Application Note

DLT 开发人员指南（带工具）

Ryan Ma

摘要

数据记录器和跟踪 (DLT) 外设是具有应用程序代码跟踪支持的非侵入式数据记录的关键元素。此外设可用于工业和汽车应用。C29x 实时微控制器提供了一种非侵入式方法来记录关键 CPU 运行时内容并提供跟踪功能，而不会增加 CPU 开销。本应用手册重点介绍简介部分描述的应用用例，演示了 DLT 的每项特性以及如何使用 SysConfig 系统配置工具设置 DLT 并进行编程。此外，还提供了有关如何设置 DLT 可视化工具的附加资料，以便在不使用外部硬件的情况下解读数据记录和跟踪内容。SysConfig 工具可集成于 Code Composer Studio 中，或作为独立程序使用，支持用户使用图形用户界面 (GUI) 生成 C 头文件和代码文件。本应用手册是根据 F29H859TU8ZEXQL 器件编写的。但是，本应用手册中的内容适用于具有 DLT 外设的所有器件。

内容

1 简介....2
2 C28x、C29x 与 ARM 日志记录....3
3 SysConfig....4
3.1 开始或停止记录....4
3.2 捕获模式....6
3.3 用于传输日志的触发器....7
4 解读 DLT 日志....8
5 编译器内在函数....8
6 DLT 工具....9
6.1 可视化....9
6.2 工具操作说明....9
6.3 将日志添加到应用程序....13
6.4 导出 DLT 日志....14
6.5 CCS Theia....16
7 总结....19
8 参考资料....19

商标

1 简介

为什么数据记录和跟踪在实时控制系统中很重要？

- 在开发或测试期间和之后调试应用程序代码
- 分析应用程序代码
- 创建控制系统的捕获日志以进行深度分析
- 跟踪应用程序代码流

许多应用程序需要数据日志或跟踪功能在器件上具有不同的用途。DLT 提供了一种记录关键运行时内容，然后通过 JTAG、UART 或 FSI 导出的方法。如果没有可用的 JTAG 连接，则在有 UART 或 FSI 实现的情况下仍可以使用 DLT 来导出数据。控制正在记录的内容的代码行可以保留在应用程序代码中，而不会影响 CPU 性能。

DLT 通过 C29x 用户指南中提供的专用指令提供数据记录和代码流执行功能。当使用 DLT 记录数据变量或在应用程序中添加代码流标记时，每个日志都附加了额外信息。额外信息取决于 DLT 捕获日志的方式。添加到每个日志的额外信息有两种模式，即时间戳或程序计数器信息。在时间戳模式下，DLT 提供所记录变量或代码标记的时间信息。在程序计数器模式下，DLT 提供信息以告知这些日志发生在何处。

有用于记录信息的专用指令。指令具有以下名称：DLTAG 和 DLREG。DLTAG 用作代码流标记。DLREG 是允许用户记录数据变量的指令。通过利用通过 C29x 处理器并行运行的多个指令，这些指令可以并行运行，并在记录数据或向应用代码添加代码流标记时提供非侵入式行为。协同处理器接口 (CPI) 从 CPU 查找这些专用指令，并向 DLT 提供数据包信息，后者记录到每个 CPU 的专用内存地址区域。CPU 或 DMA 可以从 DLT 内部内存中读取日志，并根据需要移动记录的信息。

以下是一个概览视图，从转至应用程序代码开始，以可视化 PC 上 DLT 记录的数据。

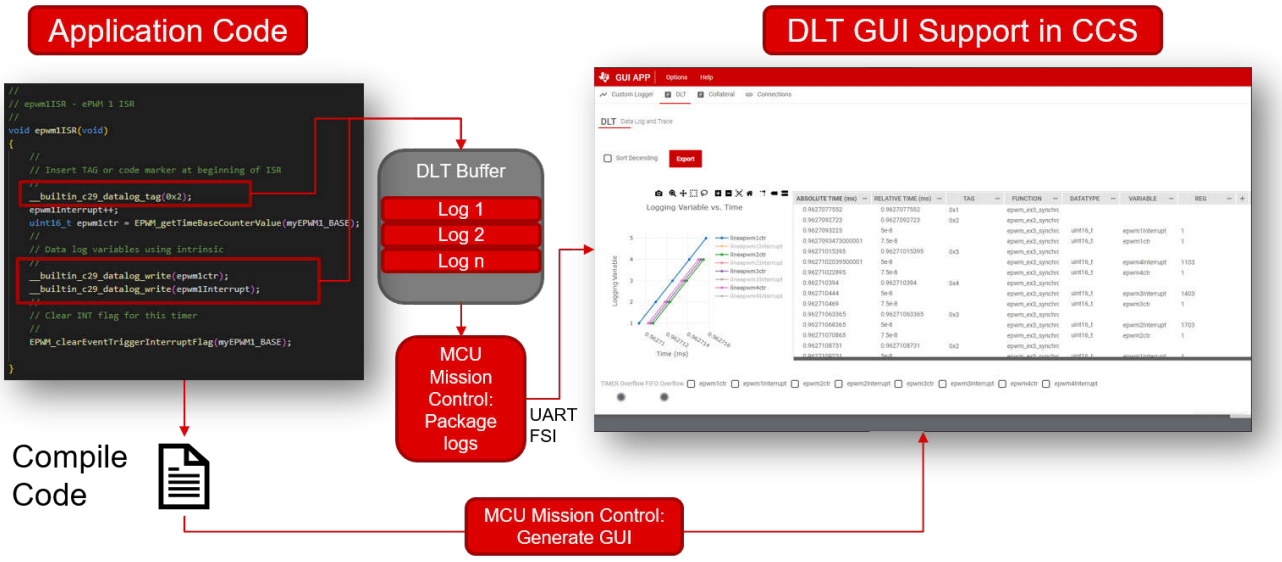

图 1-1. DLT 示例代码片段

本应用手册介绍了配置 DLT、数据日志以及使用内置编译器内在函数添加代码标记所需的步骤。DLT 利用编译器和 SysConfig 提供了快速开始使用此外设的方法。

