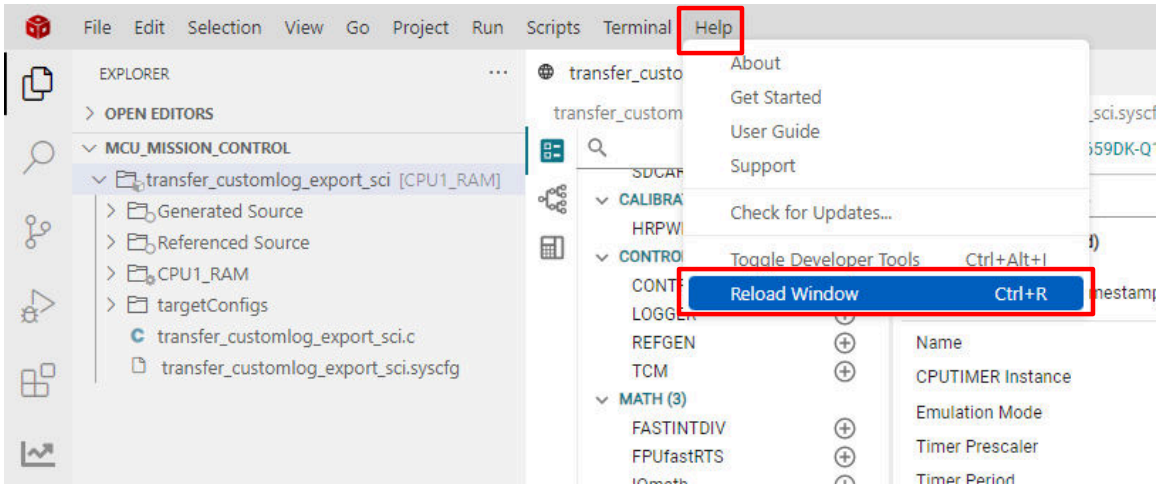


图 4-6. 重新加载窗口以查看生成的插件

要打开集成的 GUI 应用程序，请转至 *View → Plugins → {NameOfGUI}*。点击后，GUI 就会在 CCS 中打开。

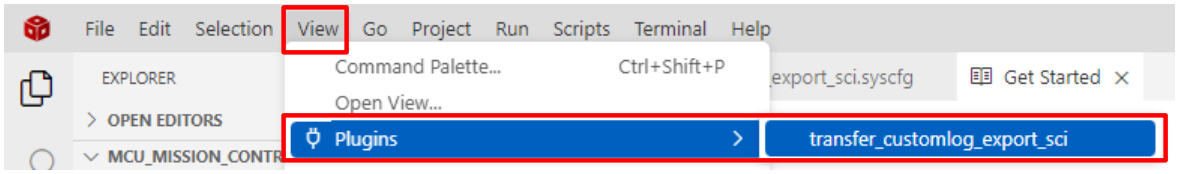


图 4-7. 查看 GUI

5 应用程序记录

在只需要 MCU 来导出数据的应用程序中，可以实现应用程序记录。此特性可将字符串、变量值或数组导出至 PC 或桥接器件。应用程序记录的原理基本类似于 *printf()* 语句，但它使用器件上的通信外设（如 SCI/UART）来发送打印消息，而不使用 JTAG。与常规 *printf()* 调用相比，*EXPORTLOG_log()* 函数支持更高的传输速度，并且无需连接调试器即可查看来自器件的调试数据。

