Application Note

マイコン制御センターツールの開発者ガイド

Ryan Ma, Delaney Woodward and Nima Eskandari

概要

マイコン制御センターツールは C2000™ マイコン用のデータロギングおよび可視化ツールです。このツールは コードコンプレッサースタジオ™ (CCS) 環境と SysConfig ツールに統合され、使いやすさが向上しています。このアプリケーションノートでは、データロギングを有効にするためにツールセットで使用できる一般的なハードウェア設定を特定し、さまざまな機能と使用事例のシナリオについて説明し、既存のプロジェクトへの追加サポートについて調べ、使用可能な SDK 例を紹介します。このドキュメントで説明されている機能は次のとおりです:

- アプリケーションロギング - PC への汎用データロギング
- 転送ブリッジ - 高速シリアルインターフェイス (FSI) 周辺装置ログを使用したデータログ、200MBPS 対応
- 通信ログ - FSI または通信周辺装置アプリケーションのデバッグを使用した拡張データロギング
- 高速時間ロギング - リアルタイムアプリケーション用に対する侵襲性が最小のデータロギング

目次

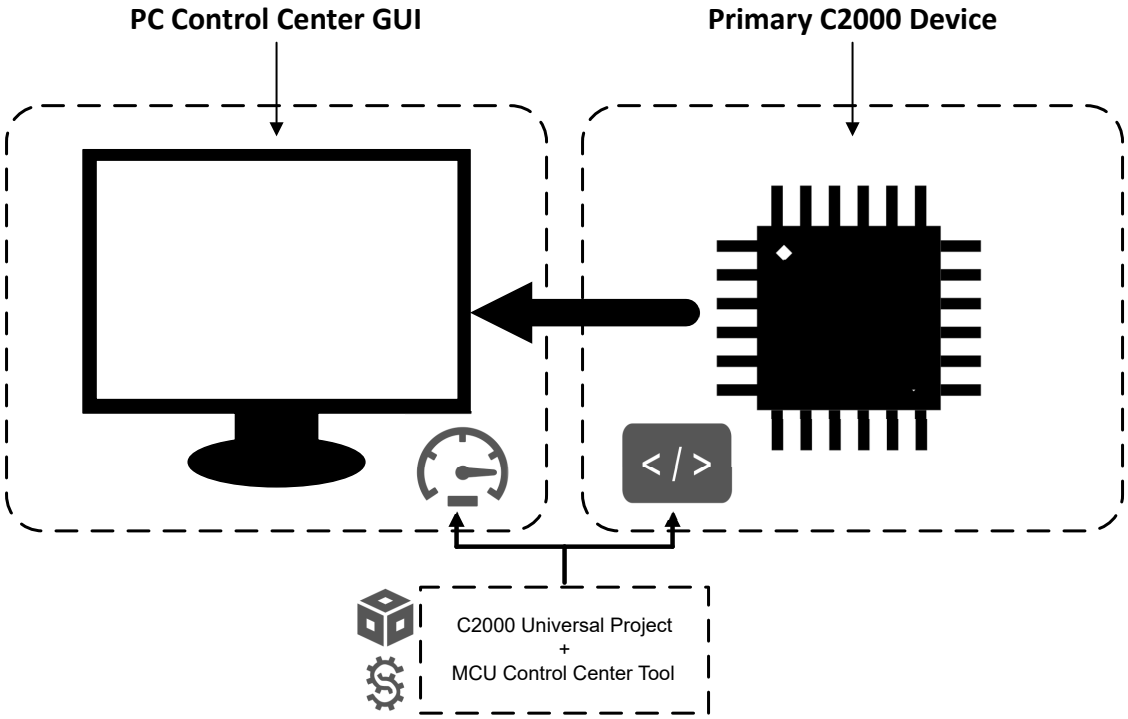
図の一覧

表の一覧

商標

1 はじめに

マイコン制御センターは SysConfig GUI ベースのツールで、通信周辺装置経由でデバイスからデータをエクスポートして、PC 上でこのデータを可視化できます。マイコン制御センターは、通信周辺装置構成、データのパッケージ化とフォーマット構成、PC GUI の作成を取り込むアプリケーションコードを生成します。マイコン制御センターの目的は、開発者が SysConfig ツールを使用してデータロギングと可視化を既存のプロジェクトに容易に組み込むことができるようにすることです。このツールで生成された GUI は CCS 内で開くことができます。



注

このドキュメントでは、ユニバーサル非同期レシーバ/トランスミッタ (UART) および シリアル通信インターフェース (SCI) の頭字語がしばしば同じ意味で使用されます。SCI とは UART プロトコルを使用するほとんどの C2000 デバイスに搭載されている周辺装置を指します。

図 1-1. マイコン制御センターの高度な概略図

2 ハードウェア設定オプション

ユーザーが利用できる特定の C2000 デバイスおよびハードウェアに応じて、データロギングをプロジェクトに統合するために使用できる 4 つの一般的なハードウェア設定があります。最初の手順は、ロギング (プライマリ) デバイスで使用できる周辺装置を判断することです。マイコンと PC の間の通信リンクを作成するには、ハードウェア設定のどこかに、PC との通信に一般的に使用できるプロトコルである USB 周辺装置を配置する必要があります。

すべての C2000 デバイスには、標準の UART プロトコルを使用してデータを記録できる SCI または UART の周辺装置が搭載されています。テキサス インスツルメンツが販売するすべての controlCARD と LaunchPAD™ に、UART メッセージを PC で必要な USB フォーマットに変換できる、フラッシュ済み UART-USB ブリッジデバイスがボード上に搭載されているため、このプロトコルをロギングデバイスで使用することには利点があります。

このプロトコルを使用する際の欠点は、SCI および UART の周辺装置の最大伝送速度が比較的低いことで、速度が重視される用途では性能に影響します。このシナリオでは、高速シリアルインターフェイス (FSI) と呼ばれる周辺装置を使用して高速でデータをログに記録できます。特定の C2000 デバイスには FSI 周辺装置が存在しており、この目的で使用できます。利用可能な周辺装置と最大速度を確認するには、『C2000 リアルタイムマイクロコントローラ周辺装置ユーザーガイド』で C2000 デバイスを検索してください。マイコン制御センターでは第 3 世代以降のデバイスでのみ利用可能な Sysconfig GUI を使用する必要があるため、このツールで使用できるデバイスはこれらだけであることに注意してください。

注

F2838x には UART モジュールがありますが、そのモジュールにはデバイスの CM コアからのみアクセスできます。Sysconfig は CM コアをサポートしていないため、マイコン制御センターを F2838x の UART 周辺装置と組み合わせて使用することはできません。

2.1 設定 #1

最初のハードウェア設定は、データロギング (プライマリ) デバイスに USB 周辺装置が搭載されている場合にのみ可能です。デバイスは USB プロトコルを使用して PC にデータを直接送信できるため、この設定ではブリッジデバイスは不要です。全体として、プライマリデバイス以外に追加のマイコンが不要なため、これは最も単純なハードウェア設定です。

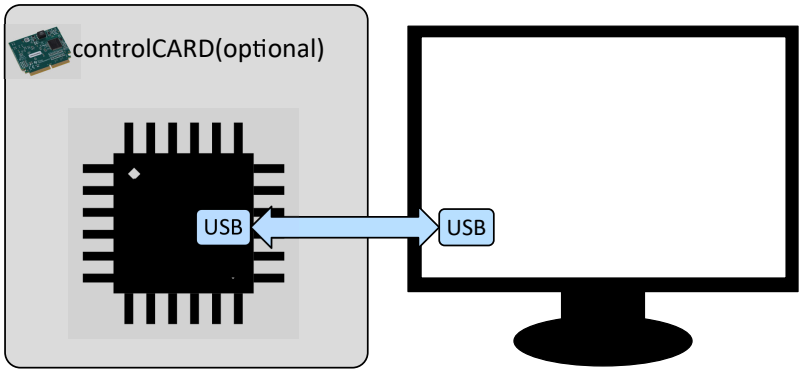


図 2-1. ハードウェア設定 #1

2.2 設定 #2

2 番目のハードウェア設定は非常に一般的な用例であり、Gen 3 およびそれ以降のすべての C2000 デバイスで可能です。この設定では、SCI または UART 周辺装置を使用してデバイスからデータをエクスポートし、ブリッジデバイスを使用してこれらのメッセージを PC が理解できる USB プロトコルに変換します。LaunchPAD または controlCARD を使用する場合、プライマリデバイスとブリッジデバイスの両方がすでに存在し、適切に配線されています。ボード上のブリッジデバイスも、必要なプロトコル変換を実行するプログラムでフラッシュ済みです (ほとんどの場合これが XDS110 プログラムです)。これにより、構成するプライマリデバイスに SCI/UART コードが残され、マイコン制御センターツールで自動的に実行できます。詳細については以降のセクションを参照してください。

ユーザーが自由に使用できる LaunchPAD または controlCARD がなく、スタンドアロン部品蚤の場合 (C2000 の部品がすでにカスタムボードにはんだ付けされているなど)、追加のブリッジデバイスが必要です。この種のブリッジデバイスは組込み開発で一般的に使用されているため、(テキサス インスツルメンツまたはサードパーティーベンダで) 低コストで購入できるハードウェアで利用可能な多くのオプションがあります。

スタンドアロン (USB なし) デバイスと接続するための別のオプションは、USB モジュールを備えた C2000 デバイスを購入し、マイコン制御センターを使用してブリッジアプリケーションの生成とフラッシュ書き込みを行うことです。