本应用报告中讨论的用例使用 DLT 在 ISR 中记录温度传感器样本和 ADC 结果。

2 C28x、C29x 与 ARM 日志记录

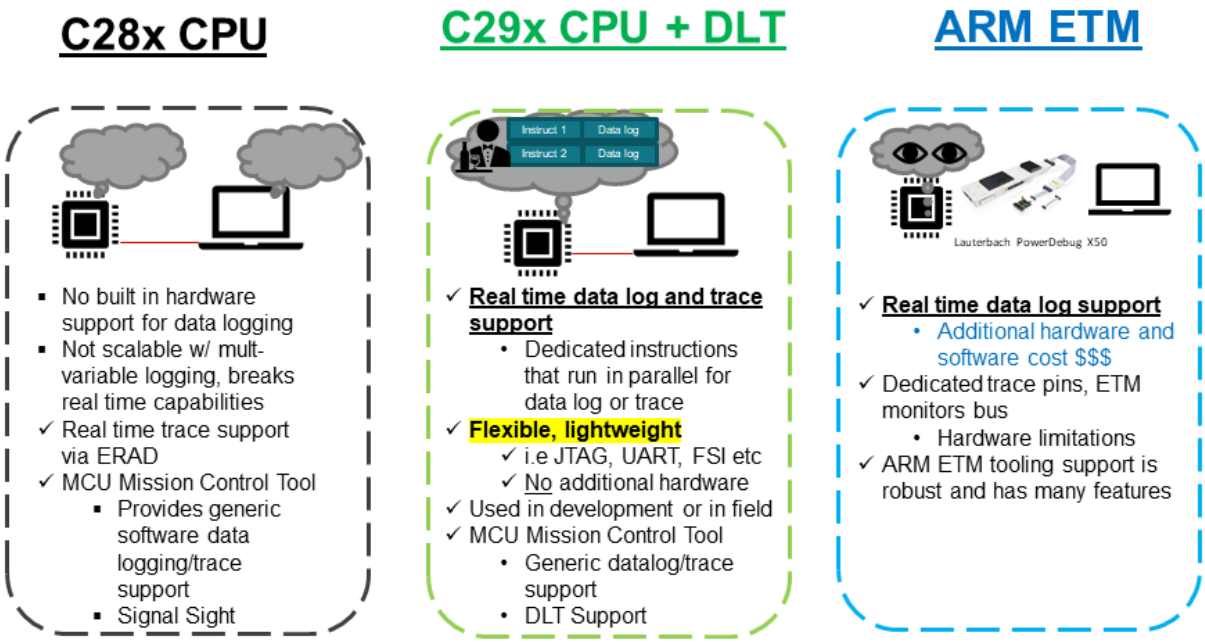

图 2-1. C28x、C29x 与 ARM 日志记录

3 SysConfig

如摘要中所述，本应用手册利用 SysConfig 来配置 DLT 外设以及其他必要元件，例如可视化 DLT 包。借助 SysConfig GUI，用户可以完成 DLT 为准备好开始在应用程序代码进行记录所需的配置。

3.1 开始或停止记录

配置 DLT 的第一步是设置记录信息的开始和停止事件。有两种主要方法可以通过代码标记 (DLTAG) 或使用 ERAD 事件来开始或停止记录信息。第三种方法侧重于安全性，可防止基于所启用链路过滤器记录的信息发生。标签或 ERAD 过滤器控制何时开始或停止数据记录。链路过滤器用于根据当前链路的访问权限过滤出代码到数据日志的部分。

基于标签的过滤具有一些选项，例如 *开始标签基准*、*开始标签掩码* 和结束标签配置。这些选项可用于开始或停止 DLT 进行记录。掩码与到达应用程序代码中的代码标记进行与运算。如果当前标签值和标签过滤器起始掩码的与输出等于开始标签基准，并且启用了基于标签的过滤，则 DLT 将开始记录。使用结束标签停止 DLT 记录也是如此。开始/结束标签基准可以是任何 16 位值。