此特性可直接与 节 2.1 或 节 2.2 中的设置配合使用，或者对于 节 2.3 和 节 2.4，可利用 节 6 中的模块用于已配对的桥接器件。

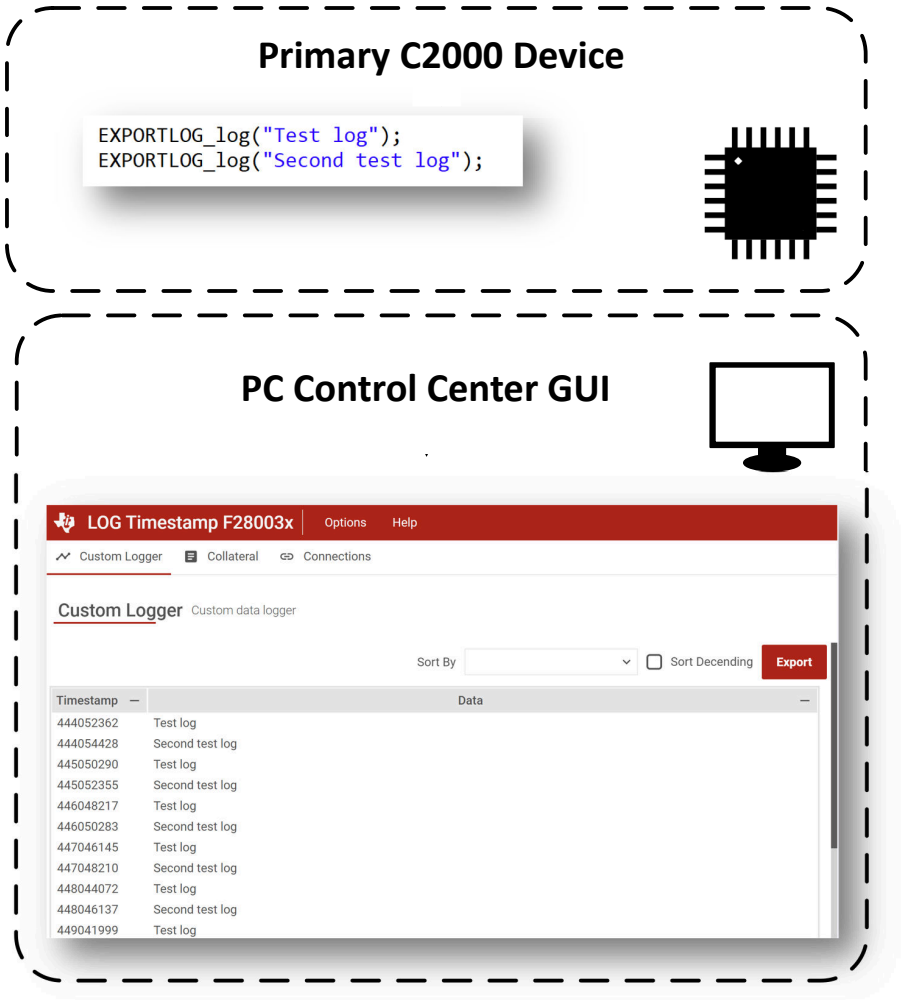


图 5-1. 应用程序记录示意图

5.1 o 应用程序记录操作指南

下面简要概述了 表 3-1 中的应用程序记录实现及其所涉各层。利用应用程序记录，用户可打印字符串及各种数据类型（如有符号、无符号整数和浮点数）的数组。

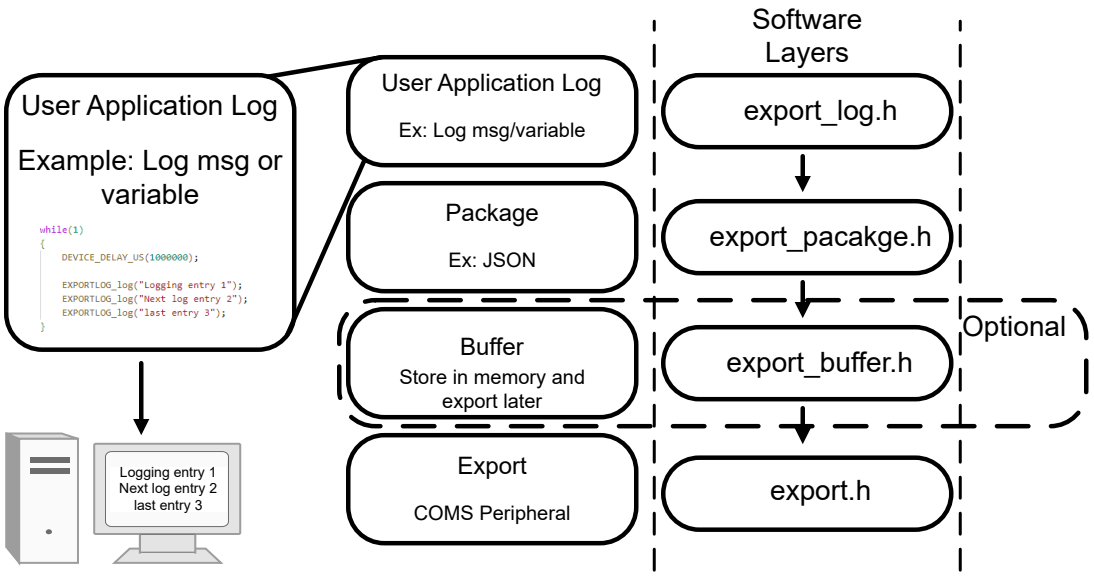


图 5-2. 应用程序记录器简要示意图

通过以下步骤，可轻松将此特性添加到 CCS 工程中。

SysConfig 配置

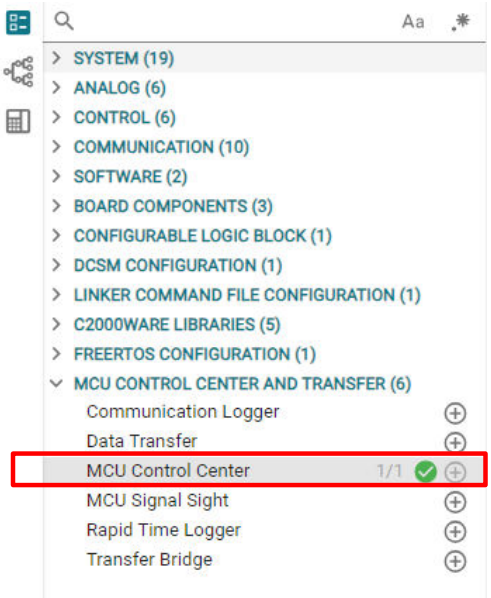


图 5-3. 添加 MCU Control Center 模块

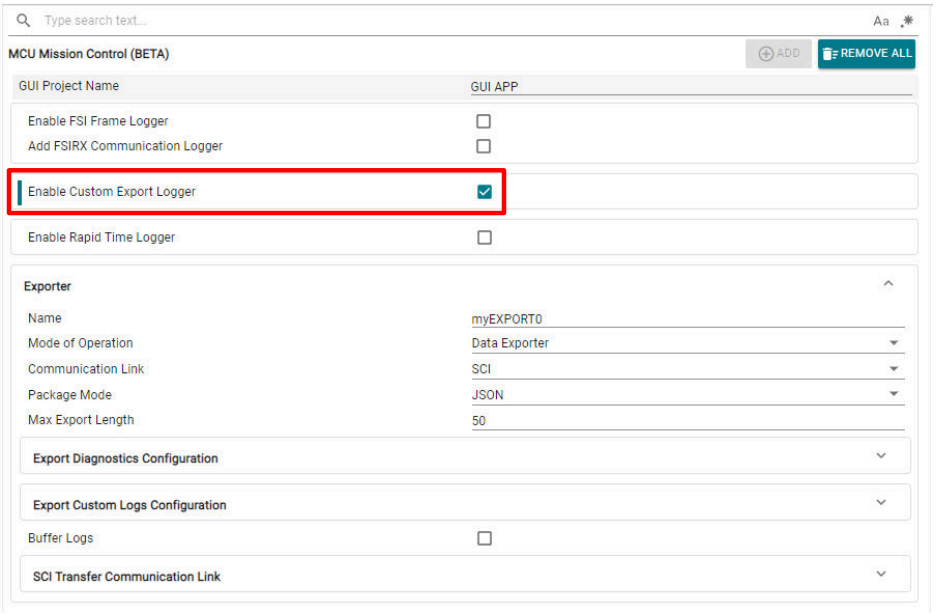


图 5-4. 自定义导出记录器配置

为所需通信外设配置导出器模块设置。在本操作指南示例中，SCI 用作通信外设。在 *Transfer Communication Link* 设置下，SysConfig 会自动为器件添加一个可用的通信外设，并提供可编辑的默认配置。

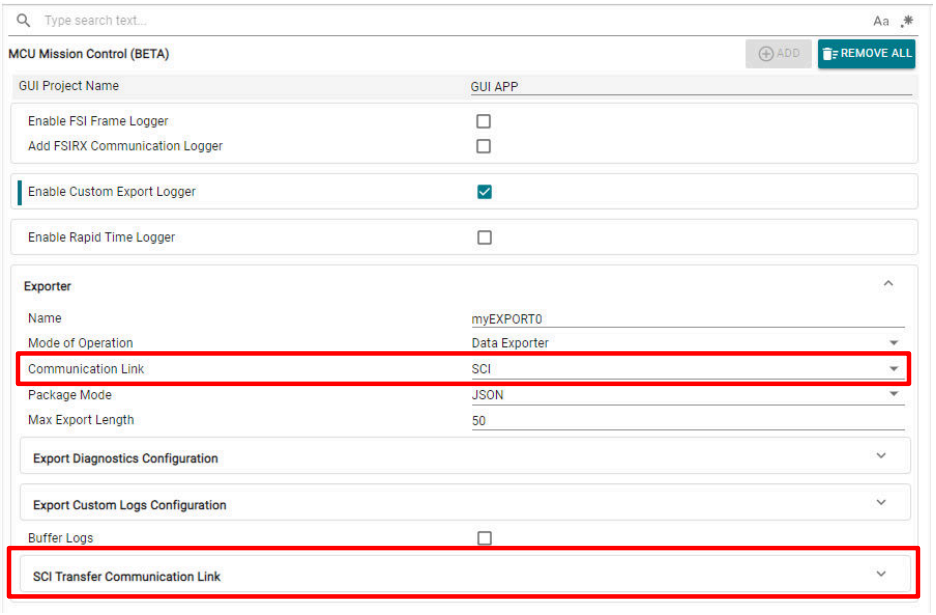


图 5-5. 导出子模块配置

下图展示了添加导出器子模块时所有相关的可配置通信外设设置。

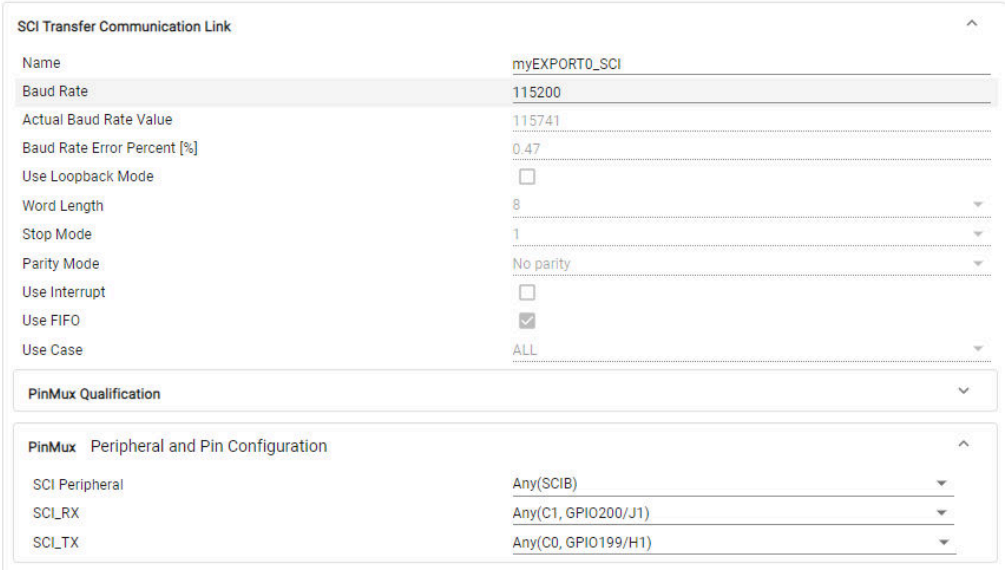


图 5-6. 传输通信链路

可添加额外设置来增强数据日志采集功能，这些设置位于 Export Custom Logs Configuration 下。勾选 Export Log Support 复选框会生成可在导出文件或 export_log.h 生成的文件下使用的函数。

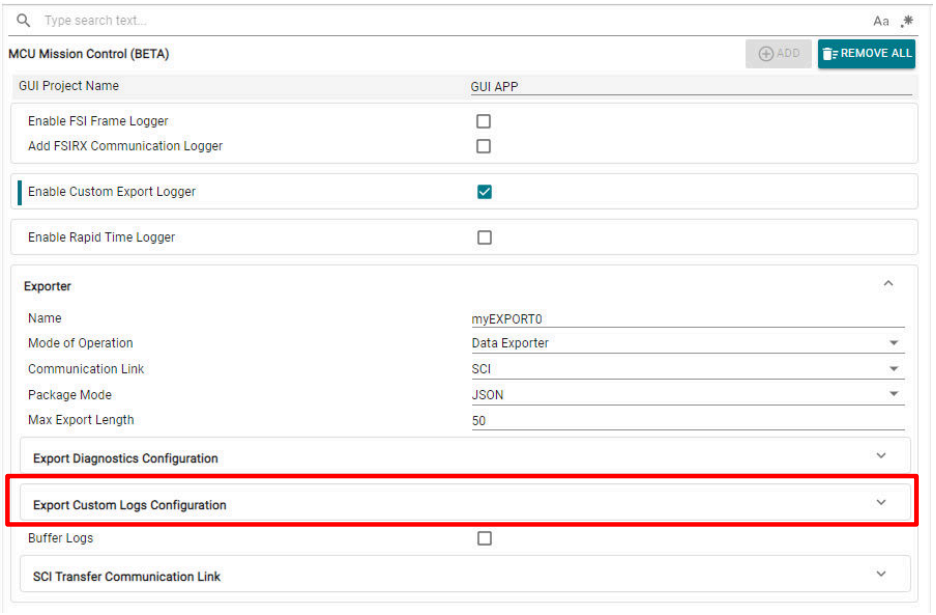


图 5-7. 导出自定义日志配置

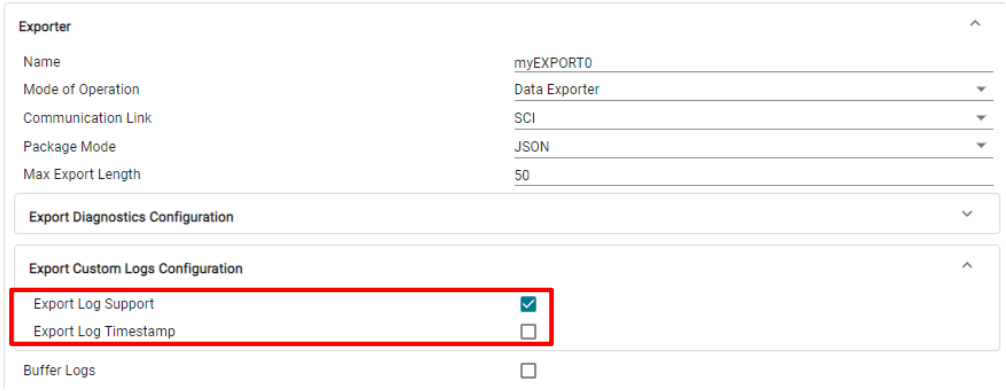


图 5-8. 导出日志支持和时间戳

导出日志时间戳是另一个特性，可添加以记录某条日志的采集时间。如果启用了导出日志时间戳，SysConfig 会自动添加一个 CPU 计时器模块供配置。

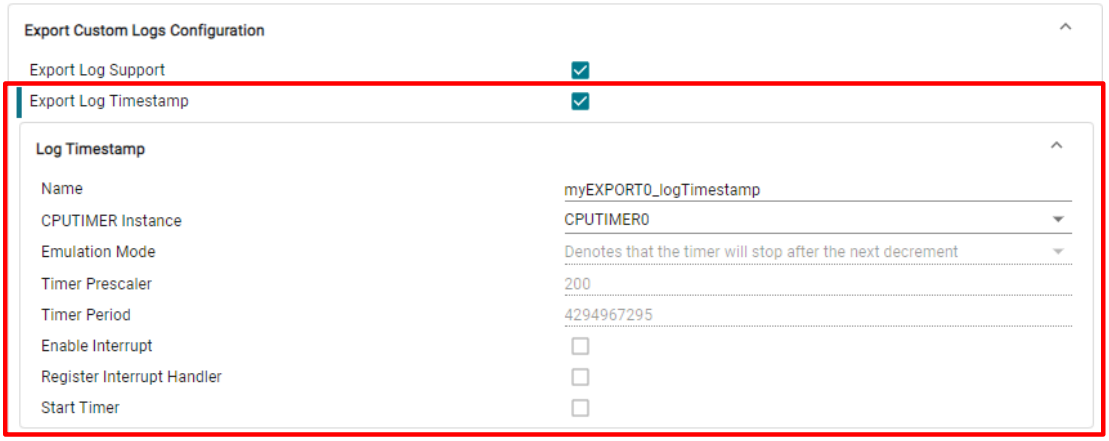


图 5-9. 导出日志时间戳配置

应用代码

这些是需要为 GUI 和数据记录软件层添加的所有模块。要实现对字符串或变量的数据记录，唯一所剩的步骤就是在应用程序代码中调用 export/export_log.h 层辅助函数。确保在主代码顶部包含 export/export_log.h 文件。