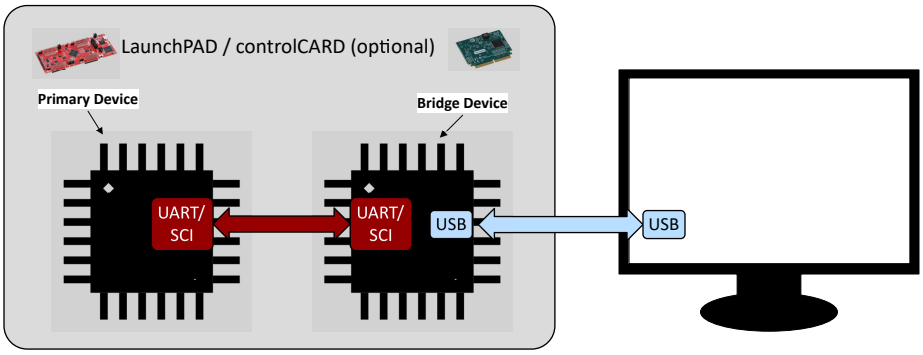


図 2-2. ハードウェア設定 #2

2.3 設定 #3

3 番目のハードウェア設定は、データロギング (プライマリ) デバイスに FSI 周辺装置があり、ブリッジデバイスに FSI と USB の両方の周辺装置がある場合にのみ可能です。この設定では、スタンドアロン部品の FSI 周辺装置を使用してデバイスからデータをエクスポートし、ブリッジデバイスを使用してこれらのメッセージを PC が理解できる USB プロトコルに変換します。ブリッジデバイスに controlCARD を使用する場合、USB 信号は PC に接続できる USB コネクタにすでに正しく接続されています。プライマリデバイスの FSI コードとブリッジデバイス上の FSI-USB コードの両方を構成する必要があり、これらはどちらも 2 つの独立した Sysconfig プロジェクト内で、マイコン制御センターツールにより自動的に実行できます。詳細については以降のセクションを参照してください。

ユーザー自由に使える controlCARD がなく、FSI および USB の周辺装置を備えブリッジデバイスとして使用できるスタンドアロン部品のみである場合、USB ポートに接続するために USB コネクタ回路が必要であることに注意してください。

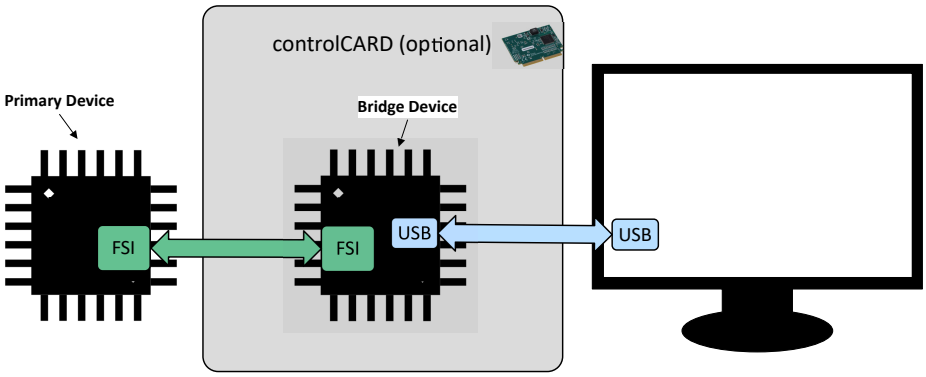


図 2-3. ハードウェア設定 #3

2.4 設定 #4

4 番目のハードウェア設定は、データロギング (プライマリ) デバイスと最初のブリッジデバイスの両方に FSI 周辺装置が搭載されている場合にのみ可能です。この設定では、FSI 周辺装置を使用してスタンドアロンデバイスからデータをエクスポートし、1 つのブリッジデバイスを使用してこれらのメッセージを UART プロトコルに変換し、2 つ目のブリッジデバイスを使用してこれらのメッセージを UART プロトコルから PC が理解できる USB プロトコルに変換します。

C2000 デバイスの LaunchPAD または controlCARD を FSI 周辺装置と組み合わせて使用する場合、両方のブリッジデバイスがすでに存在し、適切に配線されています。ボード上の 2 つ目のブリッジデバイスも、必要な UART と USB 間のプロトコル変換を実行するプログラムでフラッシュ済みです (ほとんどの場合これが XDS110 デバッガです)。これにより、構成するプライマリデバイスの FSI コードと、構成する最初のブリッジデバイスの FSI から UART へのブリッジコードが残ります。これら両方のデバイスの構成は、マイコン制御センターツール (2 つの独立した Sysconfig プロジェクト) で自動的に実行できます。詳細はこのドキュメントの後半で説明しています。

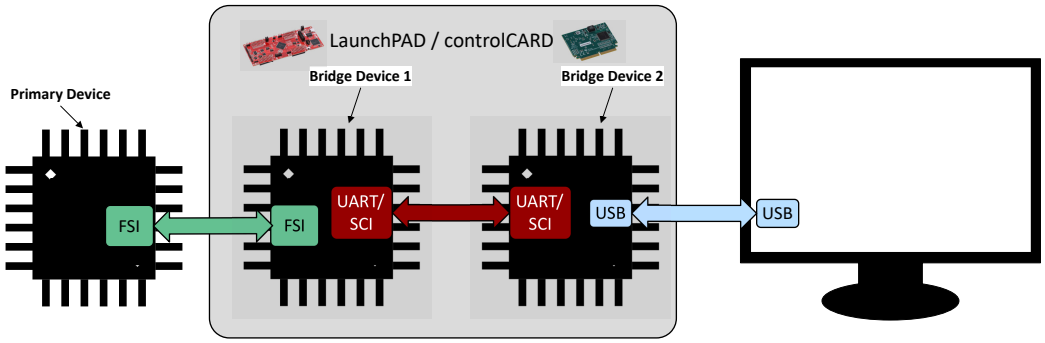


図 2-4. ハードウェア設定 #4

3 ソフトウェアレイヤ

このセクションでは、マイコン制御センターツールのさまざまな機能を実装するためのソフトウェア抽象化レイヤについて説明します。マイコン制御センターモジュールを追加および構成すると、これらのファイルは Sysconfig によって自動生成されます。

表 3-1. SysConfig 生成ファイル

ロギングレイヤ	説明	関連するヘッダー C ファイル
ユーザーアプリケーション	このレイヤは最高の抽象化レベルを提供します。ここでは、各機能のユーザーアプリケーションコード内で呼び出す最上位の API 関数について説明します。(アプリケーションロギング、通信ロガー、ブリッジ、および高速時間ロギング)	logger/coms_logger.h export/export_log.h logger/rt_log.h bridge/bridge.h
パッケージ	このレイヤはログをデバイスから送信する前にパッケージ化します。利用できるパッケージレイヤは JSON と START/END です。	export/export_package.h
バッファ	このレイヤはログメッセージを送信する前にログメッセージのバッファを提供します。これにより、2 つの異なるコード領域でデータをキャプチャおよび送信する柔軟性が得られます。	export/export_buffer.h
輸出管理	このレイヤは、適切な周辺装置のバッファに書き込むことにより、通信周辺装置経由でデータをエクスポートする機能を提供します。	export/export.h

4 GUI の作成

GUI レイヤは PC 上の GUI コンポーザを活用しており、CCS 環境で実行できます。GUI コンポーザの詳細については、「GUI コンポーザのドキュメント」を参照してください。マイコン制御センターの Sysconfig モジュールをプロジェクトに追加すると、GUI コンポーザアプリケーションを作成するための以下のファイルが生成されます。

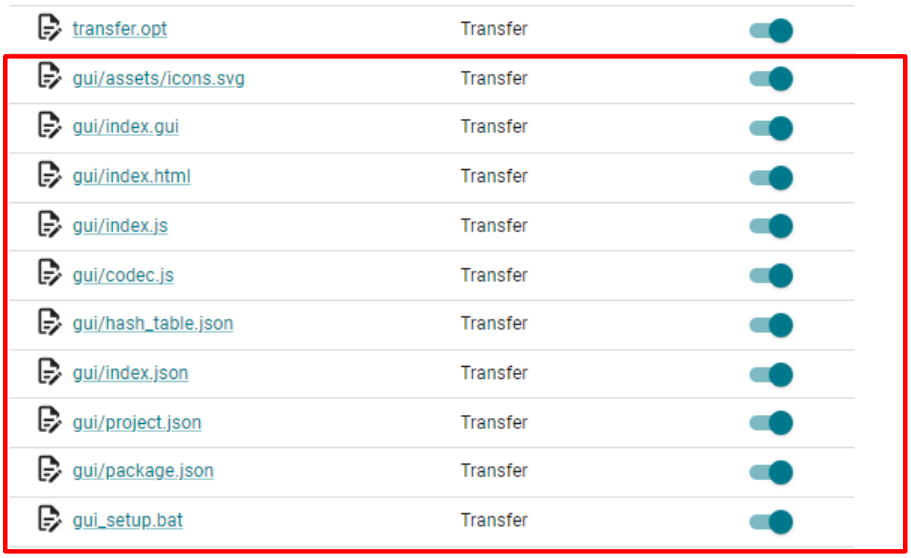


図 4-1. SysCfg が生成する GUI レイヤファイル

CCS のカスタム GUI アプリケーションを生成するには、以下のようなビルドの後処理のステップが必要です。特定のプロジェクトの *[Properties] -> [General] -> [Variables]* に移動し、*GUI_SUPPORT* という変数を追加します。