下面的流程图说明了当用户同时具有基于标签和 ERAD 的条件来启动或停止记录时会发生的情况。

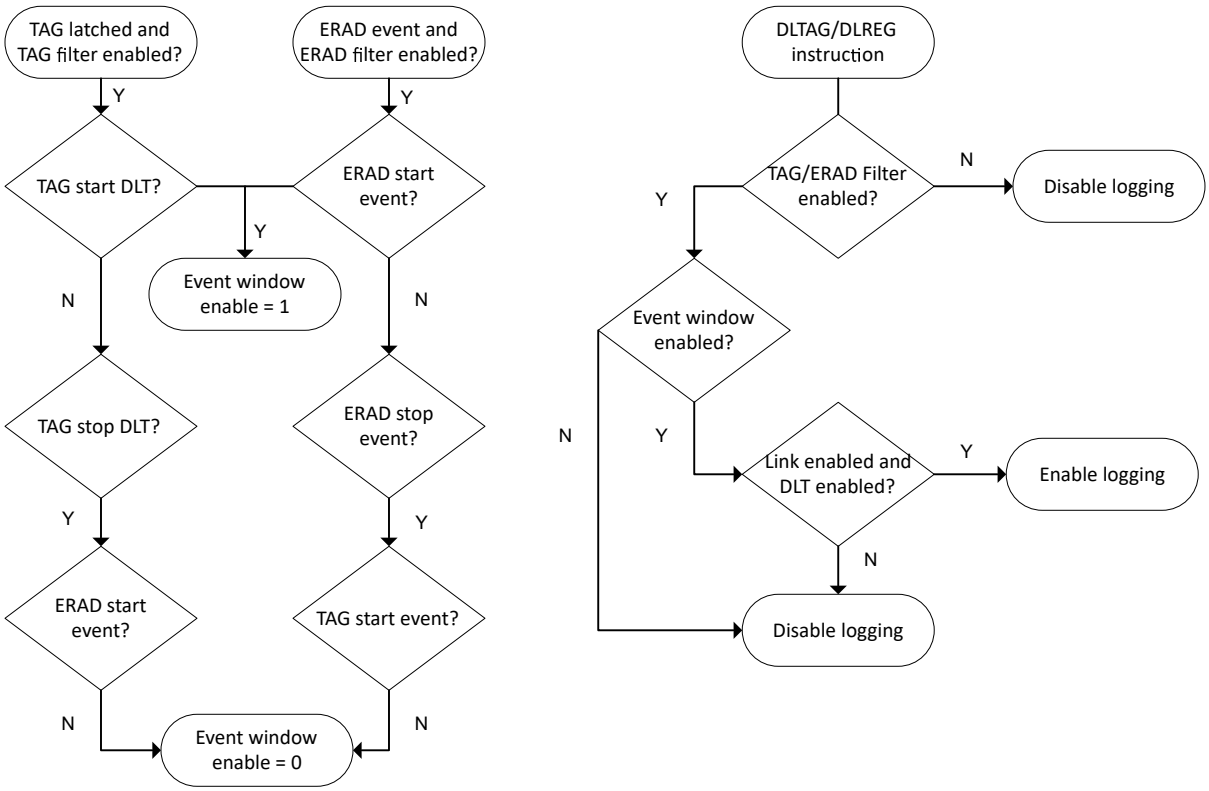

图 3-1. 开始和停止事件决策图

配置开始或停止结束标签可以使用任何 16 位值。SysConfig 提供了有用的 GUI 来配置这些功能。

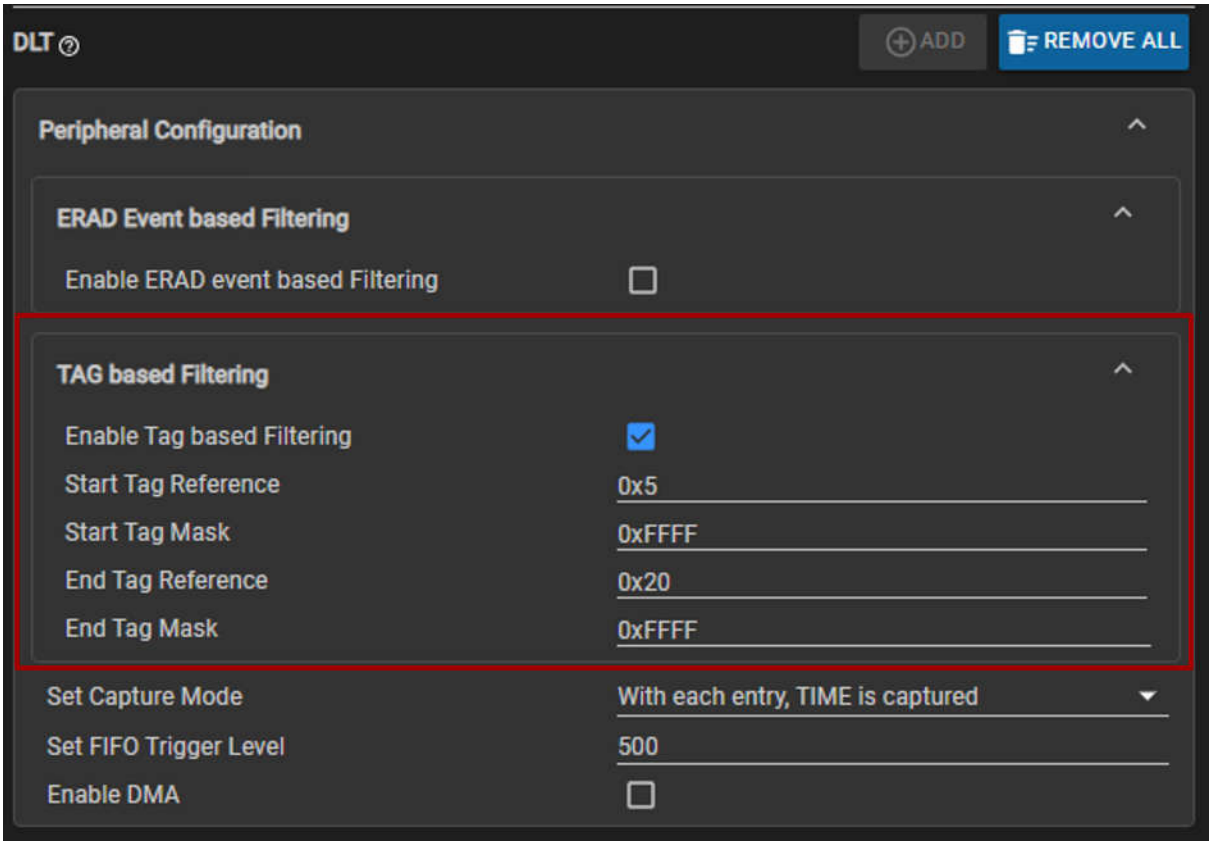

图 3-2. DLT 标签过滤器 SysConfig

必须选择*启用基于标签的过滤*选项。概述了用于配置开始标签基准值和结束标签基准值的配置。*开始标签掩码*和*结束标签掩码*与相应的开始或结束基准值进行与运算，以控制日志的发生时间。在上述配置中，0x5 是所使用的开始标签基准值，0x20 是结束标签基准值。这些值是任意值，可以使用任何值进行配置。SysConfig 的 board.c 文件中生成的代码如下所示。

```
//*****************************************************
//
// DLT Configurations
//
//*****************************************************
void DLT_init(){
    DLT_enableModule();
    DLT_enableLinkPermission(SSU_LINK2);
    DLT_enableTagFilter();
    DLT_ConfigTagFilterParams configParams;
    configParams.startReference = 5;
    configParams.startMask = 65535;
    configParams.endReference = 32;
    configParams.endMask = 65535;
    DLT_configTagFilter(&configParams);
    DLT_setCaptureMode(DLT_CAPTURE_TIME);
    DLT_setFIFOTriggerLevel(500);
    DLT_enableInterrupt(DLT_INT_FIFO_TRIG);
    DLT_resetTimer();
}
```

图 3-3. DLT 初始化片段

3.2 捕获模式

有两种独立模式可通过 DLT 启用。每个捕获模式都有一种特定的格式，用于说明每个日志的附加信息在 DLT 内部存储器中的显示方式。该内部存储器充当 FIFO，用于存储代码标记和变量的日志。请参阅“解读 DLT 日志”部分，以详细了解这些日志在两种捕获模式之间的关系。

第一种模式是时间捕获模式，在该模式下，每个日志都包含有关何时到达日志的信息。在此模式下，计时器的来源是 IPC 计数器和 DLT 的内部计数器。代码标记 (DLTAG) 使用 IPC 作为源，数据记录变量 (DLREG) 使用 DLT 的内部计数器作为源。对于代码标记 (DLTAG)，计时器值称为 TIMER1，且将源自 IPC 计数器。此时间是 IPC 启动的绝对时间。对于数据记录变量 (DLREG)，计时器值称为 TIMER2，且将源自 DLT 的内部计数器。此计时器值始终是已达到的前一个代码标记 (DLTAG) 之间的时间基准。也可将此时间视为上一个代码标记之间的相对时间。

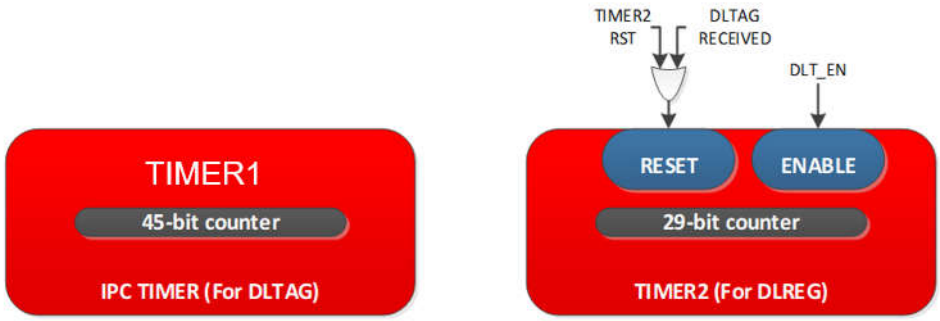

图 3-4. TIMER1(DLTAG) 与 TIMER2(DLREG)

第二种模式是程序计数器模式，在该模式中，每个日志都包含有关到达日志位置的信息，而不是包含计时器值。SysConfig 提供了一种在初始化时以任一模式配置 DLT 的方法。

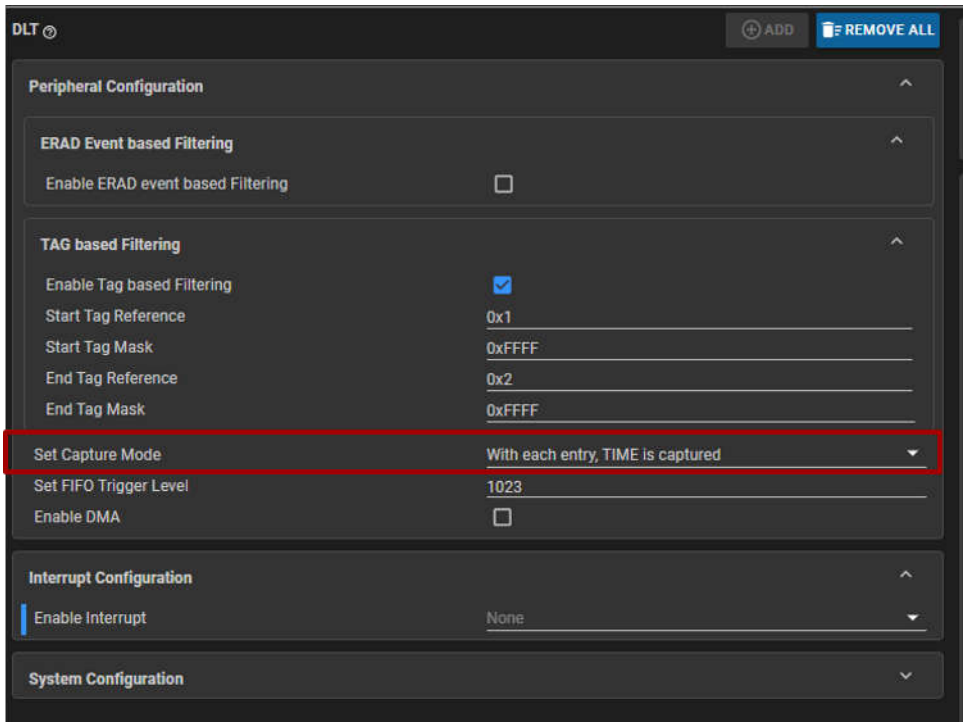

图 3-5. DLT SysConfig - 捕获模式

3.3 用于传输日志的触发器

有几种方法可以从 DLT 缓冲区读取数据并使用 CPU 或 DMA 接口将日志写入器件上的另一个位置。一旦数据移动，就可以使用器件上的任何通信外设将日志传输出去。DLT 可根据缓冲区中的日志数量触发 CPU 中断或 DMA 传输请求。请参阅 FIFO_CONTROL.WR_CTR_TRIG_LEVEL，了解可以设置的最大触发电平。当 FIFO 达到 500 个记录的元素时，以下配置将生成 DLT 中断。