头文件

```
//
// Included Files
//
...
#include "export/export_log.h"
```

构建工程之前，请务必按照 节 4 启用 GUI_SUPPORT。启用该功能后，构建工程并在 CCS 中打开 GUI。

应用程序代码记录

```
EXPORTLOG_log("Logging entry 1");
EXPORTLOG_log("Next log entry 2");
EXPORTLOG_log("last entry 3");
```

GUI 的 COM 端口设置

加载 .out 文件，但尚不运行。转至 Options → Serial Port Settings → <COM Port>，验证为 GUI 选择了正确的 COM 端口

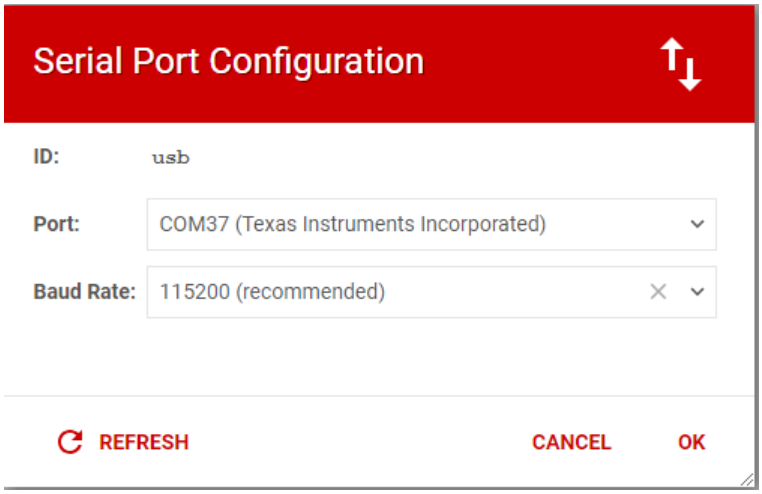


图 5-10. 串行端口设置

最终输出

选择正确的 COM 端口后，运行应用程序代码。下图显示了最终输出。用户可以修改这些打印消息，也可以根据特定应用程序和调试场景的需求，移动在应用程序中调用 EXPORTLOG_log() 函数的位置。有关 C2000ware SDK 提供的其他应用程序记录示例的介绍，请参阅 节 9。

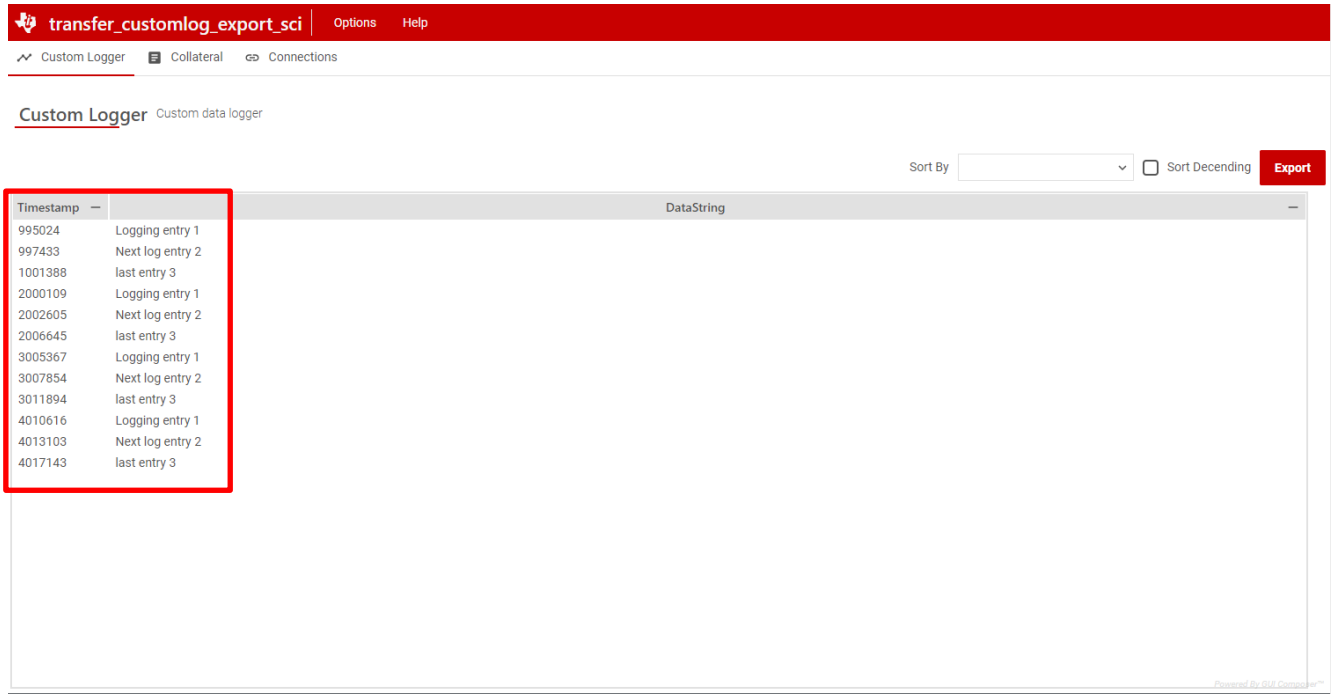


图 5-11. 记录器输出

6 传输桥

当从主器件导出的数据未使用 PC 的支持的通信协议 (USB) 时，可通过一台（或一连串）桥接器件实现数据包至 USB 协议的转换。在 LaunchPAD 和 controlCARD 上常见将 UART 协议转换为 USB 协议的桥接器件，许多第三方厂商也提供了此类产品。但对于 节 2.3 或 节 2.4 所示的设置，需要采用不同类型的桥接固件：FSI 转 USB 或 FSI 转 UART 桥接固件。此时可使用 MCU Control Center 工具提供的传输桥模块生成代码，以接收快速通信外设数据包，缓冲该数据，继而通过另一通信外设（UART 或 USB）传输该数据。在桥接器件上缓冲数据，既保障了数据从主器件高速导出（不会对主应用程序性能造成重大影响），又能以较低速率（经 UART 转 USB 或直接通过 USB）将数据传输到 PC。

图 6-1 直观展示了这一概念。通信链路 B 代表桥接器件接收数据所用的通信外设（通常为 FSI），通信链路 A 代表将数据从桥接器件传至 PC 所用的通信外设（通常为 UART 或 USB）。可以选择将缓冲器置于两条通信链路之间。请注意，此特性仅提取通信链路 B 所收数据包的载荷并直接转发至通信链路 A，传输桥不执行任何额外数据处理。