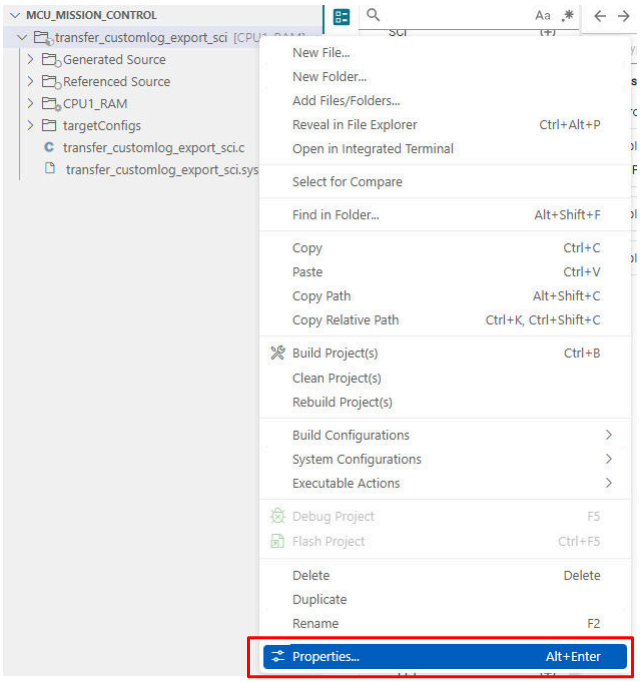


図 4-2. プロジェクトのプロパティを開く

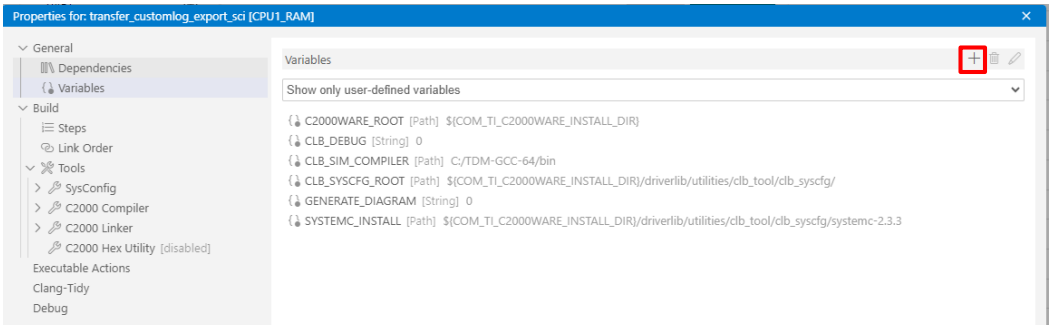


図 4-3. CCS 変数を追加

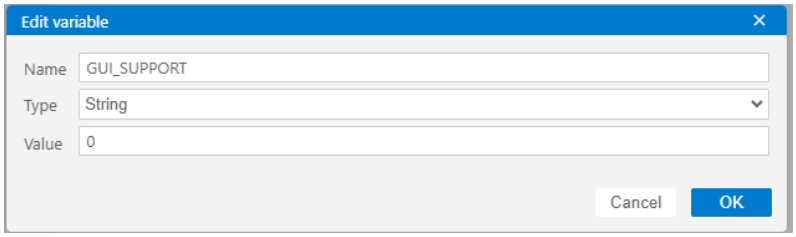


図 4-4. GUI_SUPPORT の変数を追加

さらに、プロジェクトのプロパティで、*[Build]* → *[Steps]* の順に選択して、ビルドの後処理のステップに次の行を追加します:

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

プロジェクトをビルドする前に、GUI_SUPPORT 変数が 1 に設定されていることを確認します。これにより、Sysconfig が自動生成した GUI ファイルが、各プロジェクトビルドの最後にある CCS のインストールフォルダの適切な場所にコピーされます。このフォルダに配置すると、CCS 環境内から GUI アプリケーションを直接起動できます。ビルドすると、自動生成された GUI プロジェクトの名前は以下の Sysconfig 設定で選択した名前と一致します。

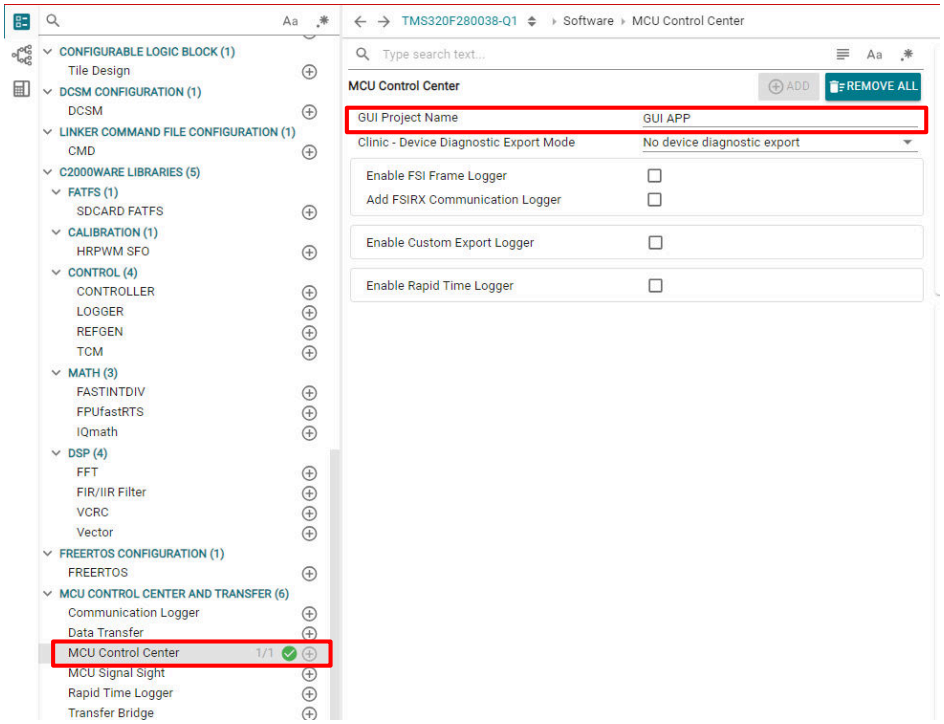


図 4-5. プラグイン名

プロジェクトのビルド後、CCS ビューを更新して、新しく生成された GUI を CCS プラグインのドロップダウンで確認します。*[Help]* → *[Reload Window]* に移動します。

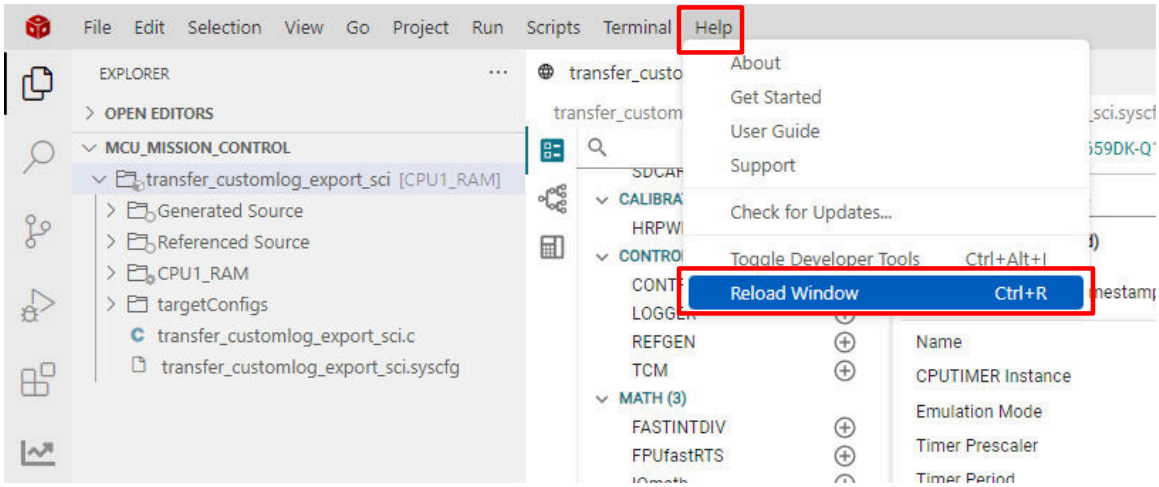


図 4-6. ウインドウの再読み込みで生成されたプラグインを表示

統合 GUI アプリケーションを開くには、*[View]* → *[Plugins]* → *[{NameOfGUI}]* に移動します。クリックすると CCS で GUI が開きます。

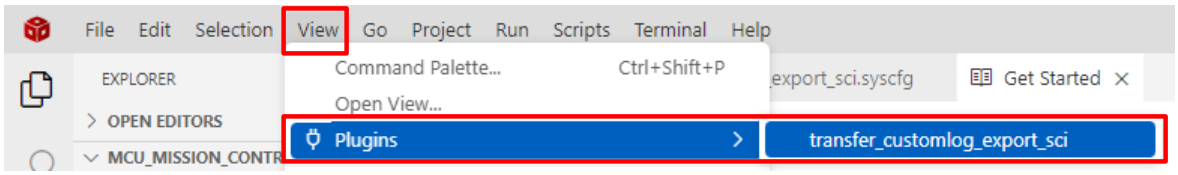


図 4-7. GUI を表示

5 アプリケーションロギング

アプリケーションにマイコンがデータをエクスポートするための要件がある場合、アプリケーションロギングを実装できます。この機能は、文字列、変数値、または配列を PC またはブリッジデバイスにエクスポートできます。アプリケーションロギングは基本的に、デバイス上の通信周辺装置 (SCI/UART など) を使用して、JTAG ではなく印刷メッセージを送信する *printf()* 命令文として機能します。通常の *printf()* の呼び出しと比べ、*EXPORTLOG_log()* 関数は転送速度が高速化され、デバイスのデバッグデータを表示するためのデバッガ接続が不要になります。