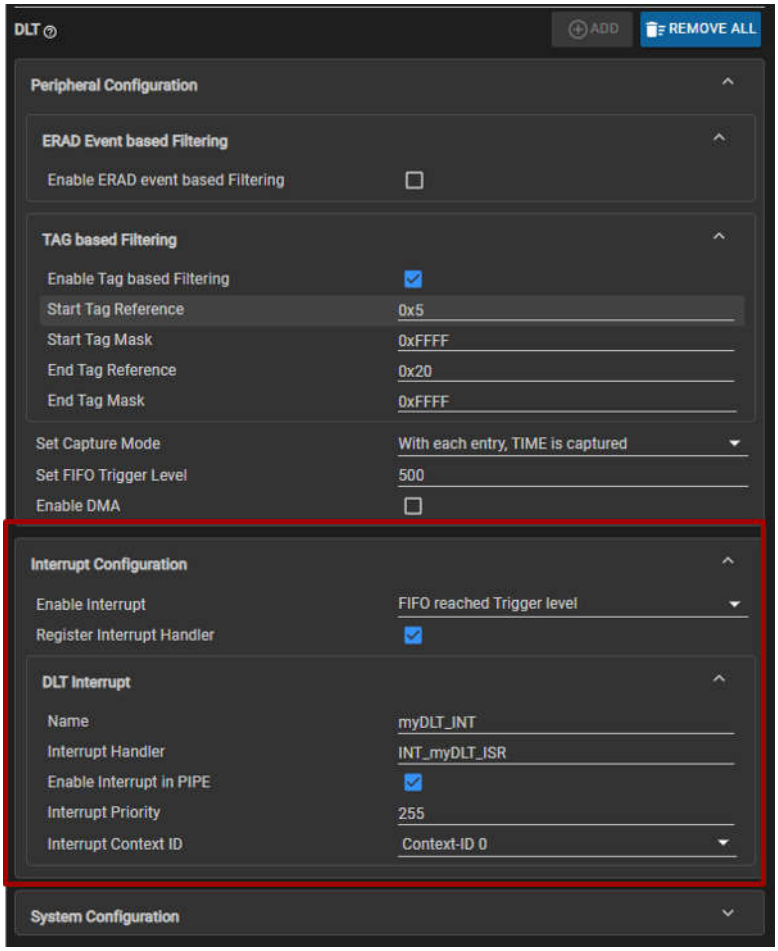

图 3-6. DLT SysConfig - 触发器 FIFO 电平和 DMA

4 解读 DLT 日志

前面几节介绍了如何启动和结束日志的设置、存在哪些不同的模式，以及如何根据 DLT 的触发级别将日志传输到其他位置。用户如何解读来自内部存储器的日志？

有两个术语：代码标记和要记录数据的变量。每一个都有一种方法可以从 DLT 的内部内存中解读日志。代码标记或 DLTAG 的解读必须与所记录数据的变量或 DLREG 略有不同。

存储日志的内部存储器位于 DLT_FIFO_REGS 的基址。内部存储器的工作方式与 FIFO 类似。当从 FIFO_BUF_H 读取时，下一个日志被推入。因此，从 FIFO 进行读取的顺序很重要。对该存储器映射寄存器进行读取的方法是首先读取 FIFO_BUF_L 内容，然后读取 FIFO_BUF_H 内容。

根据捕获模式的不同，对日志的解读也不同。从 FIFO 读取时，首先要查找的一个项目是低 32 位的 LSB。此寄存器提供有关这是代码标记 (DLTAG) 还是所记录变量 (DLREG) 的信息。其余信息可以通过技术参考手册中描述的表格进行解码。对于此应用手册，当模式设置为捕获时间值时，DLT 工具用于解读 DLT 信息。

5 编译器内在函数

C29x 用户指南中的专用指令可以通过使用 ti-cgt-c29 编译器中提供的内置编译器内在函数进行抽象。在使用内置编译器内在函数的前提下，使用专用 DLT 指令。

对于代码标记，请使用：__builtin_c29_datalog_tag(TAG)；

- 此内在函数需要任何 16 位标签值

对于数据记录变量，请使用：__builtin_c29_datalog_write(VAR)；

- 此内在函数需要任何 32 位变量

备注

每个新函数范围必须至少有一个代码标记来表示新函数范围，然后是要记录数据的变量。只接受具有代码标记。TI 不建议在函数范围内不设置后跟要记录数据的变量的代码标记。

6 DLT 工具

读取和解读 DLT 日志具有挑战性，因为在重建最终打包的记录信息时需要跟踪许多位，并确保它们正确无误。

创建了 DLT 工具以战胜此障碍。

6.1 可视化

DLT 工具有两种通信外设设计，用于将 DLT 日志从设备中导出。本应用手册重点介绍了一种基于 UART 的设计，用于将 DLT FIFO 从器件中导出，并通过 COM 端口或 GUI Composer 等 GUI 界面显示数据。另一个通信外设是 FSI，可与该工具配合使用。这需要 FSI 到 UART-USB 桥接器件。本应用手册使用 F29x MCU SDK 中以下文件夹下的一个示例：C:\ti\c2000\mcu_sdk_f29h85x_x_xx_xx_xx_xxx\examples\driverlib\single_core\transfer\transfer_adc_tempsensor_dlt。以下是有关如何将 DLT 工具添加到任何项目的详细步骤。

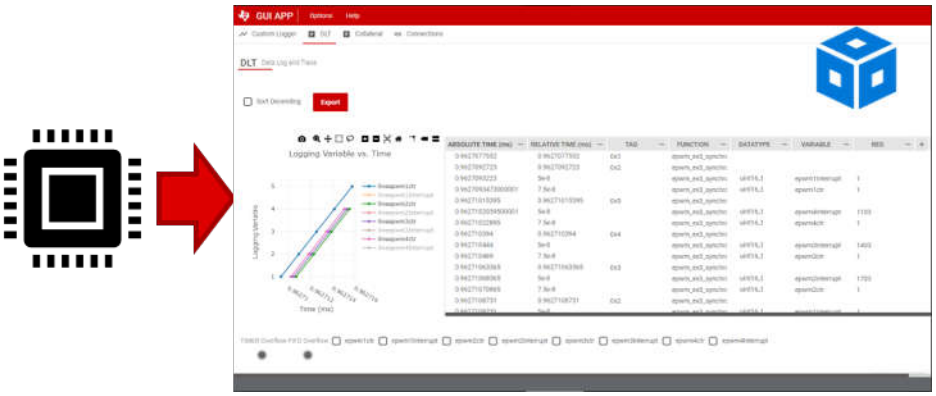

图 6-1. DLT

6.2 工具操作说明

DLT 工具可在 SysConfig 的 MCU 任务控制和传输模块下找到。

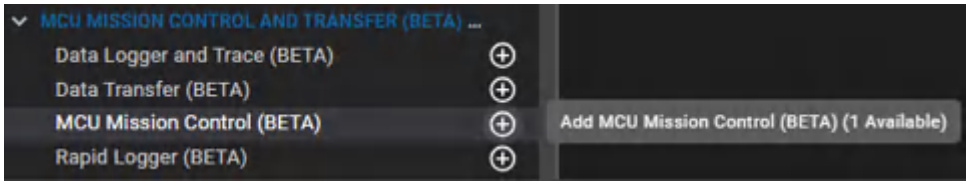

图 6-2. MCU 任务控制

添加 DLT 工具支持的第一步是添加 MCU 任务控制模块。添加此模块后，会生成一些文件，用于创建 CCS 中所示的最终 GUI。所有这些文件都将创建 GUI Composer 应用程序的前端和处理传入到 GUI 的数据所需的后端。