下图展示了传输桥特性与 节 2.3 所示设置的配合使用。若采用 节 2.4 所示配置，则需在桥接器件与 PC 之间增配一个 UART 转 USB 桥接器件。

此特性可与 节 2.2、节 2.3 及 节 2.4 所示设置配合使用，亦可与各种不同应用中的多种其他自定义硬件设置配合使用。

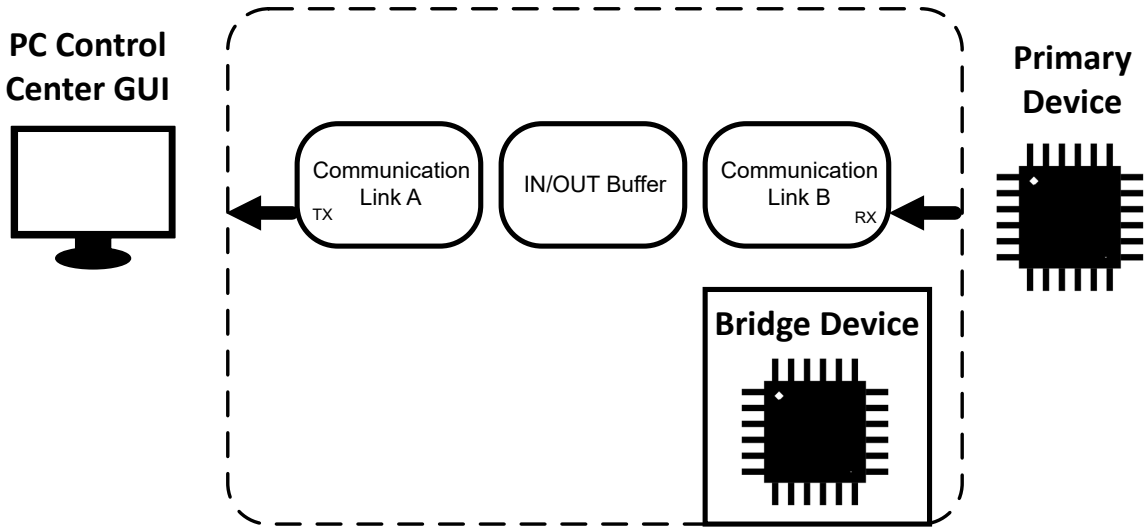


图 6-1. 传输桥示意图

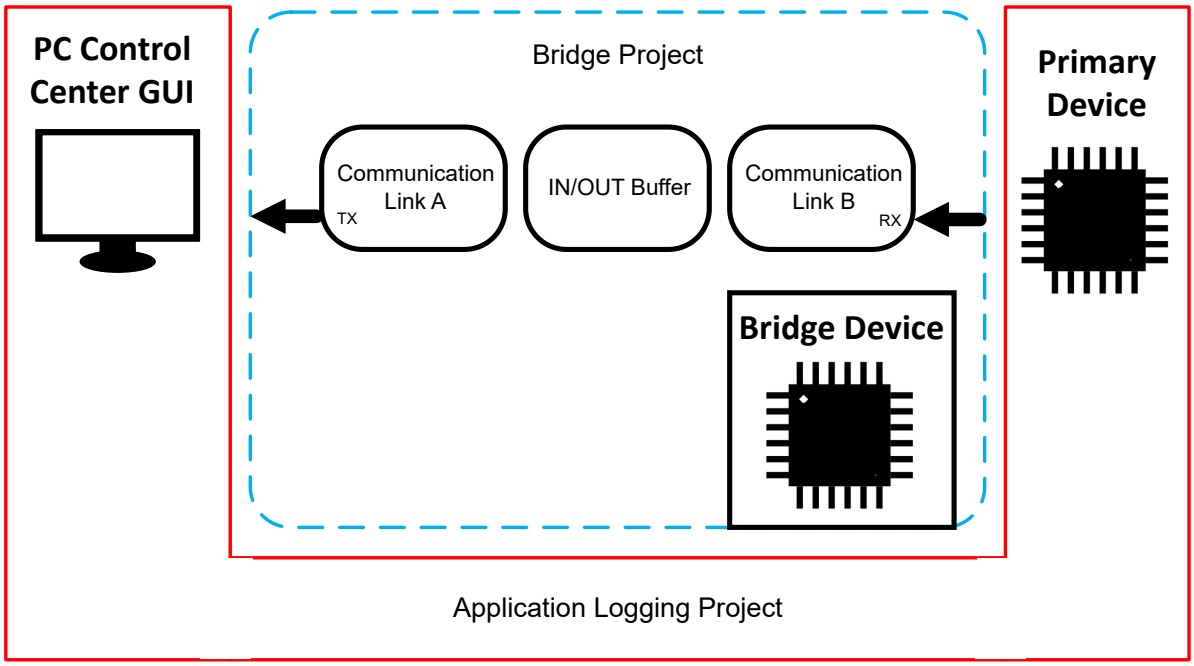
虽然 FSI 转 UART 桥是常见的应用场景，但传输桥特性也可用于为其他桥类型生成固件，这对于各种应用和硬件设置选项非常有用。下表列出了该工具当前支持的所有通信链路组合。其中一部分已用在本文重点介绍的硬件设置中，其余部分则更适合特定应用。

表 6-1. 支持的通信链路组合

通信链路 B (RX)	通信链路 A (TX)	相应硬件设置
FSI	SCI (UART)	#4
FSI	USB	#3
FSI	SPI	不适用 - 特定于应用
SCI (UART)	USB	#2
SCI (UART)	SPI	不适用 - 特定于应用
SPI	SCI (UART)	不适用 - 特定于应用
SPI	USB	#2 但使用 SPI 消息

6.1 传输桥操作指南

下面简要介绍了 表 3-1 中的桥工程及其所涉各层。传输桥模块支持用户通过一个通信外设接收数据，通过另一个通信外设传输该数据。其设计初衷是让桥接器件实现两个端点间发送的数据的通信协议转换（例如将 FSI 数据帧转换为 UART）。在某些应用场景中，桥接器件可作为缓冲器，协调高速串行通信协议与低速通信协议器件间的数据传输。以下操作指南介绍了添加此特性所需的步骤。



在此图中，蓝色虚线框内为传输桥工程，用于将数据从一种通信协议转换为另一种通信协议。通信链路 B 收到数据后，通过通信链路 A 传输数据。红色框内为主器件工程，使用了 节 5.1 中所示的相同软件实现，但选择了 FSI 而非 SCI 作为通信链路。本操作指南仅重点介绍如何配置桥工程部分。

图 6-2. 传输桥工程简要示意图

软件层

以下是桥工程所需的软件层。请注意，可以选择向桥工程添加一个缓冲器层，以增加器件在导出收到的数据前用于存储数据的存储器。

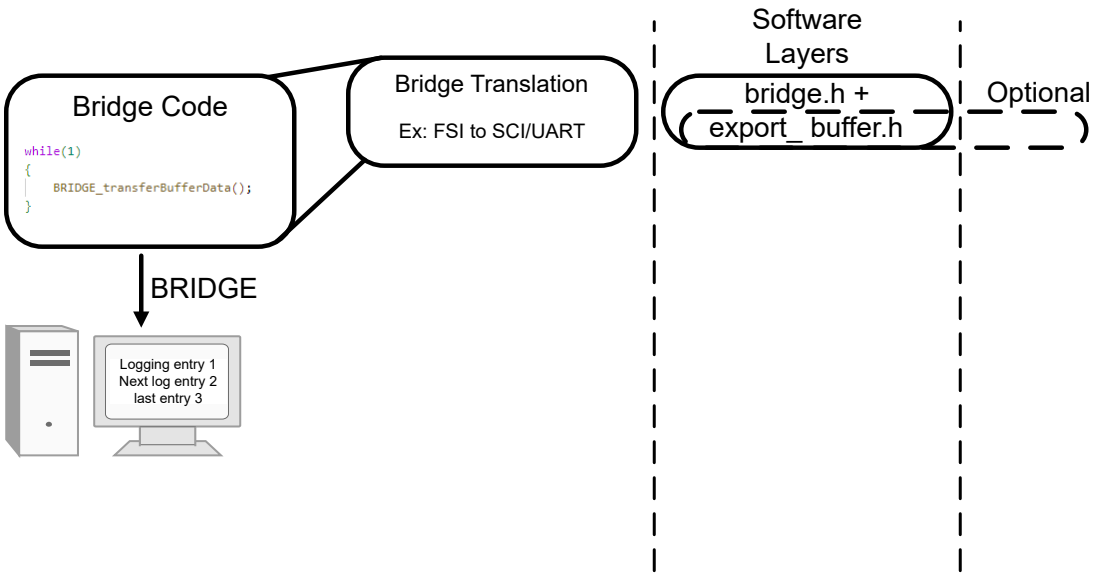


图 6-3. 桥工程软件层

SysConfig 配置

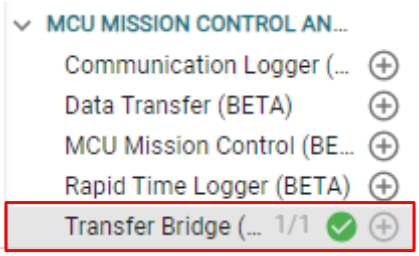


图 6-4. 传输桥模块

下图显示了在传输桥 Sysconfig 模块中自动添加的通信链路 A 和 B。通信链路 B 使用协议 B 接收传入的数据并存储至缓冲器（可选）。通信链路 A 读取收到的数据（来自通信链路 B 或缓冲器），然后通过协议 A 向外传输。本操作指南以典型应用场景为例：桥接器件接收 FSI 协议数据包，并通过 SCI 外设（UART 协议）进行传输。

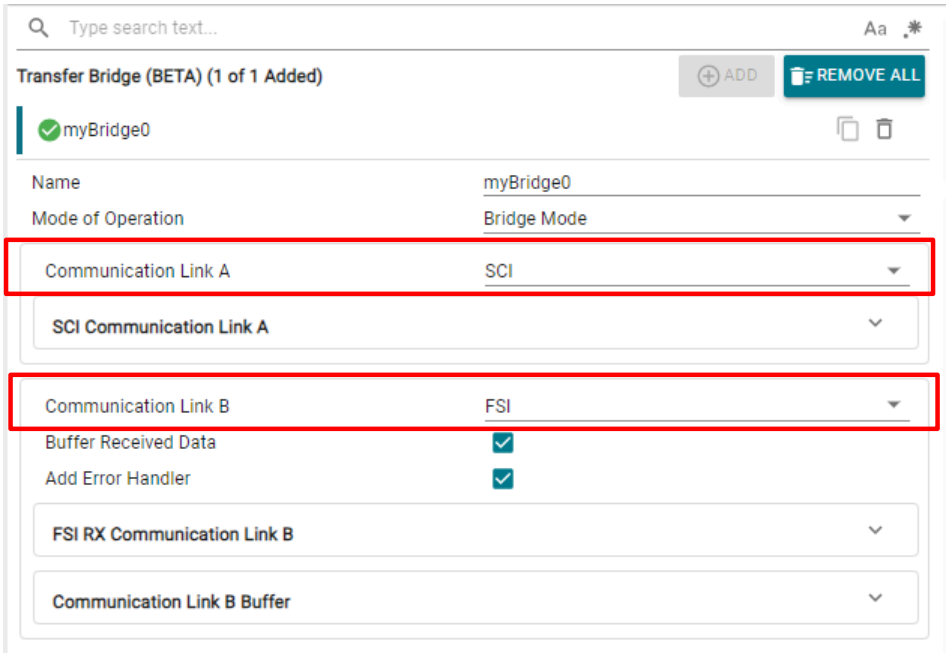


图 6-5. 传输桥通信链路

以下设置是可选的，可用于在桥接器件上添加缓冲器。该选项允许器件在缓冲器满载前积累更多数据再发送。当记录器件向桥接器件发送数据的速度高于桥接器件的处理速度时，此特性很有用。