この機能は、セクション 2.1 または セクション 2.2 の設定で直接使用でき セクション 2.3 と セクション 2.4 では セクション 6 のモジュールを使用してペアリングされたブリッジデバイスと組み合わせて使用できます。

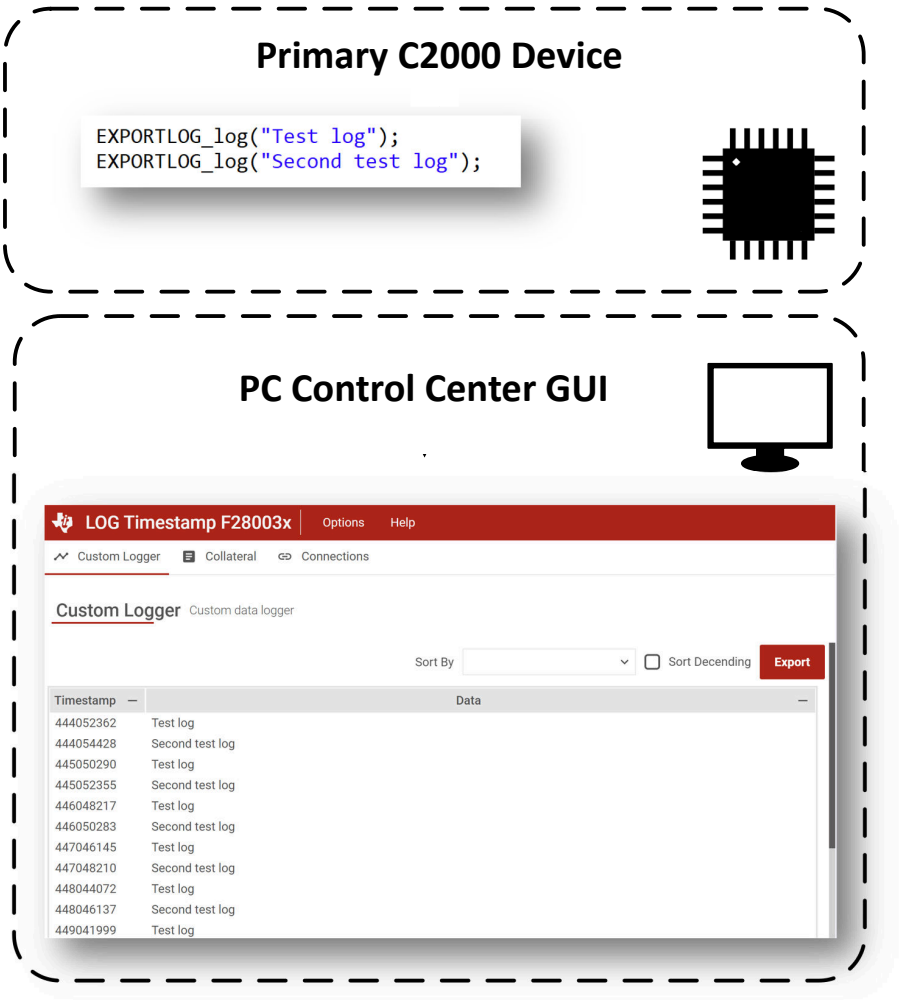


図 5-1. アプリケーションロギング図

5.1 アプリケーションログの手引き

アプリケーションロギングの実装と 表 3-1 に関連するすべてのレイヤの概要を以下に示します。アプリケーションロギングを使用すると、符号付き整数、符号なし整数、浮動小数など、さまざまなデータ型の文字列と配列を出力できます。

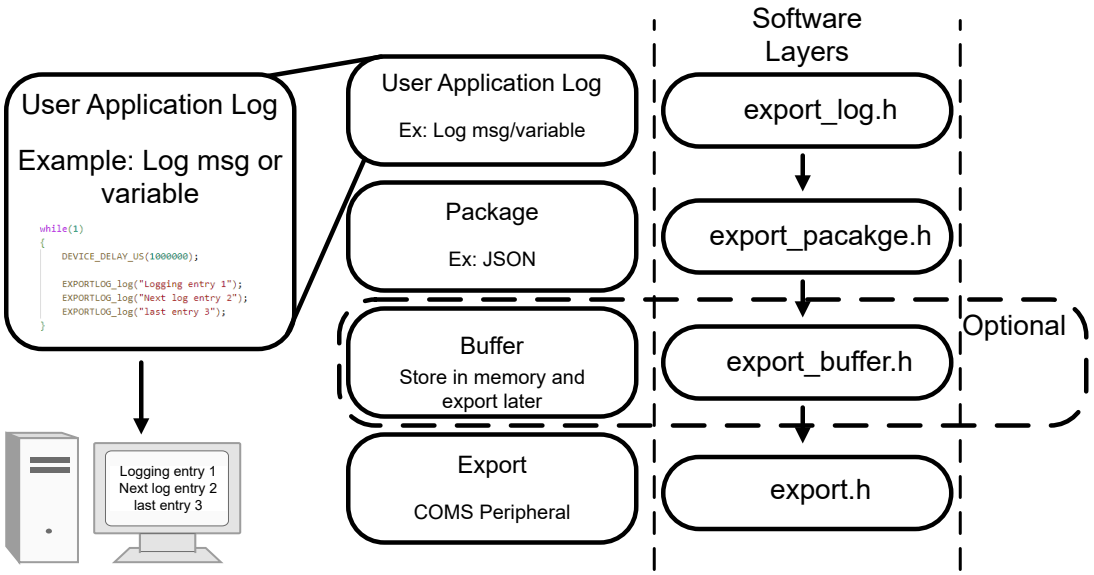


図 5-2. ハイレベルアプリケーションロガー

この機能を CCS プロジェクトに簡単に追加するには、以下の手順を実行します。

SysConfig 構成

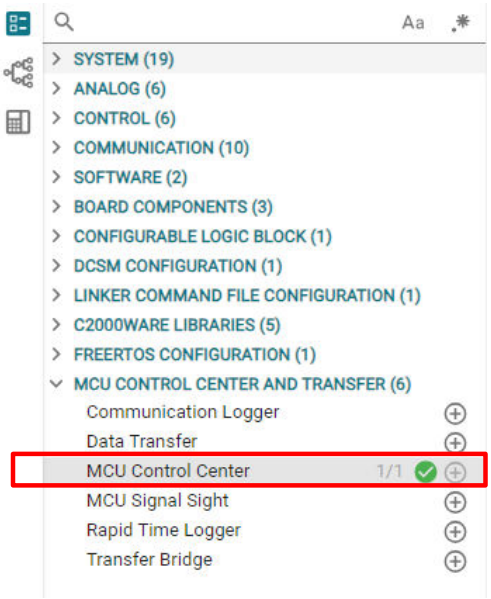


図 5-3. マイコン制御センターモジュールを追加

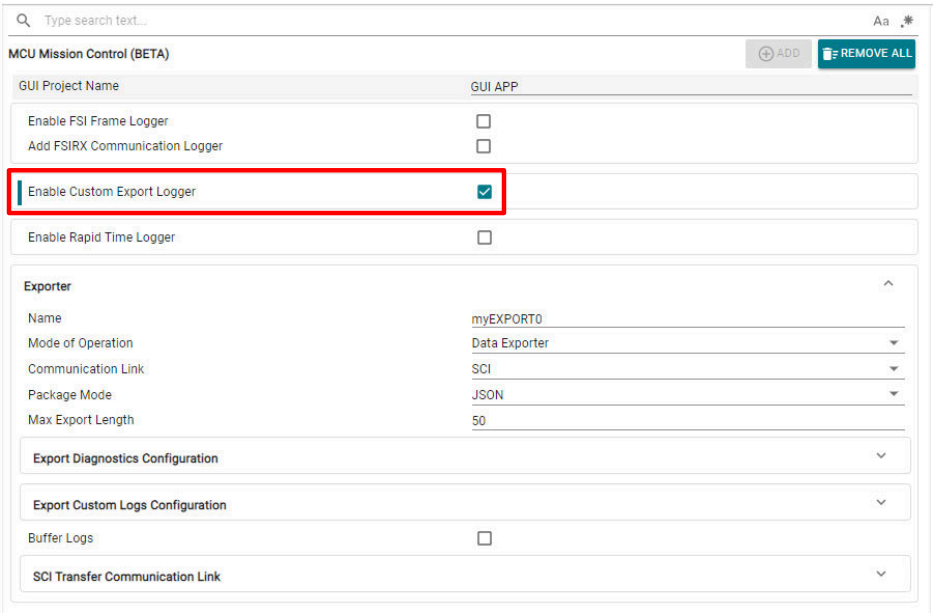


図 5-4. カスタムエクスポートロガー構成

必要な通信周辺装置のエクスポータモジュールを構成します。この手引きの例では、SCI を通信周辺装置として使用しています。*通信リンクの転送*設定に、デバイスで使用可能な通信周辺装置が SysConfig によって自動的に追加されます。一部のデフォルト設定は編集可能です。