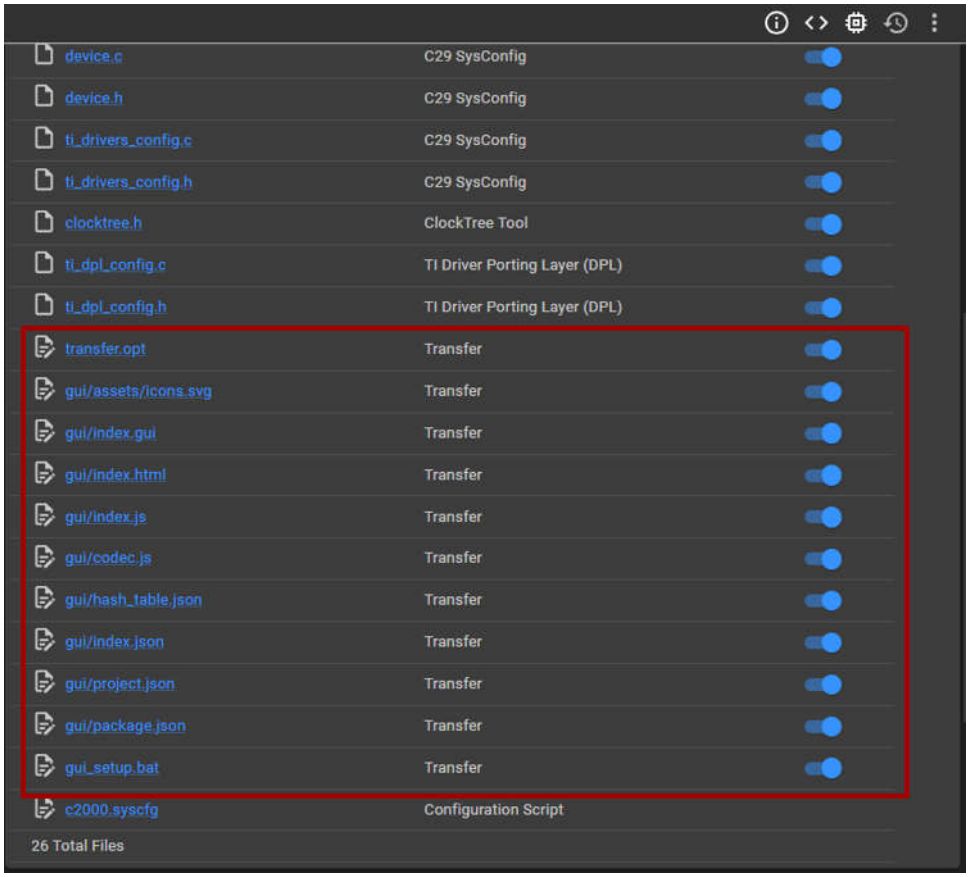

图 6-3. MCU 任务控制 - 生成的文件

该子模块生成 DLT 工具可视化所需的所有必要 GUI 元素。要添加的下一项是启用自定义导出记录器。

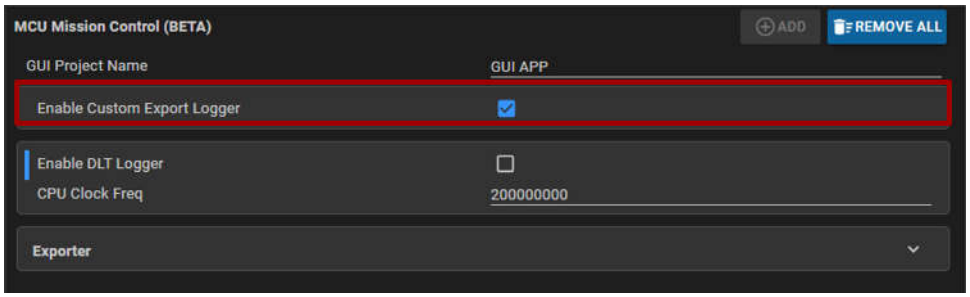

图 6-4. MCU 任务控制 - 启用自定义导出记录器

此模块帮助定义用于将 DLT 数据包发送到 GUI 的软件包模式，以及要使用的通信外设。启用后，下一个要启用的选项是 *DLT 记录器*。

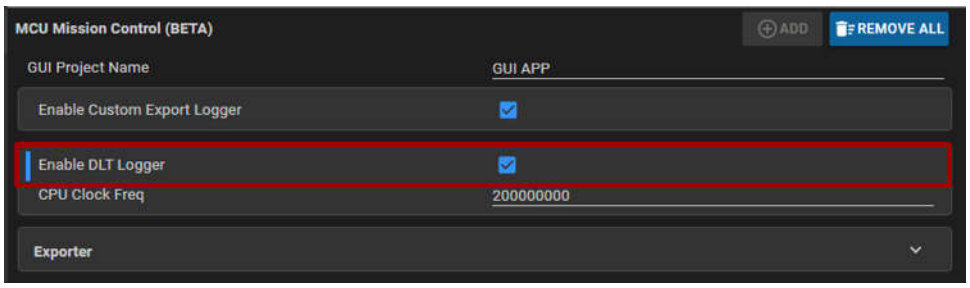

图 6-5. MCU 任务控制 - 启用 DLT 记录器

启用 *启用自定义导出记录器* 后，可以选择一些可配置选项。可以在 *导出器* 选项卡下查看这些选项。该选项卡如图 6-6 所示。

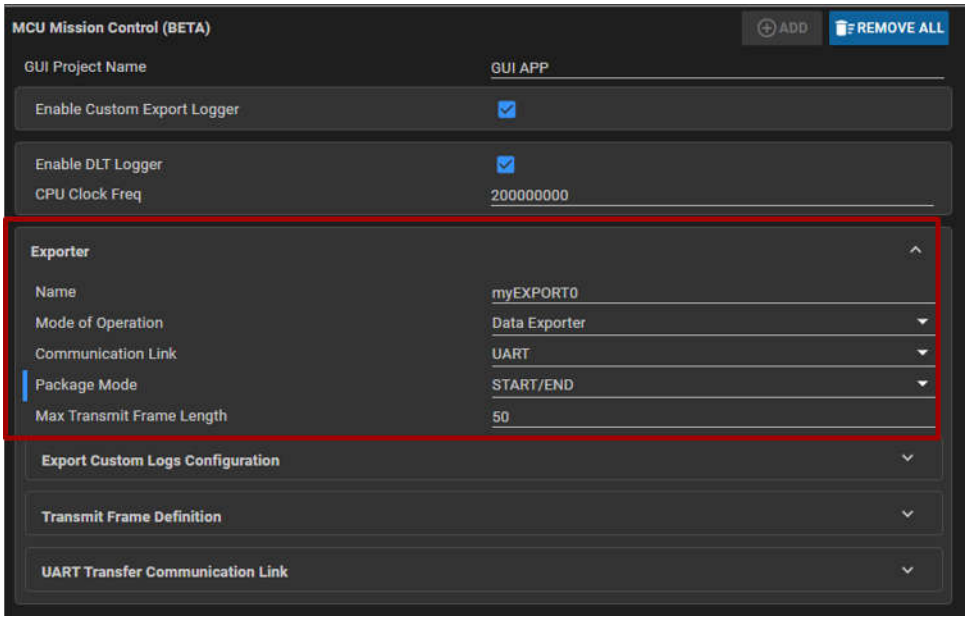

图 6-6. MCU 任务控制 - 导出器模块

将 *软件包模式* 更改为 *开始/结束*。SysConfig 会生成一个软件层，通过可在以下生成的文件中找到的简单 API 发送数据。

图 6-7. MCU 任务控制 - 导出器模块代码生成

生成的文件用于在将数据导出器件之前以指定的格式对数据进行打包。在“导出”和 export_package.c 下，有多种函数可用于将数据从器件中导出。在本应用手册中，重点介绍了由 SysConfig 生成的 DLT 支持 API。以下是用于导出 DLT 日志和通过 GUI 可视化的主文件。