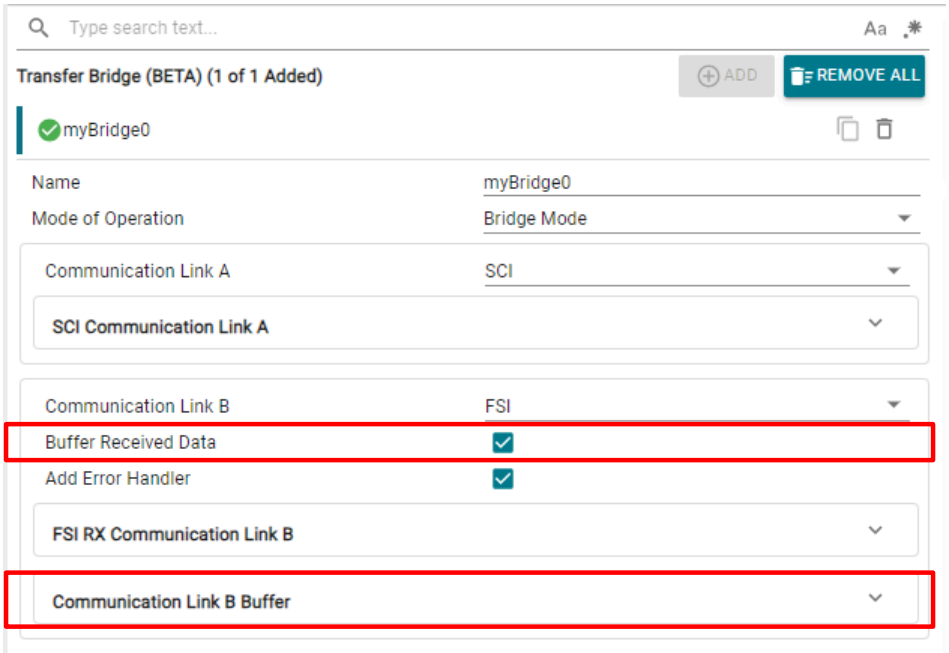


图 6-6. 桥接缓冲器（可选）

头文件

```
//
// Included Files
//
...
#include "bridge/bridge.h"
```

传输桥初始化

只有启用了缓冲器时，才需要执行此步骤。将以下代码添加至主初始化序列中的 *Board_init()* 之后。

```
//
// Logging Inits
//
BRIDGE_init();
```

传输桥应用程序记录代码

最后一步是添加应用程序代码来处理收到的数据包，并将转换后的数据包传输出器件。

```
while(1)
{
    BRIDGE_transferBufferData();
}
```

测试桥接器件

要测试桥接器件是否正常工作，可以执行以下步骤。这些步骤假设有不同的器件负责记录或向桥接器件传输数据。

1. 将传输桥应用程序刷写到桥接器件上
2. 将主器件的 FSITX 引脚连接至桥接器件 FSIRX 引脚
   a. 将主器件 FSITX_CLK 连接至桥接器件 FSIRX_CLK
   b. 将主器件 FSITX_D0 连接至桥接器件 FSIRX_D0
   c. 将主器件 FSITX_D1 连接至桥接器件 FSIRX_D1（可选 - 如果在*帧配置*中配置了双数据通道）
3. 通过 USB 连接器将桥接器件连接至 PC
4. 在主器件上运行应用程序记录工程代码
5. 打开串行端口应用程序，并配置端口设置以匹配 SCI 模块配置

最终输出显示在串行终端端口中。

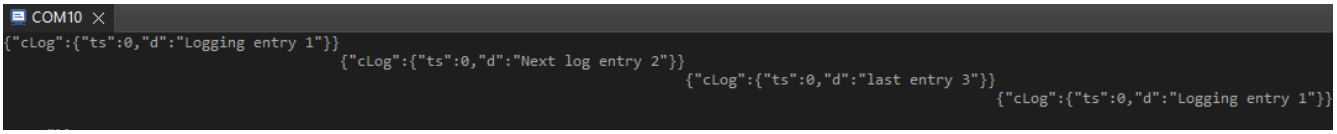


图 6-7. 传输桥最终输出

7 通信记录器

除有效载荷之外，许多通信协议还包含一些有意义的数据。例如，FSI 协议在每帧中包含帧类型、用户数据、CRC 字节和帧标签字段。通信记录器特性提供了桥接软件，该软件可额外从每个收到的数据包中提取所有非有效载荷数据。此特性使桥接器件能够接收任何封装格式的通信外设数据包，并将数据导出到 PC GUI 以进行分析。发送到 GUI 的数据包采用了适当的格式，以便消息的所有组成部分均专用于向用户显示。

尽管应用场景各异，但该特性的一个用途是在应用程序中对 FSI 实现进行调试。在这个案例中，应用程序已在使用 FSI 与另一个 MCU 通信，但存在一些需要调试的通信错误或问题。通信记录器可以在生成的 GUI 内以容易理解的格式显示主器件传输的 FSI 帧的每个部分。

另一个应用场景是使用快速外设进行数据记录，类似于 节 6 特性的应用场景，但 GUI 中记录了更多消息细节。通信记录器特性用于接收和缓冲（可选）来自主器件的高速 FSI 消息，提取消息内容的所有元素，然后通过 UART 或 USB 将其发送到 PC 以在 GUI 中可视化。

下图展示了通信记录器特性与 节 2.3 中所示设置的配合使用。可以注意到，对于此特性，使用了桥接器件 SysConfig 工程来生成 Control Center PC GUI，因为桥接器件与 GUI 之间的数据打包/解包方案必须保持完全一致。如果使用 节 2.4，则需要在桥接器件和 PC 之间连接另一个 UART 转 USB 桥接器件。

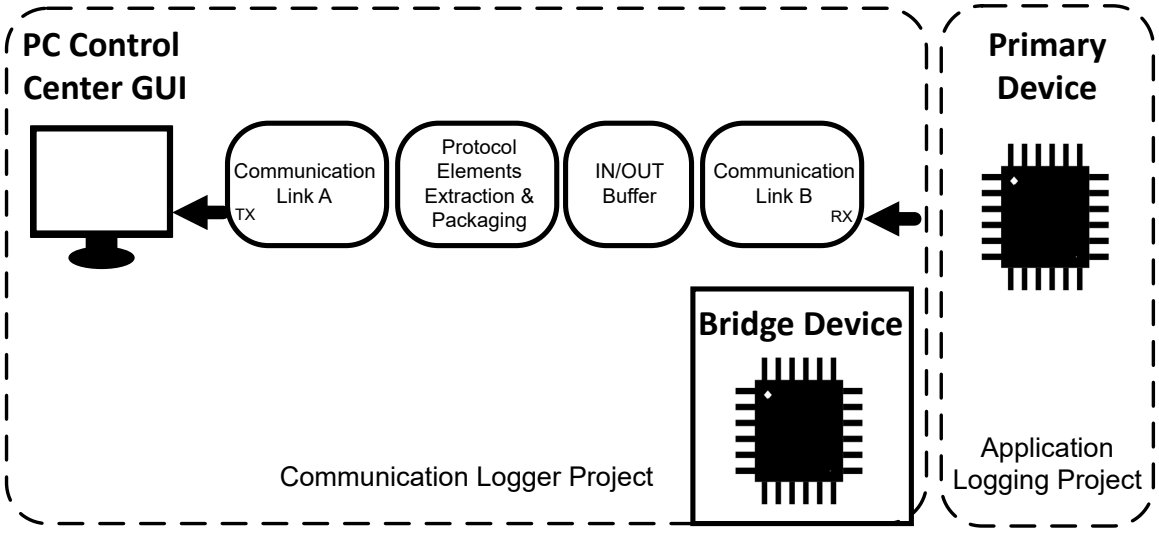


图 7-1. 通信记录器示意图

备注

目前，此特性仅支持 FSI 协议作为传入协议（协议 B）。

7.1 通信记录器操作指南

下面简要介绍了 表 3-1 中的通信记录器特性及其所涉各层。通信记录器特性将收到的 FSI 帧的每个元素显示在 Control Center GUI 中的记录表的单独一列。以下操作指南介绍了将此功能添加到 CCS 工程所需的步骤。