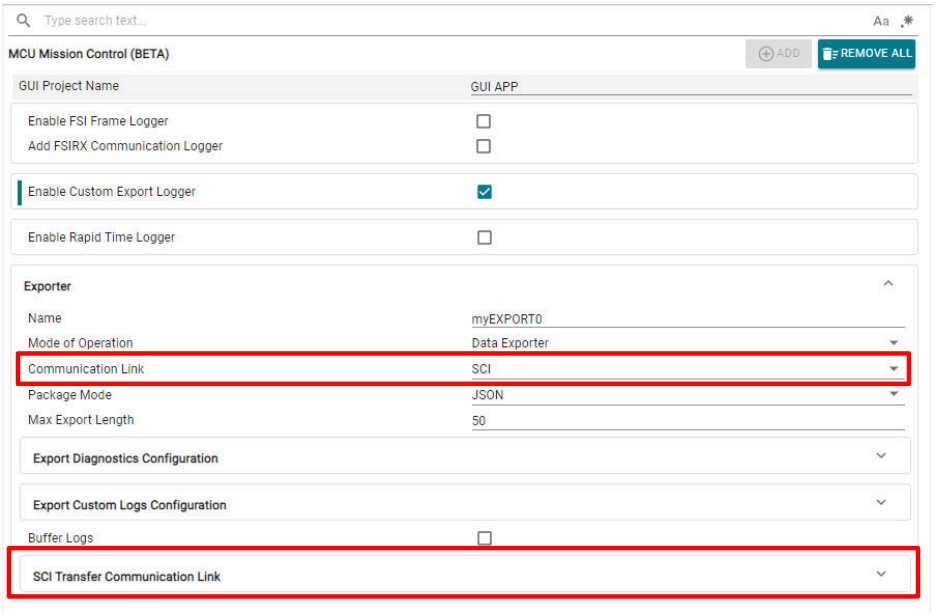


図 5-5. サブモジュールのエクスポート構成

エクスポータサブモジュールの追加に関連する、すべての構成可能な通信周辺装置設定を以下に示します。

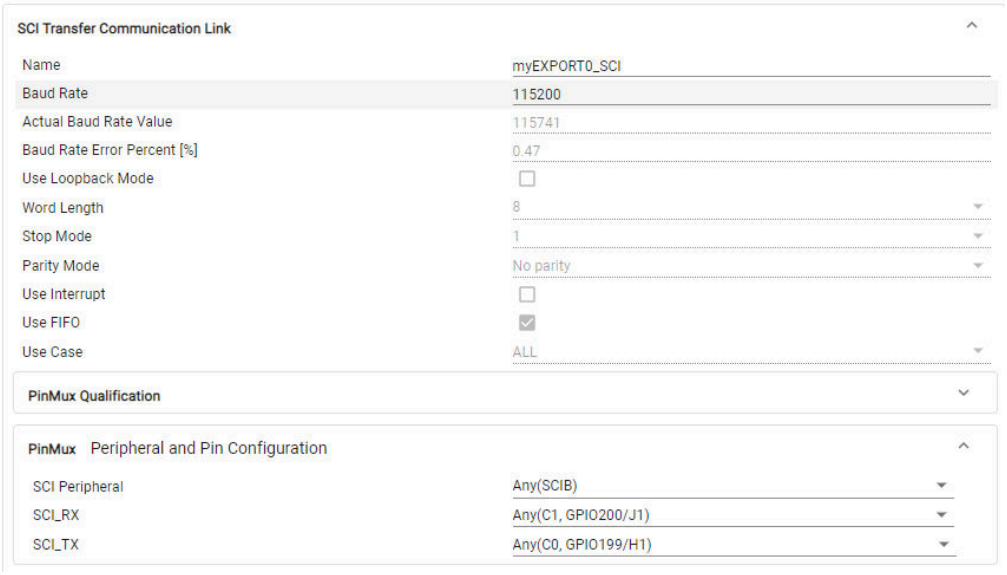


図 5-6. 転送通信リンク

データログキャプチャを拡張するために追加できるその他の設定は、カスタムログのエクスポート構成にあります。[ログサポートのエクスポート] チェックボックスは、エクスポートまたは export_log.h で生成されたファイルで使用できる機能を生成します。

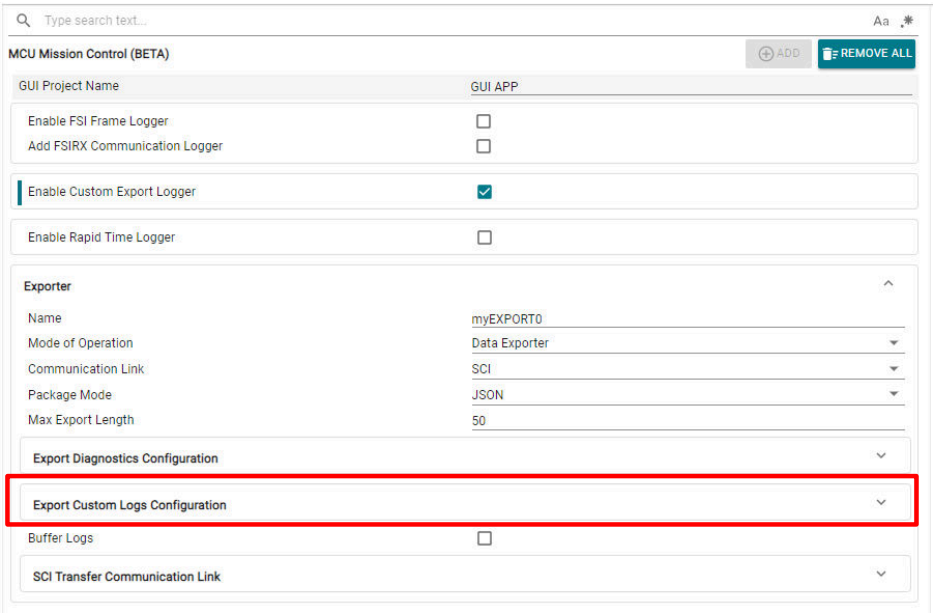


図 5-7. カスタムログのエクスポート構成

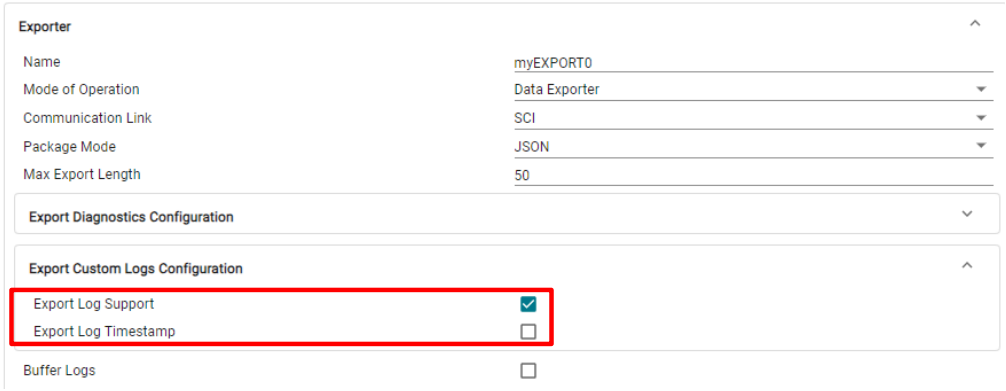


図 5-8. ログサポートとタイムスタンプをエクスポート

ログタイムスタンプのエクスポートは、ログがキャプチャされた時刻を記録するために追加できる別の機能です。ログタイムスタンプのエクスポートが有効になっている場合、構成する CPU タイマモジュールが SysConfig によって自動的に追加されます。

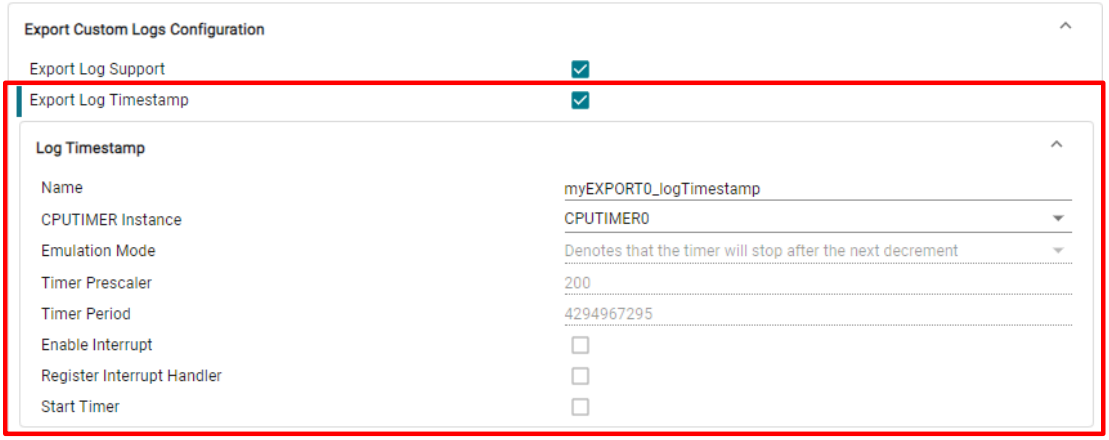


図 5-9. ログタイムスタンプのエクスポート構成

アプリケーション コード

これらは、GUI およびデータロギングソフトウェアレイヤに追加する必要があるすべてのモジュールです。データログ文字列または変数に必要な残りの手順は、アプリケーションコードから export/export_log.h レイヤヘルパー関数を呼び出すことだけです。必ずメインコードの先頭に export/export_log.h ファイルを含めてください。