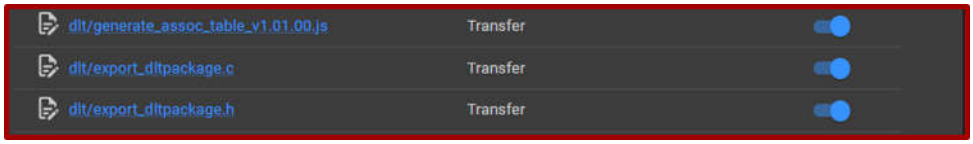

图 6-8. MCU 任务控制 - DLT 记录器代码生成

在 dlt/export_dltpackage.h 中，有一个高级 API 可用于将大小为 2 的 uint32_t 数组导出到 GUI。

在 *导出器* 模块下，打开“传输帧定义”并添加一个名为 *DLTlog* 的新密钥类型，其中的值类型为 32 位无符号整数。哈希表 ID 必须为 *49*，GUI Composer 才能知晓 GUI 端接收到的是什么数据包。

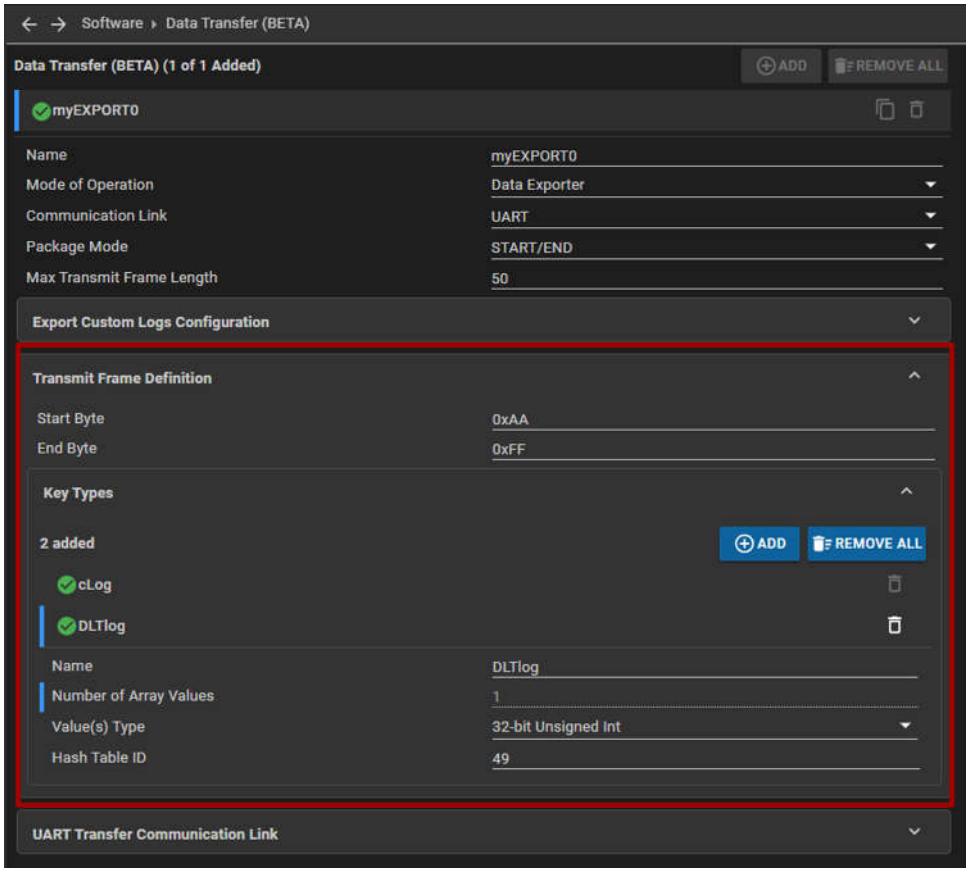

图 6-9. MCU 任务控制 - 传输帧定义

要使正确的引脚多路复用器在 controlSOM 上使用 UART，请为 UART_TX 选择 GPIO42，为 UART_RX 选择 GPIO43。

添加 MCU 任务控制并启用客户导出记录器后，在 *启用 DLT 记录器* 复选框下，必须添加设备正在使用的 CPU 时钟频率。时钟频率用于确定与每个日志相关联的计时器值。

启用 *启用 DLT 记录器* 复选框时，将生成 *dlt/generate_assoc_table_v1.01.00.js* javascript 文件，该文件用于通过编译器工具的抽象语法树 (AST) 转储解析源代码信息。此 javascript 文件用于查找正在使用的 DLT 内在函数，并能够在 GUI 上显示这些内在函数。

必须包含头文件才能添加到应用项目中，从而使用生成的 SysConfig 支持。

```
//
// Included Files
//
#include "board.h"
#include "export/export_log.h"
#include "dlt/export_dltpackage.h"
#include "bitfield_structs.h"
```

图 6-10. MCU 任务控制 - 必须包含的内容

6.3 将日志添加到应用程序

示例中记录的数据有非常具体的变量。要记录的几个变量是 sensorSample 和 sensorTemp。可传递到内在函数中的变量也可以是数组的元素。例如，如果正在使用大小为 30 的数组，内在函数可以与 array[29] 一起传递给数据

日志第 30 个元素。这是用于添加到参考设计以记录变量的代码片段。每个新函数范围中至少放置一个标签，以确保遵守这些指南。在参考设计的 psfbpcmc.c 文件下，添加以下内在函数。

```
//
// adcA1ISR - ADC A Interrupt 1 ISR
//
void adcA1ISR(void)
{
    //
    // Create Start Tag/Marker for DLT
    //
    __builtin_c29_datalog_tag(0x05);

    //
    // Read the raw result
    //
    sensorSample = ADC_readResult(ADCARESULT_BASE, ADC_SOC_NUMBER0);

    //
    // Datalog sample
    //
    __builtin_c29_datalog_write(sensorSample);

    //
    // Convert the result to a temperature in degrees C
    //
    sensorTemp = ADC_getTemperatureC(sensorSample, ADC_REFERENCE_INTERNAL, 3.3f);

    //
    // Datalog temperature
    //
    __builtin_c29_datalog_write(sensorTemp);

```

图 6-11. DLT 代码标记和数据日志内在函数 - PSFB_updateSensedValues()

如 图 6-11 所示，日志记录开始到 DLT 日志缓冲区，因为传递到代码标记内在函数的值是 0x05，并且此值与 DLT 初始化中配置的起始参考值匹配。

6.4 导出 DLT 日志

插入所有编译器内在函数后，可以通过从 FIFO 读取并调用 DLT 导出包软件 API 轻松地从器件导出。在后台循环中，可以使用以下代码片段发送 DLT 封装。

以下函数清空了 DLT 日志缓冲区中的内容，并写入 API 以将数据从设备中导出。DLT 中断内容用于在 FIFO 达到 500 个元素时设置 empty_dlt_fifo 标志。

```
//
// Loop indefinitely
//
while(1)
{
    uint32_t dlt_packet[2] = {0,0};

    while (empty_dlt_fifo && DLT_getFIFOWordStatus())
    {
        //
        // Export data to GUI via UART
        //
        dlt_packet[1] = CPU1DLTFifoRegs.FIFO_BUF_L;
        dlt_packet[0] = CPU1DLTFifoRegs.FIFO_BUF_H;
        EXPORTDLTLOG_logUint32Array(dlt_packet, 2);

        if(DLT_getFIFOWordStatus() == 0)
        {
            //
            // Start DLT log when FIFO is empty
            //
            empty_dlt_fifo = 0;
        }
    }

}
```