软件层

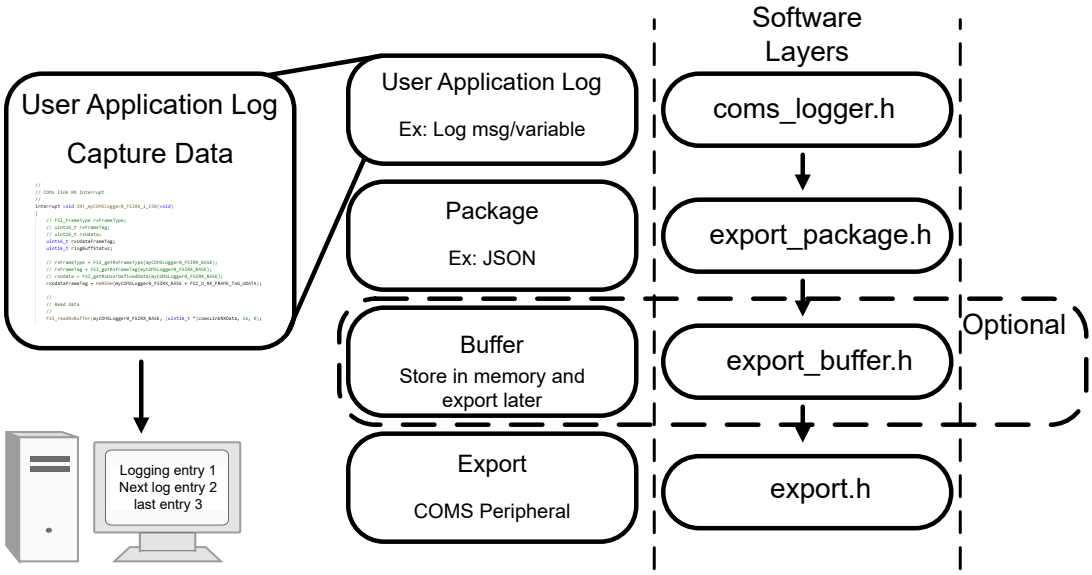


图 7-2. 通信记录器软件层

Sysconfig 配置

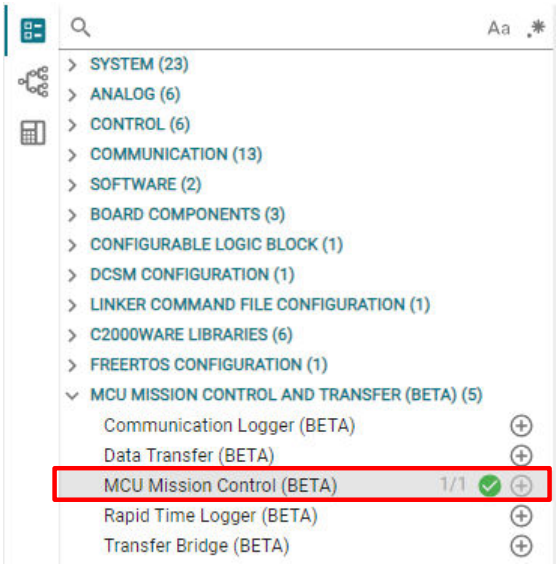


图 7-3. MCU Control Center 模块

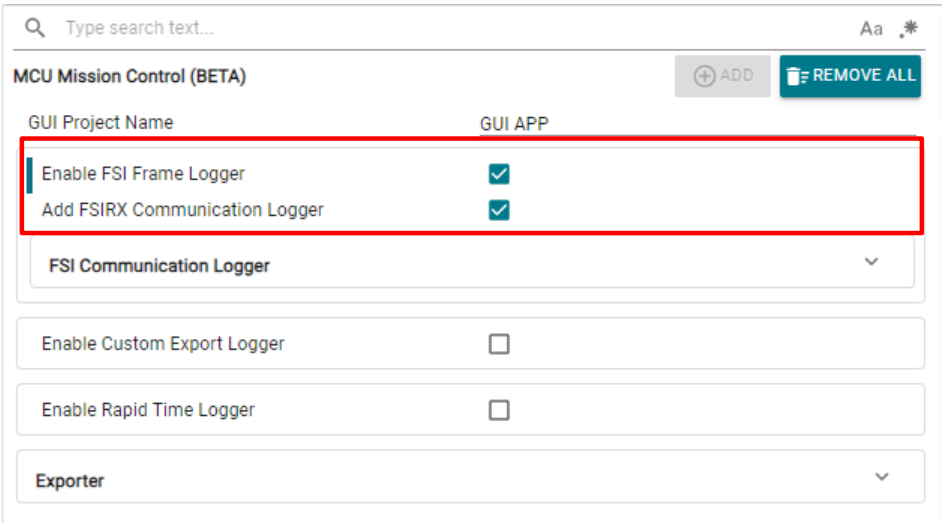


图 7-4. 启用 FSI 记录器和通信记录器

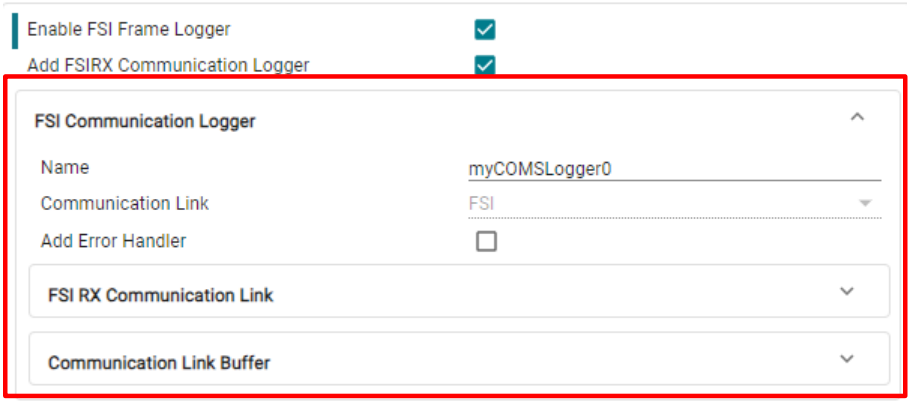


图 7-5. FSI 通信记录器配置

头文件

```
//
// Included Files
//
...
#include "export/export.h"
#include "logger/coms_logger.h"
```

通信记录器初始化

```
//
// Logging Inits
//
EXPORT_init();
COMSLOG_init();
```

通信记录器应用程序代码

```
while(1)
{
    COMSLOG_transferBufferData();
}
```

通信记录器错误处理（可选）

```
void COMSLOG_transferBufferOverflow() {

    //
    // Received too much data too quickly. The transfer buffer overflowed
    // Make the buffer larger
    //
    ESTOP0;

}

void COMSLOG_comsLinkError(uint16_t status) {

    //
    // FSI receive error occurred
    // Bad frames received
    //
    ESTOP0;
}
```

构建工程

构建通信记录器应用程序代码，并确保按照 节 4 中的步骤在 CCS 内生成 GUI。

测试通信记录器工具

1. 将通信记录器应用程序刷写到桥接器件上
2. 将桥接器件的 FSIRX 引脚连接至主器件 FSITX 引脚
   a. 将主器件 FSITX_CLK 连接至桥接器件 FSIRX_CLK
   b. 将主器件 FSITX_D0 连接至桥接器件 FSIRX_D0
   c. 将主器件 FSITX_D1 连接至桥接器件 FSIRX_D1（可选 - 如果在*帧配置*中配置了双数据通道）
3. 使用 USB 连接器将桥接器件连接至 PC
4. 在主器件上运行应用程序记录器应用程序工程
5. 从 CCS 内打开生成的 GUI
6. 最终输出如下所示

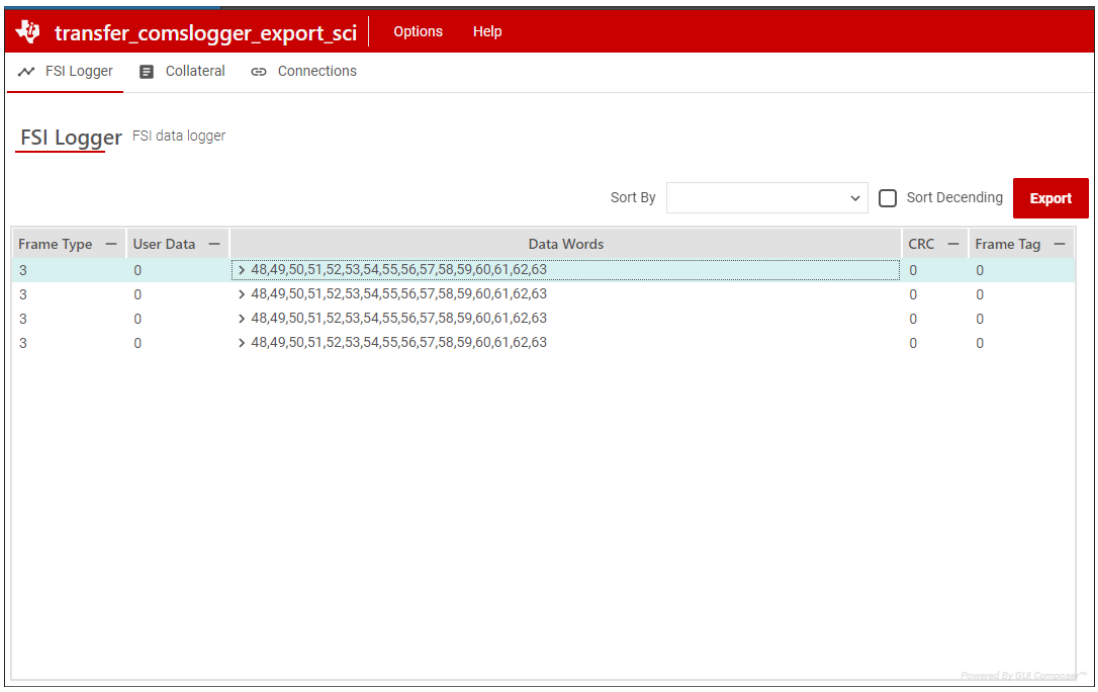


图 7-6. 最终输出

8 快速实时记录器

此前通过通信记录器特性，已可以查看每个 FSI 数据包中提供的附加数据元素。实时快速记录器通过以下方式进一步利用了这些附加元素：使用“用户数据”元素对日志消息类型进行编码，使用“帧类型”元素存储从应用程序发出的变量。这种形式的数据记录速度最快，因为它利用 FSI 协议独有的 FSI 封装特性作为应用程序的封装层。配合 节 2.3 或 节 2.4 所示的设置，可在主器件上使用快速实时记录器特性，同时在桥接器件上实现一种特殊的通信记录器特性。

在主应用程序具有严格的时序要求时，通常利用该特性来调试或查看应用程序变量。通过充分利用主器件上的 FSI 协议特性，可实现快速且优化的数据传输。同时，桥接器件从每个 FSI 元素提取数据并传输至 PC（直接通过 USB 外设传输或使用 UART 通过辅助 UART 转 USB 桥接器件传输）。快速实时记录器 Sysconfig 模块生成一个 JSON 文件供 GUI 应用程序使用，该文件包含每个应用程序变量和消息类型的编码。这种机制使每个数据包都能在 GUI 中有意义地显示，而无需在 FSI 数据包中传输额外字符。

8.1 快速实时记录操作指南

下面简要概述了 表 3-1 中的快速实时记录器特性及其所涉各层。以下操作指南将详细介绍在主器件上启用快速实时记录器特性以及在桥接器件上增强通信记录器特性的操作步骤。