インクルードファイル

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

プロジェクトを構築する前に、セクション 4 を調べて GUI_SUPPORT を有効にしてください。これを有効にしたら、プロジェクトを構築して CCS 内で GUI を開きます。

アプリケーションコードロギング

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

GUI の COM ポート設定

.Out ファイルをロードしますが、まだ実行しません。[Options] → [Serial Port Settings...] → [<COM Port>] の順に選択し、GUI で正しい COM ポートが選択されていることを確認します

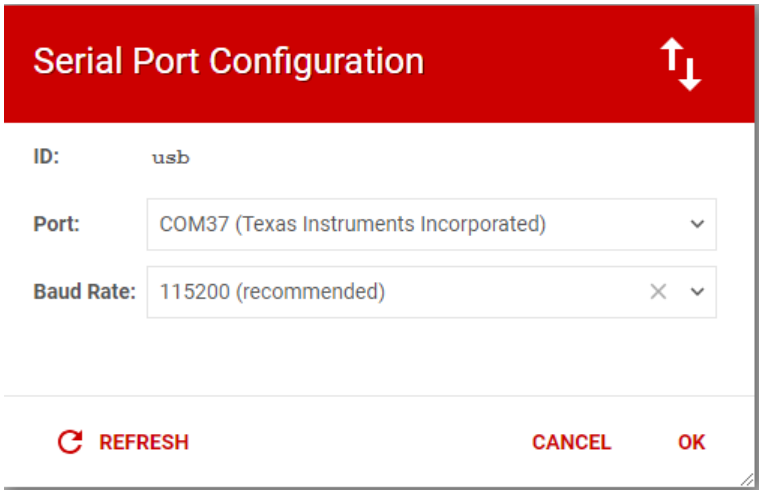


図 5-10. シリアル ポート設定

最終出力

正しい COM ポートを選択したら、アプリケーションコードを実行します。下の図は最終出力を示しています。これらの印刷メッセージが特定のアプリケーションおよびデバッグシナリオで必要であっても、ユーザーはこれらを変更、またはアプリケーション内で EXPORTLOG_log() 関数が呼び出される場所に移動することができます。C2000ware SDK で提供されている他のアプリケーションロギング例の説明については、セクション 9 を参照してください。

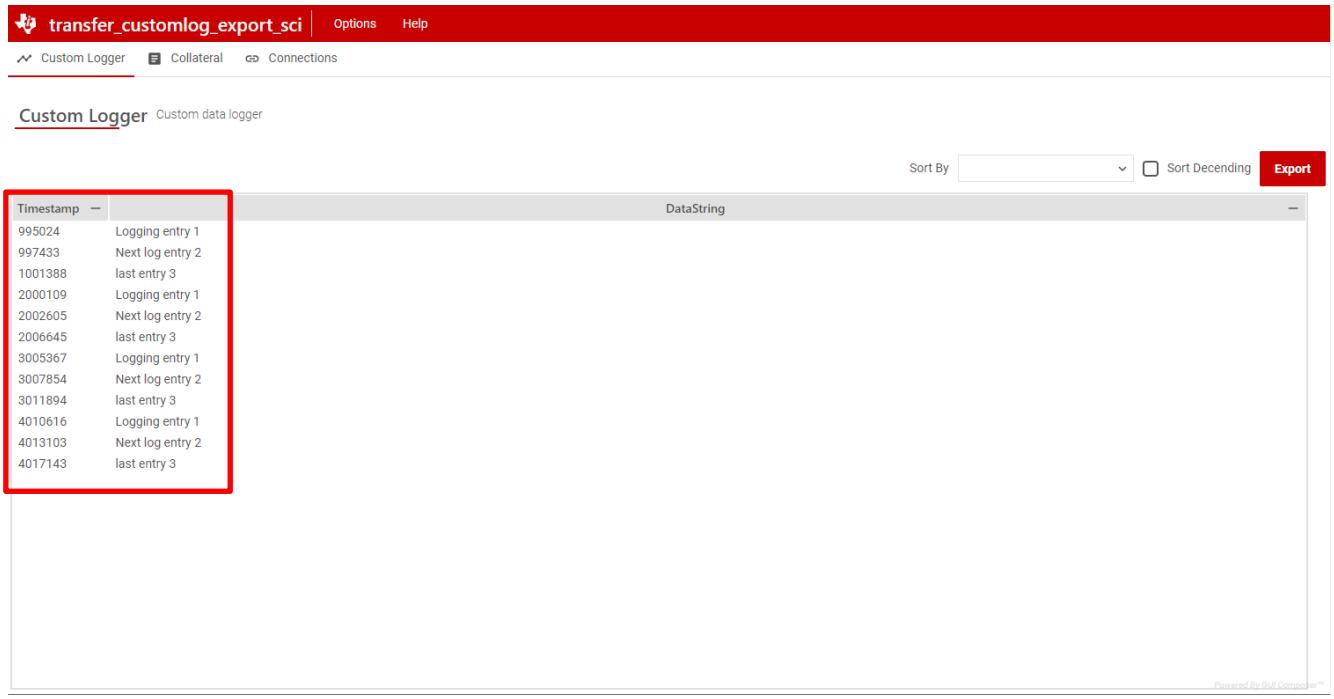


図 5-11. ロガー出力

6 転送ブリッジ

プライマリデバイスからエクスポートされるデータが PC (USB) でサポートされる通信プロトコルにない場合、ブリッジデバイス (または一連のブリッジデバイス) を使用してデータパケットを USB プロトコルに変換できます。UART プロトコルを USB プロトコルに変換するブリッジデバイスは、LaunchPAD、controlCARD、およびサードパーティーベンダの多くで一般的に使用されています。ただし、セクション 2.3 または セクション 2.4 の設定では、FSI-USB または FSI-UART ブリッジファームウェアのいずれかの、異なるタイプのブリッジファームウェアが必要です。ここで、マイコン制御センターツールの転送ブリッジモジュールを使用して、高速通信周辺装置のパケットを受信し、データをバッファしてから、別の通信周辺装置 (UART または USB) を介してデータを転送するコードを生成できます。ブリッジデバイス上のパケットをバッファリングすると、プライマリデバイスからのデータを素早くエクスポートできますが (プライマリアプリケーションに大きな影響を及ぼすことなく)、データを低速で PC に転送することもできます (UART、USB、または USB 経由)。

この概念は 図 6-1 で説明されています。通信リンク B はブリッジデバイス (多くの場合 FSI) とのデータ受信に使用する通信周辺装置を表し、通信リンク A はブリッジデバイスから PC (多くの場合 UART または USB) にこのデータを戻すために使用する通信周辺装置を表しています。オプションとしてバッファが 2 つの通信リンクの間に配置されています。この機能では、通信リンク B が受信したデータパケットのペイロードのみが抽出されて直接通信リンク A に転送されます。転送ブリッジは追加のデータ処理を実行しません。

下の図は、セクション 2.3 の設定で使用されている転送ブリッジ機能を示しています。セクション 2.4 の設定を使用する場合は、ブリッジデバイスと PC の間に追加の UART-USB ブリッジデバイスを配置します。

この機能は セクション 2.2、セクション 2.3、または セクション 2.4 に示す設定や、さまざまなアプリケーションの多くのカスタムハードウェア設定で使用できます。