图 6-12. DLT 代码标记和数据日志内在函数 - psfb_main.c

```
//
// DLT ISR export data when data logging level is reached
//
void INT_myDLT_ISR(void)
{
    //
    // Stop data logging after reaching
    // FIFO trigger count
    //
    __builtin_c29_datalog_tag(0x20);
    empty_dlt_fifo = 1;
    //
    // Clear DLT Interrupt events to receive more
    //
    DLT_clearEvent(DLT_INT_FIFO_TRIG | DLT_INT_INT);
}
```

图 6-13. DLT 中断内容

6.5 CCS Theia

所有这一切的最后一步是添加编译后处理步骤以生成我们的 GUI。要启用 GUI 支持，请转到“项目属性”->“通用”->“变量”，然后添加两个分别名为 DLT_SUPPORT 和 GUI_SUPPORT 的变量。将值设置为 1。

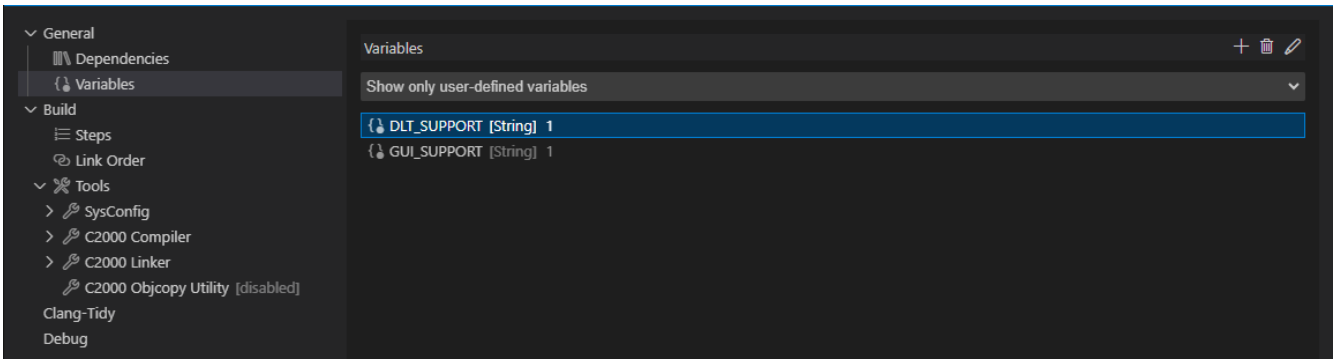

图 6-14. 用户定义的变量

需要执行此步骤才能生成 GUI，请复制并粘贴以下编译后处理步骤。

```
if ${DLT_SUPPORT} == 1 ${NODE_TOOL} ${BuildDirectory}\syscfg\dlt\generate_assoc_table_v1.01.00.js "${CG_TOOL_ROOT}/bin/c29clang.exe -Xclang -ast-dump=json" "-c -I"${COM_TI_MCU_SDK_F29H85X_INSTALL_DIR}/source" -I"${COM_TI_MCU_SDK_F29H85X_INSTALL_DIR}/source/driverlib" -I"${COM_TI_MCU_SDK_F29H85X_INSTALL_DIR}/source/bitfields" -I"${COM_TI_MCU_SDK_F29H85X_INSTALL_DIR}/examples/device_support/include" -I"${COM_TI_MCU_SDK_F29H85X_INSTALL_DIR}/source/rtlibs/dcl" -I"${COM_TI_MCU_SDK_F29H85X_INSTALL_DIR}/source/rtlibs/dsp/fpu/fpu32/fft" -I"${COM_TI_MCU_SDK_F29H85X_INSTALL_DIR}/source/rtlibs/iqmath" -I"${COM_TI_MCU_SDK_F29H85X_INSTALL_DIR}/source/kernel/freertos/Source/include" -I"${COM_TI_MCU_SDK_F29H85X_INSTALL_DIR}/source/kernel/freertos/Source/portable/CCS/C2000_C29x" -I"${PROJECT_ROOT}/" -I"${CG_TOOL_INCLUDE_PATH}" -I"${PROJECT_ROOT}/CPU1_RAM/syscfg" -I"${CG_TOOL_INCLUDE_PATH}" -DDEBUG -g"