软件层

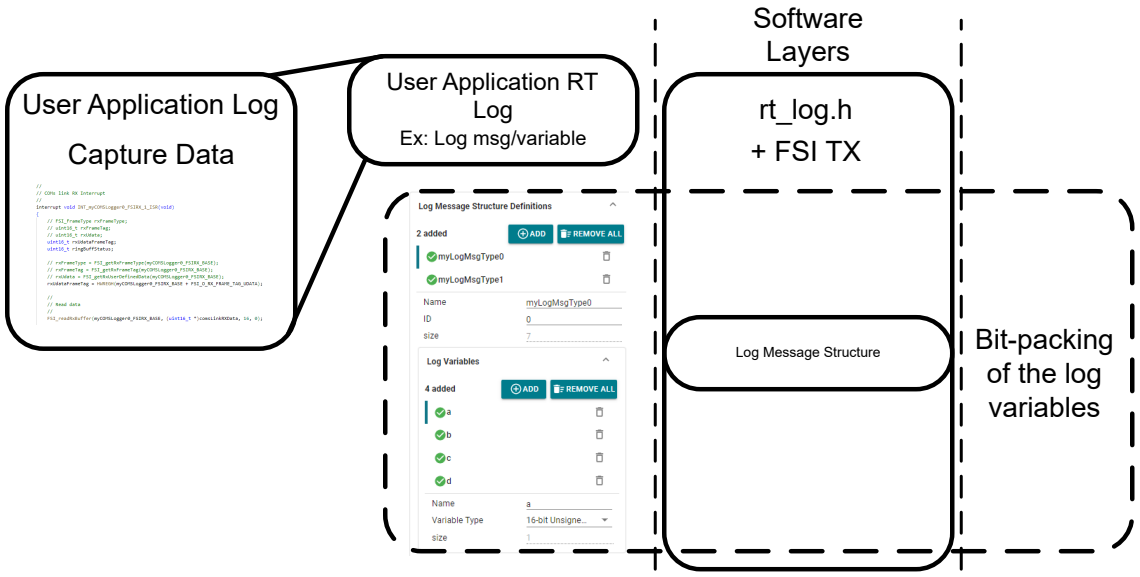


图 8-1. 快速实时记录器软件层

SysConfig 配置

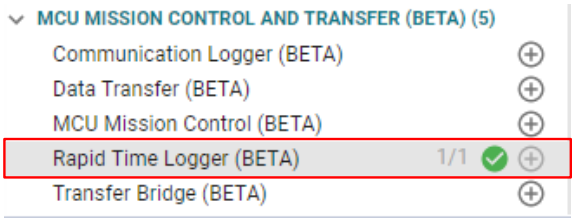
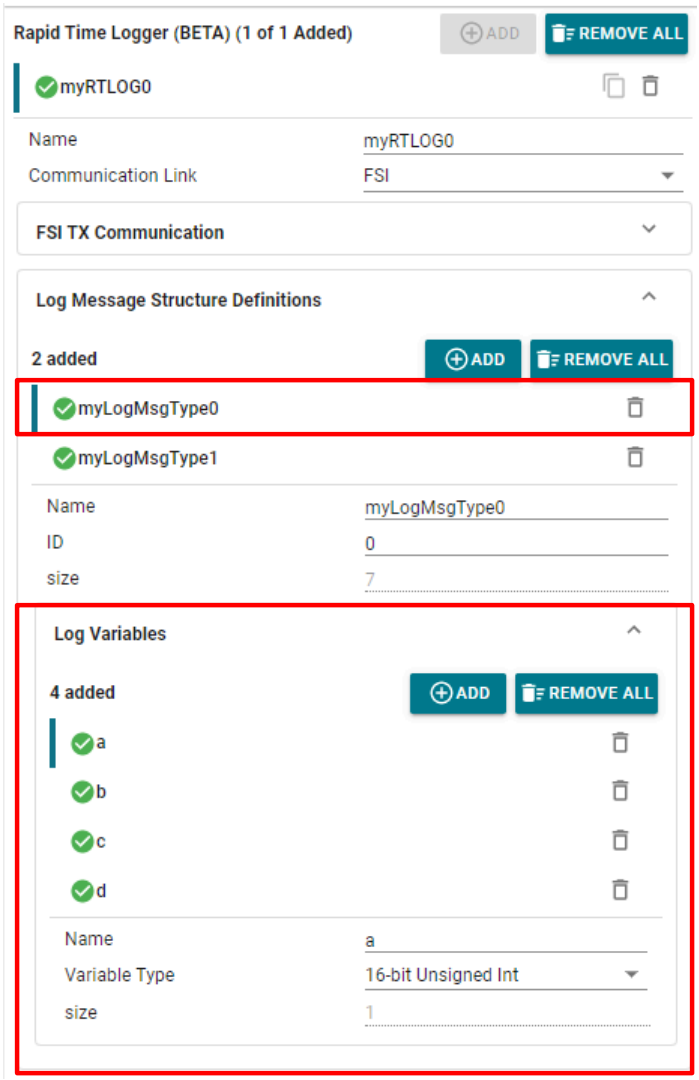


图 8-2. 快速实时记录器 SysConfig

本操作指南在 GUI 中添加了两种示例消息结构。在 *日志消息结构* 视图下，添加了两个实例。每个实例代表数据包中指定的一种特定消息结构。不同的消息结构定义可用于各种特定应用目的（例如：消息类型 0 在应用程序中的某些事件前发送，消息类型 1 在事件后发送）。对于此示例，在第一个日志消息结构定义中添加了四个变量，分别名为 *a*、*b*、*c* 和 *d*。在第二个日志消息结构定义中，添加了另一个名为 *e* 的变量。

表 8-1. 日志变量设置示例

变量	变量类型
a	16 位无符号整数
b	32 位无符号整数
c	32 位浮点数
d	包含 16 位无符号整数的数组 数组长度：2
e	包含 32 位浮点数的数组 数组长度：8



在上图中，每个日志变量可具有唯一的名称、变量类型和大小。根据变量类型自动计算大小。在本操作指南中，根据表格为每种结构类型配置 a、b、c、d 和 e。

图 8-3. 日志消息结构 0 定义

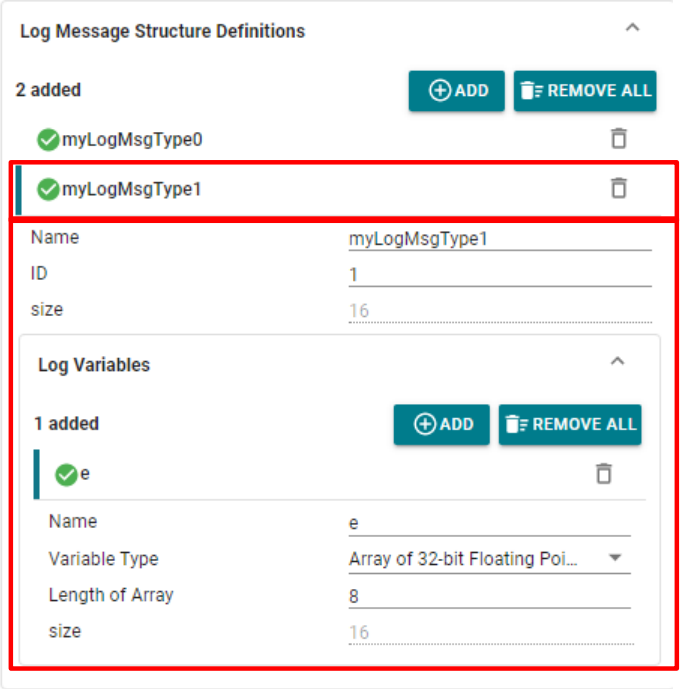


图 8-4. 日志消息结构 1 定义

头文件和全局变量

```
//
// Included Files
//
...
#include "logger/rt_log.h"
uint16_t a = 0;
uint32_t b = 6798004;
float c = -189.4934;
uint16_t d[2] = {19872, 290};
float e[8] = {
                    1243.43, -4399.24, -23.392, 0.0213,
                    -2093, 238.4993, -2390.300, 329.401
            };
volatile uint16_t toggle = 0;
```

快速实时记录器初始化

```
//
// Logging Inits
//
RTLOG_init();
```

在应用程序代码中添加快速实时日志

```
// Insert delay if required for debugging purposes
DEVICE_DELAY_US(1000000);
if (toggle == 0)
{
    RTLOG_writeLog_0(a, b, c, d);
}
else {
    RTLOG_writeLog_1(e);
}
    toggle ^= 1;
```

通信记录器额外步骤

接下来仅需按照 节 7.1 中所述为桥接器件设置通信记录器特性，并执行一些额外步骤。TI 提供了一个 JSON 文件，该文件包含通过 FSI TX 帧发送的所有变量的编码，通信记录器使用这些编码来解码快速实时记录器消息。完成主工程配置并对桥接工程执行 节 5.1 中的步骤后，需执行以下步骤。

1. 在 MCU Control Center SysConfig 模块中选择 *Enable Rapid Time Logger*。

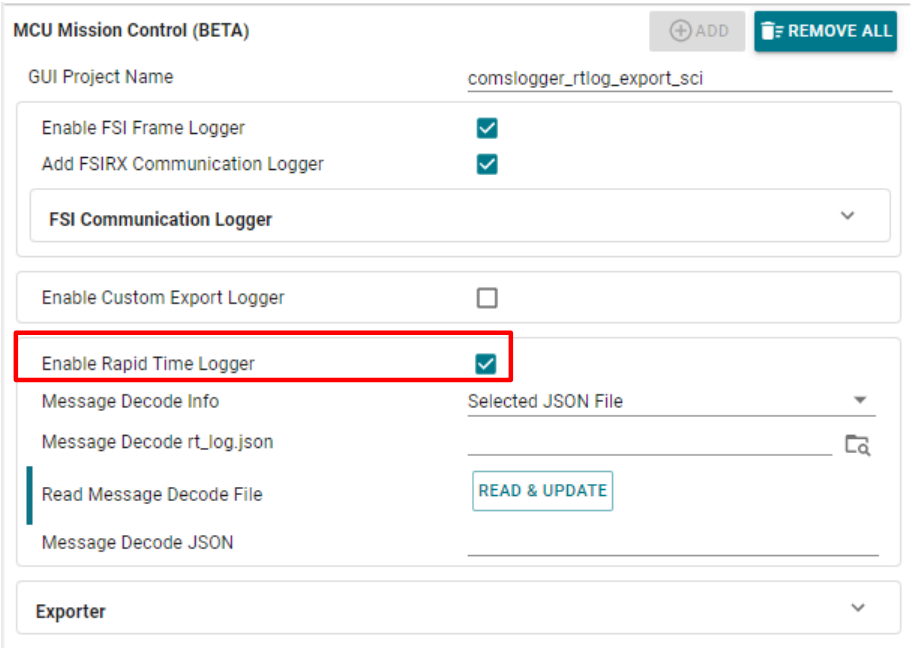


图 8-5. 启用快速实时记录器

2. 在 Rapid Time Logger 工程的 build 文件夹中搜索 rt_log.json 文件。
   a. 例如，<workspace_ccs>/<name_of_project>/<build_folder>/syscfg/logger/rt_log.json