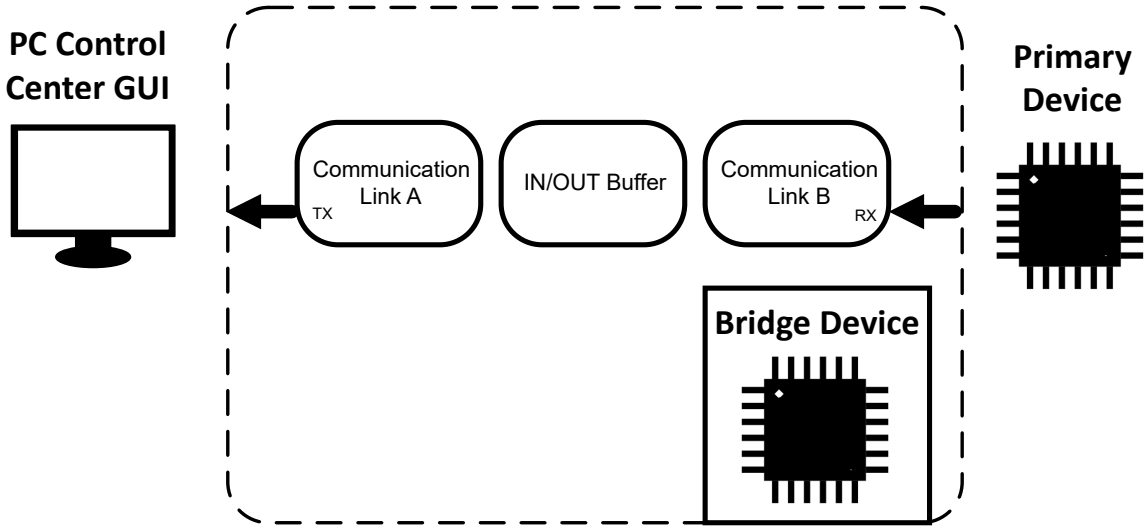


図 6-1. 転送ブリッジ図

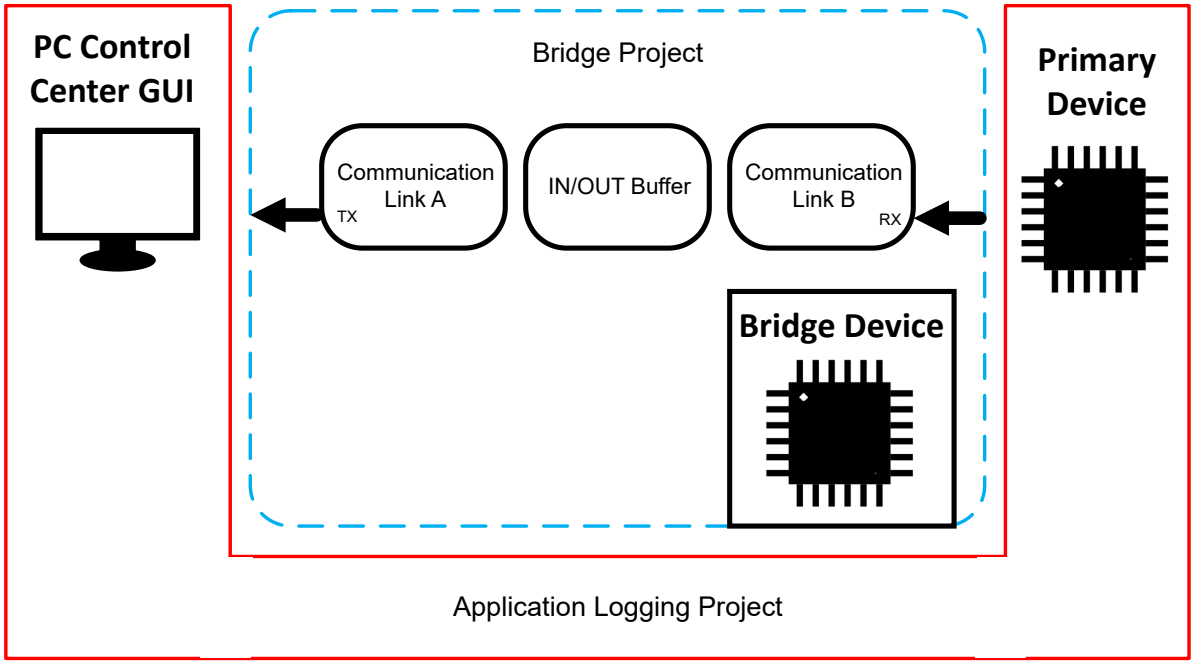
FSI から UART へのブリッジは一般的な用途ですが、転送ブリッジ機能を使用すると他のブリッジタイプのファームウェアも生成でき、さまざまなアプリケーションやハードウェアの設定アップオプションに役立ちます。次の表は、このツールが現在サポートしているさまざまな通信リンクの組み合わせをすべて示しています。一部はこのドキュメントで取り上げるハードウェア設定で使用され、それ以外は特定用途向けです。

表 6-1. サポート対象の通信リンクの組み合わせ

通信リンク B (RX)	通信リンク A (TX)	対応するハードウェア設定
FSI	SCI (UART)	#4
FSI	USB	#3
FSI	SPI	N/A -特定用途向け
SCI (UART)	USB	#2
SCI (UART)	SPI	N/A -特定用途向け
SPI	SCI (UART)	N/A -特定用途向け
SPI	USB	#2 (ただし SPI メッセージ付き)

6.1 転送ブリッジの手引き

ブリッジプロジェクトと 表 3-1 に関係するすべてのレイヤの概要を以下に示します。転送ブリッジモジュールを使用すると、1 つの通信周辺装置でデータを受信し、別の通信周辺装置でこのデータを送信できます。その目的は、2 つのエンドポイント間で送信されるデータの通信プロトコルを、ブリッジデバイスが変革することです (たとえば、FSI フレームを UART に変換)。特定の使用事例において、ブリッジデバイスは、高速シリアル通信プロトコルを使用するデバイスの、低速の通信プロトコルに対するバッファとして機能します。この機能を追加するために必要な手順については、次の手引きで説明します。



この図の青い点線は転送ブリッジプロジェクトで、ある通信プロトコルから別の通信プロトコルにデータを変換するために使用されます。通信リンク B はデータを受信し、通信リンク A を介してデータを送信します。図の赤で示されている部分は主なデバイスプロジェクトで、SCI ではなく FSI が通信リンクとして選択される場合を除き、セクション 5.1 と同じソフトウェアを実します。この手引きでは、ブリッジプロジェクト部分のみを構成する方法に焦点を当てます。

図 6-2. ハイレベルのトランスファーブリッジプロジェクト

ソフトウェアレイヤ

ブリッジプロジェクトに必要なソフトウェアレイヤを以下に示します。オプションとして、バッファレイヤをブリッジプロジェクトに追加して、デバイス上の受信データをエクスポート前に格納するメモリを増やすこともできます。

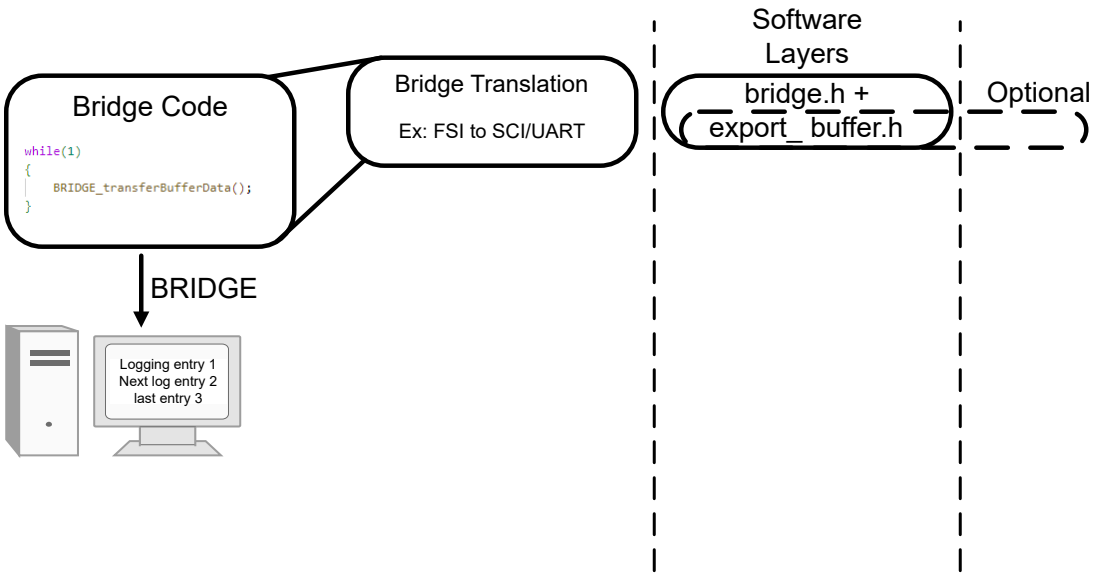


図 6-3. ブリッジプロジェクトのソフトウェアレイヤ

SysConfig 構成

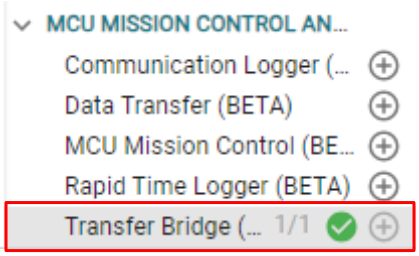


図 6-4. 転送ブリッジモジュール

次の図は、転送ブリッジの Sysconfig モジュール内に自動的に追加される通信リンク A および B を示しています。通信リンク B は、プロトコル B を使用して受信データを受信し、このデータをバッファに格納します (オプション)。通信リンク A は (通信リンク B またはバッファから) 受信したデータを読み取り、プロトコル A でデータを送信します。この手引きでは、ブリッジデバイスが FSI プロトコルパケットを受信して SCI 周辺装置 (UART プロトコル) パケットで送信する、一般的な使用例を示します。

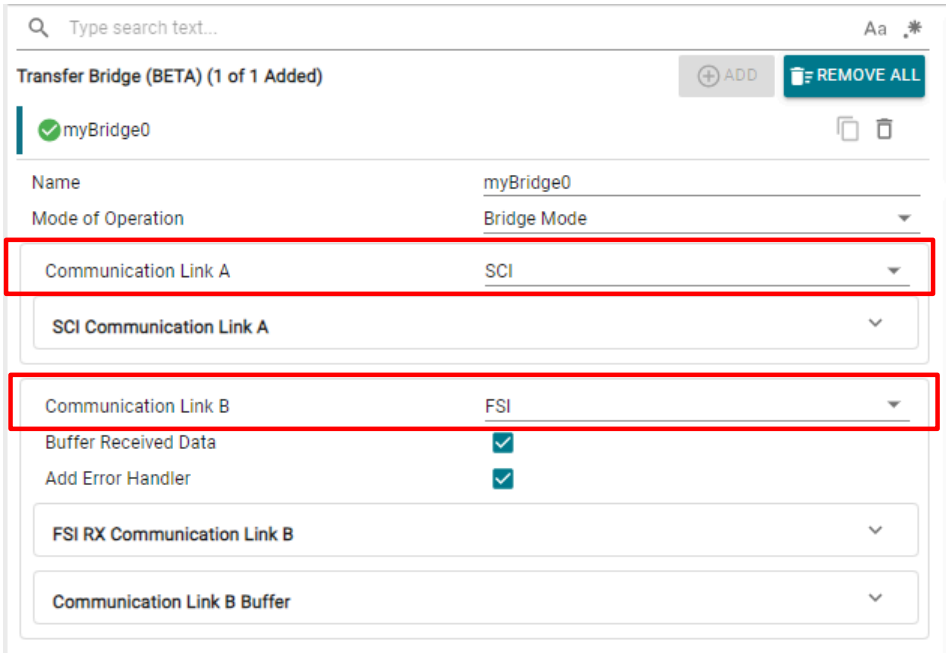


図 6-5. 転送ブリッジの通信リンク

以下の設定はオプションで、ブリッジデバイスにバッファを追加するために使用できます。このオプションを使用すると、バッファがいっぱいになってデータを送信する前に、より多くのデータをデバイスに蓄積できるようになります。この機能は、ロギングデバイスが、ブリッジデバイスが処理できる速度より早い速度でブリッジデバイスにデータを送信する場合に役立ちます。