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

备注

这适用于 CPU1_RAM 配置，如果将用于闪存，请在第一个编译后命令中将 CPU1_RAM 更新为 CPU1_FLASH 文件夹。

添加这些编译后处理步骤后，继续操作并编译项目。当项目构建完成，可以查看 GUI 后，转到“视图”->“重新加载窗口”。窗口完成重新加载后，转到“视图”->“插件”->“gui_app”。显示的名称取决于 MCU 任务控制模块下 *GUI 项目名称* 中提供的名称。

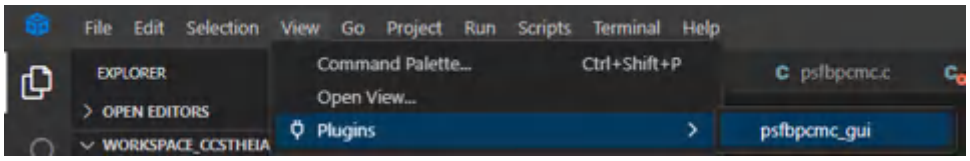

图 6-15. CCS Theia 插件

称为 DLT 的 GUI 选项卡，所有 DLT 内容都位于该选项卡中。此表和图表显示了应用于源代码的所有标签标记和变量。如果在源代码中添加了更多内在函数，则需要重构项目，并且需要重新打开 GUI。

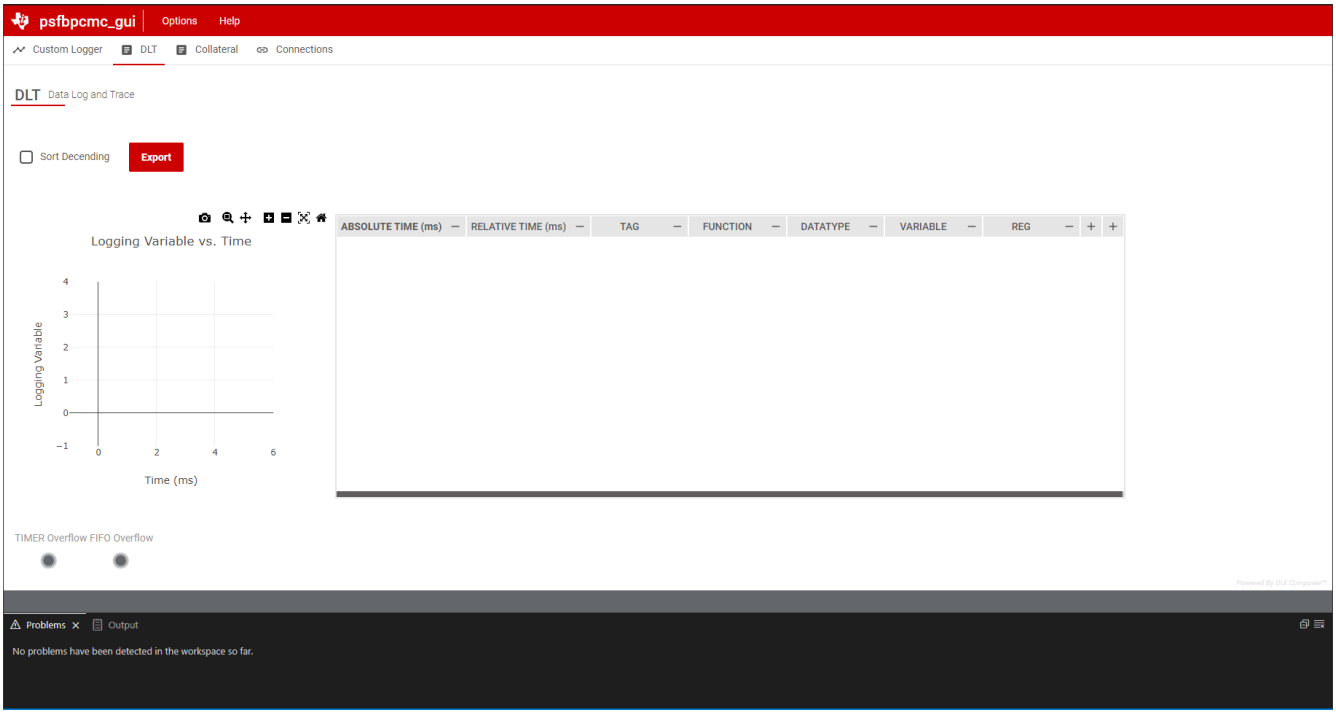

图 6-16. CCS Theia 中的基本 GUI

转到 *选项→串口设置...* 来选择正确的 COM 端口。

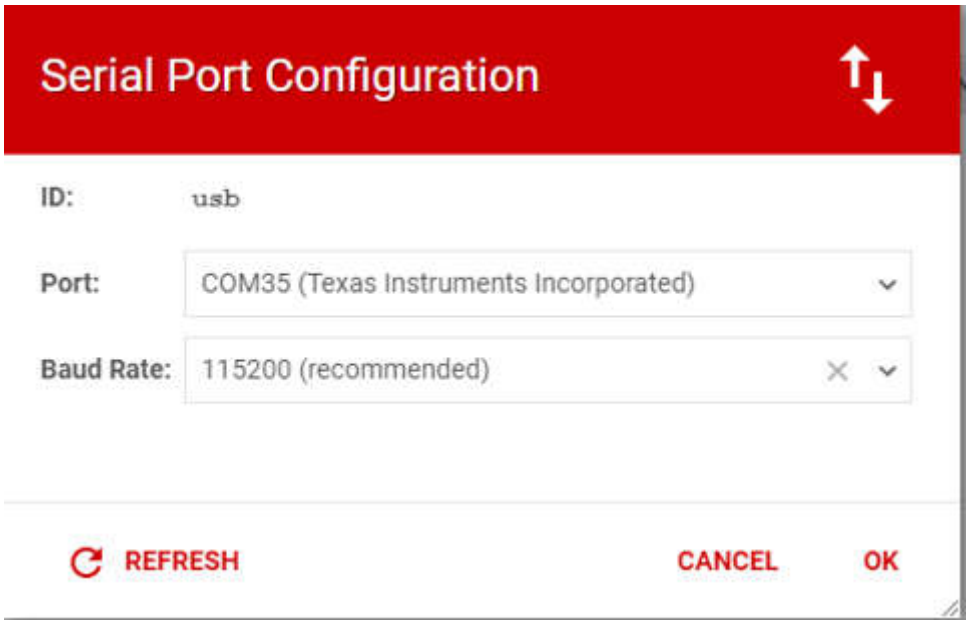

图 6-17. 串行端口配置

必须选择正确的 COM 端口才能使 GUI 正常工作。

连接至器件并将程序加载到器件中，图形会填充数据。

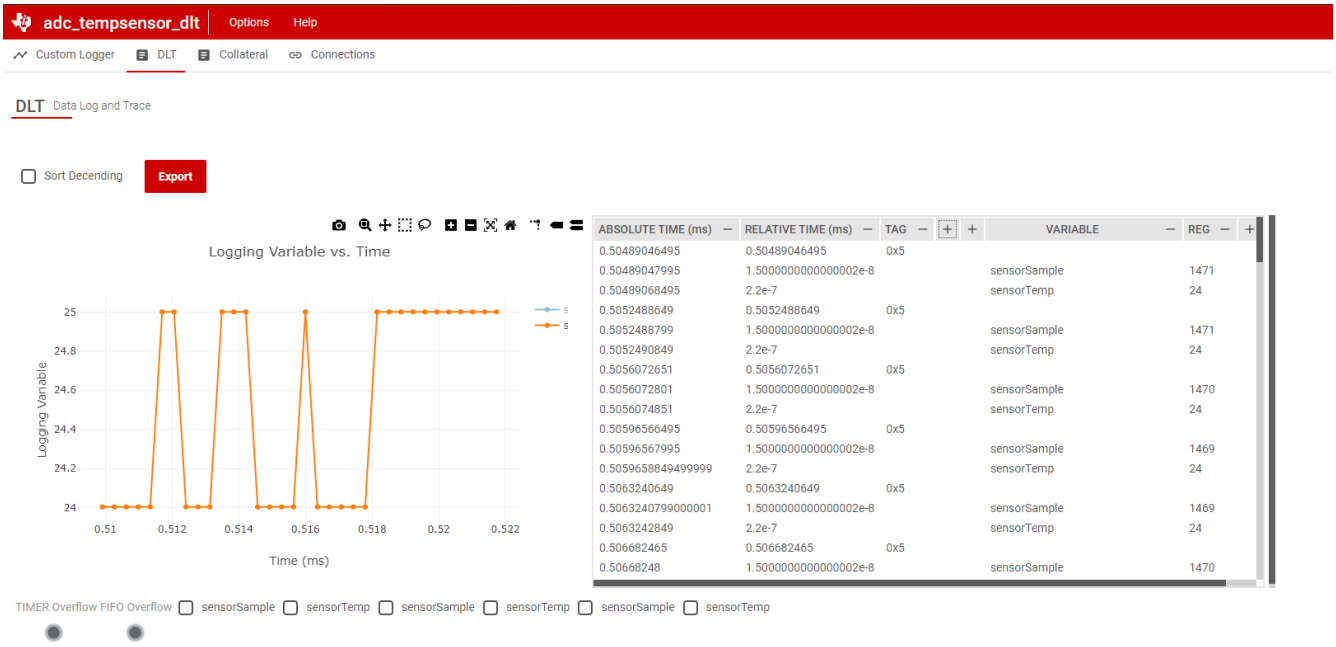

图 6-18. 最终输出 GUI Composer

显示的列是转换后的 DLT 包日志。绝对时间是转换为毫秒的 IPC 计时器值。相对时间始终相对于上一个代码标记或 DLTAG。*标签* 列是传递到 __builtin_c29_datalog_tag 编译器内在函数中的值。这是一种跟踪应用程序代码的方法。函数列告知作为数据日志的变量来自哪个函数。*数据类型* 列是指源代码中变量的数据类型。*变量* 列是指所记录数据的变量。*寄存器* 列是在应用程序代码中到达日志时变量的值。其他列会被最小化，因为这些列与溢出以及日志是代码标记还是变量日志有关。

表 6-1. DLT GUI

绝对时间	相对时间	TAG	功能	数据类型	变量	寄存器	计时器 1 OVF	计时器 2 OVF	代码标记还是变量日志？
IPC 计时器的绝对时间	代码标记或 DLTAG 之间的相对时间	在 _builtin_c29_datalog_tag 中使用的标签值	源代码中使用 DLT 的源文件信息和函数名称	变量的数据类型（即 float、int、uint 等）	在传入 __builtin_c29_datalog_write 的应用程序代码中使用的变量名	所记录的变量的寄存器值	如果 OVF 为 1，则发生了溢出	如果 OVF 为 1，则发生了溢出	0 - 变量日志 1 - 代码标记

7 总结

本应用手册重点介绍了 DLT 的所有主要功能，以及如何在应用中利用 DLT。手册中介绍了 DLT 和可视化 DLT 日志内容所需的所有配置。有关 DLT 模块上寄存器的更多具体指南，请参阅器件特定技术参考手册。

8 参考资料

- 德州仪器 (TI)，F29H85x 和 F29P58x 实时微控制器 技术参考手册
- 德州仪器 (TI)，C29 DLT 外设概览视频 网页
- 德州仪器 (TI)，C29x Academy 网页
- 德州仪器 (TI)，C29x SDK 工具页

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