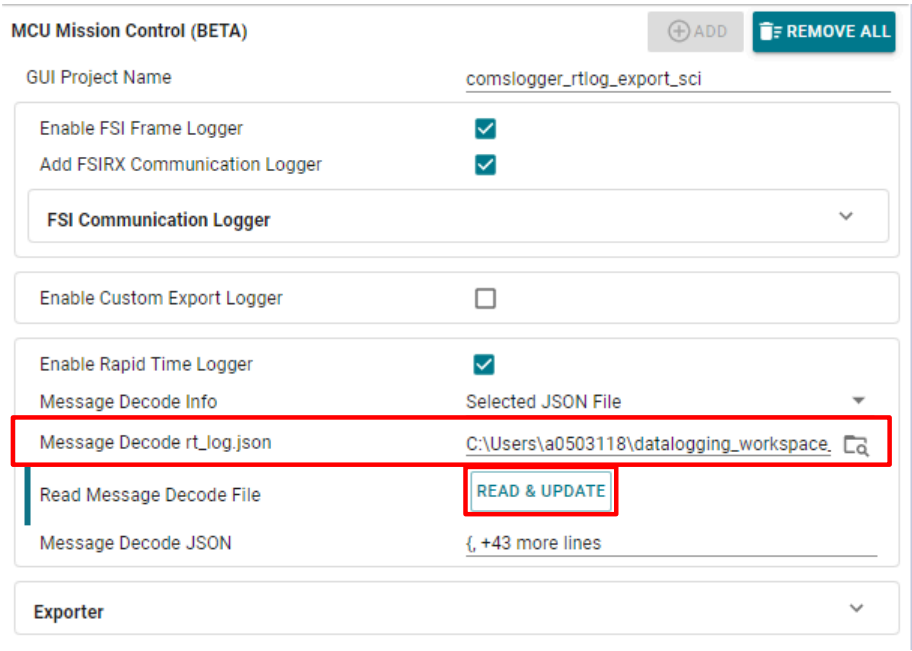


图 8-6. 导航到 JSON rt_log.json 文件

构建工程

构建通信记录器应用程序代码，并确保按照 节 4 中的步骤在 CCS 内生成 GUI。

测试实时记录器特性

1. 将通信记录器应用程序工程刷写到桥接器件上
2. 将通信记录器件的 FSIRX 引脚连接至实时记录器件的 FSITX 引脚
   a. 将主器件 FSITX_CLK 连接至桥接器件 FSIRX_CLK
   b. 将主器件 FSITX_D0 连接至桥接器件 FSIRX_D0
   c. 将主器件 FSITX_D1 连接至桥接器件 FSIRX_D1（可选 - 如果在*帧配置*中配置了双数据通道）
3. 使用 USB 连接器将桥接器件连接至 PC
4. 在主器件上运行快速实时记录器应用程序工程
5. 在 CCS 内打开生成的 GUI
6. 最终输出如下所示

最终输出

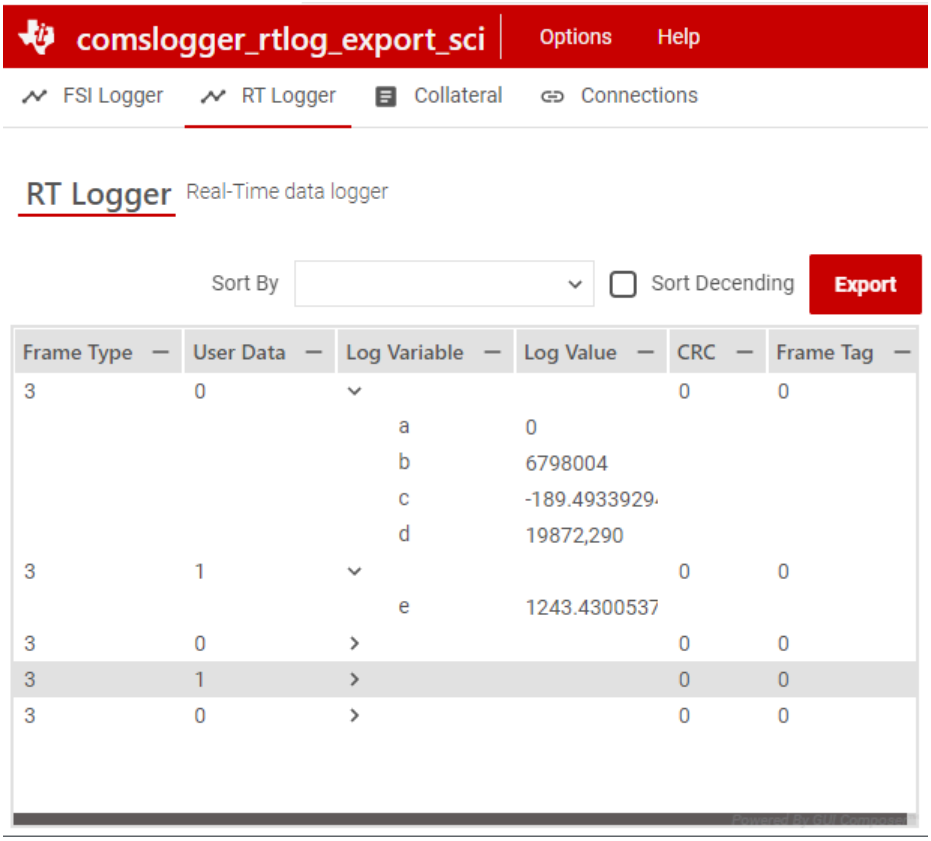


图 8-7. 在 PC GUI 中查看快速实时日志数据

9 传输示例概述

本文档前面所述的操作指南介绍了如何向现有 CCS 工程添加数据记录支持。此外，对于每个 MCU Control Center 特性的 C2000 示例代码，可参考 C2000Ware SDK 中的以下路径：*C2000Ware_VERSION#/driverlib/[DEVICE_GPN]/examples/[CORE_IF_MULTICORE]/transfer/*。下表给出了每个示例的说明，以及这些示例如何一起用于导出器件和桥接器件。

主器件示例	桥接器件示例	说明	硬件设置
transfer_customlog_export_usb	不适用	• *特性*：应用程序记录 • *主要通信协议*：USB • *主要封装格式*：JSON • 每秒在 GUI 中记录一次基于文本的简单消息（基本上就是一次使用 USB 的 printf）。	设置 #1
transfer_customlog_export_sci	(1)	• *特性*：应用程序记录 • *主要通信协议*：UART • *主要封装格式*：JSON • 每秒在 GUI 中记录一次基于文本的简单消息（基本上就是一次使用 UART 的 printf）。	设置 #2
transfer_customlog_export_sci_buffer	(1)	• *特性*：应用程序记录 • *主要通信协议*：UART • *主要封装格式*：JSON • 通过独立进程每秒在 GUI 中缓冲并记录基于文本的简单消息（基本上就是一次使用 UART 的 printf，适用于时序严格的系统）。	设置 #2
transfer_customlog_export_sci_logArrays	(1)	• *特性*：应用程序记录 • *主要通信协议*：UART • *主要封装格式*：JSON • 每秒在 GUI 中记录一次基于文本和数字的数据数组。	设置 #2
transfer_raw_fsi_tx	transfer_comslogger_export_sci	• *特性*：通信记录器 • *主要通信协议*：FSI • *主要封装格式*：无（仅含 FSI 帧） • 使用 FSI 记录原始数据 - 提取所有原始 FSI 数据并通过 SCI 传输以在 GUI 中显示	设置 #4
transfer_customlog_export_fsi	transfer_bridge_sci	• *特性*：应用程序记录器和传输桥 • *主要通信协议*：FSI • *主要封装格式*：JSON • 使用 FSI 记录数据（JSON 格式）- 接收 JSON 格式的 FSI 数据并通过 SCI 传输有效载荷以在 GUI 中显示	设置 #4
transfer_rtlog	transfer_comslogger_rtlog_export_sci	• *特性*：实时记录器和通信记录器 • *主要通信协议*：FSI • *主要封装格式*：自定义（根据帧结构） • 使用 FSI 以自定义字节打包格式记录实时数据 - 提取收到的 FSI 数据元素并通过 SCI 传输以在 GUI 中智能显示。	设置 #4

(1) 这个示例需要 UART 转 USB 桥接器件。所有 C2000 LaunchPAD 和 controlCARD 都已在电路板上集成该桥接器件，无需额外的硬件即可在 PC 上查看数据。使用定制电路板实现日志记录器件时，需要一个外部 UART 转 USB 桥接器件。

10 总结

MCU Control Center 提供了一种数据记录实施方案，它通过自定义 GUI 应用程序来采集和分析记录的数据。该工具提供的特性可添加到任何工程中，从而实现各种应用场景。MCU Control Center 工具通过对自动生成目标代码的支持、直观的图形用户界面 (GUI) 以及对 C2000 器件上现有通信外设的高效利用，实现了从 C2000 器件采集数据的无缝化操作。

11 参考资料

- 德州仪器 (TI)，MCU Signal Sight 工具开发指南，应用手册
- 德州仪器 (TI)，C2000 实时微控制器外设，用户指南
- 德州仪器 (TI)，C2000 Sysconfig，应用手册
- 德州仪器 (TI)，DLT 开发指南（带工具），应用手册

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