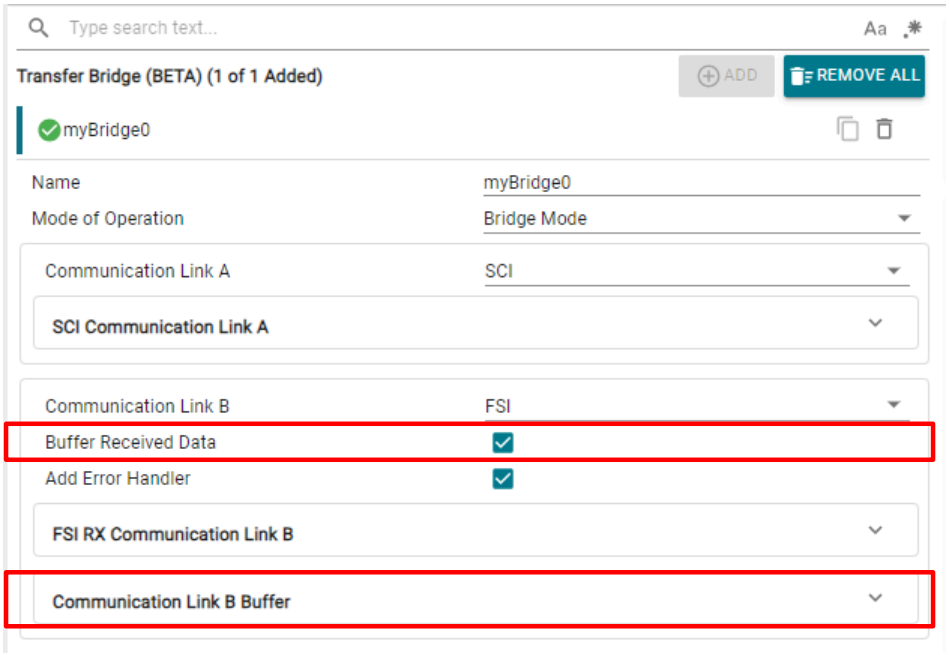


図 6-6. ブリッジバッファ (オプション)

インクルードファイル

```
//
// Included Files
//
```

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

転送ブリッジの初期化

この手順は、バッファが有効な場合にのみ必要です。これを *Board_init()* の後のメイン初期化シーケンスに追加します。

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

転送ブリッジアプリケーションのロギングコード

最後の手順では、データパケットの受信を処理し、変換されたパケットをデバイスから転送するアプリケーションコードを追加します。

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

ブリッジデバイスのテスト

ブリッジデバイスが正常に機能しているかどうかをテストするには、次の手順に従います。これらの手順では、ブリッジデバイスに対して別のデバイスがロギングまたはデータを送信していると仮定しています。

1. 転送ブリッジアプリケーションプログラムでブリッジデバイスをフラッシュ
2. プライマリデバイスの FSITX ピンをブリッジデバイスの FSIRX ピンに接続
   a. プライマリデバイスの FSITX_CLK をブリッジデバイスの FSIRX_CLK に接続
   b. プライマリデバイスの FSITX_D0 をブリッジデバイスの FSIRX_D0 に接続
   c. プライマリデバイスの FSITX_D1 をブリッジデバイスの FSIRX_D1 に接続 (オプション - デュアルデータレーンが*フレーム構成*で構成済みの場合)
3. USB コネクタでブリッジデバイスを PC に接続
4. プライマリデバイスでアプリケーションロギングプロジェクトコードを実行
5. シリアルポートアプリケーションを開いて SCI モジュール構成に合わせてポートを設定

最終出力がシリアル端末ポートに表われます。

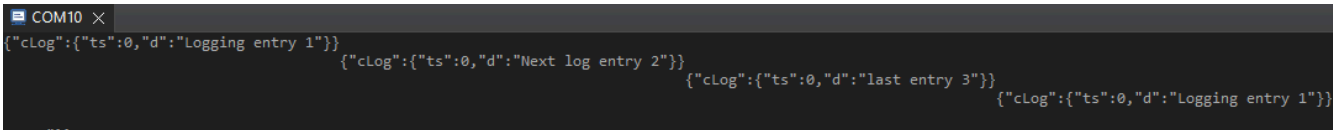


図 6-7. 転送ブリッジの最終出力

7 通信ロガー

多くの通信プロトコルにはペイロード以外の重要なデータが含まれています。たとえば、FSI プロトコルには、各フレームのフレームタイプ、ユーザーデータ、CRC バイト、フレームタグフィールドが含まれます。通信ロガー機能には、各受信データパケットからペイロード以外のデータをさらに抽出する、ブリッジソフトウェアが用意されています。この機能により、ブリッジデバイスは任意のパッケージ形式で通信周辺装置のデータパケットを受信して、そのデータを PC の GUI にエクスポートし分析できます。GUI に送信されるパケットは、メッセージのすべての部分をユーザが確認できるようにフォーマットされます。

使用事例はさまざまですが、この機能の 1 つの用途はアプリケーション内の FSI 実装のデバッグです。この場合、アプリケーションはすでに FSI を使用して他のマイコンと通信していますが、通信エラーまたはデバッグが必要な問題が発生します。通信ロガーは、プライマリデバイスが送信した FSI フレームのすべての部分を、生成された GUI の中で人間が読める形式で表示できます。

別の使用事例は、セクション 6 の機能の使用事例と似ていますが、より多くのメッセージの詳細が GUI に記録される、高速周辺装置を使用したデータロギングです。通信ロガー機能は、プライマリデバイスからの高速 FSI メッセージを受信して (オプションで) バッファに格納し、メッセージ内容のすべての要素を抽出し、UART または USB 経由で PC に送信し GUI で視覚化するために使用します。

下の図は、セクション 2.3 の設定で使用されている通信ロガー機能を示しています。この機能では、ブリッジデバイスと GUI 間のデータのパッキング/アンパッキング方式を広く理解する必要があるため、ブリッジデバイスの Sysconfig プロジェクトを使用して制御センターの PC GUI を生成することに注意してください。セクション 2.4 を使用する場合は、UART から USB への追加のブリッジデバイスをブリッジデバイスと PC の間に配置します。

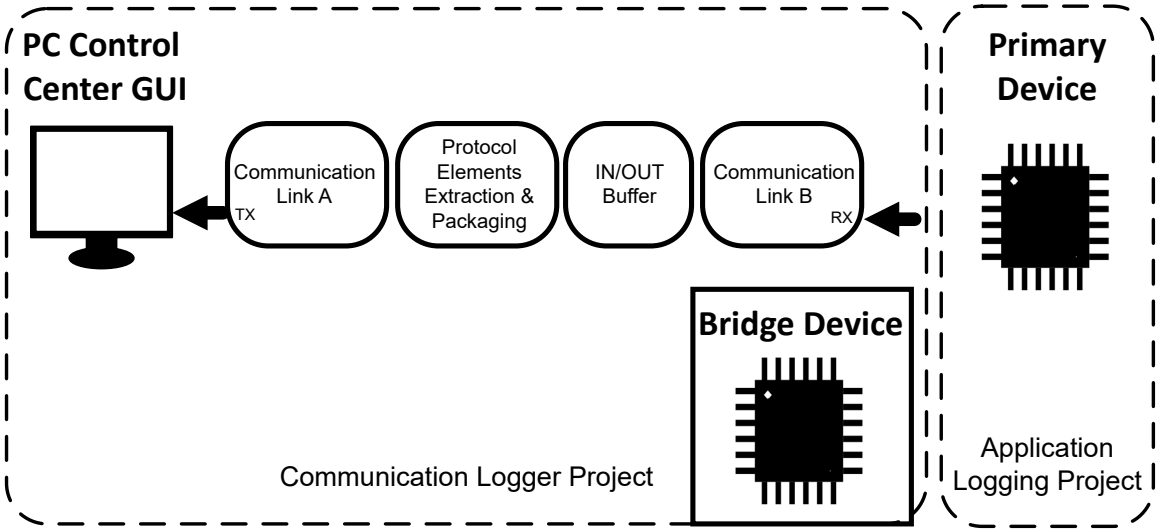


図 7-1. 通信ロガー図

注

現在、この機能は受信プロトコル (プロトコル B) として FSI プロトコルのみをサポートしています。

7.1 通信ロガーの手引き

通信ロガー機能と表 3-1 に関連するすべてのレイヤの概要を以下に示します。通信ロガー機能では、受信した FSI フレームの各要素を制御センター GUI のロギングテーブルの別の列に表示します。この機能を CCS プロジェクトに追加するために必要な手順については、次の手引きで説明します。

ソフトウェアレイヤ

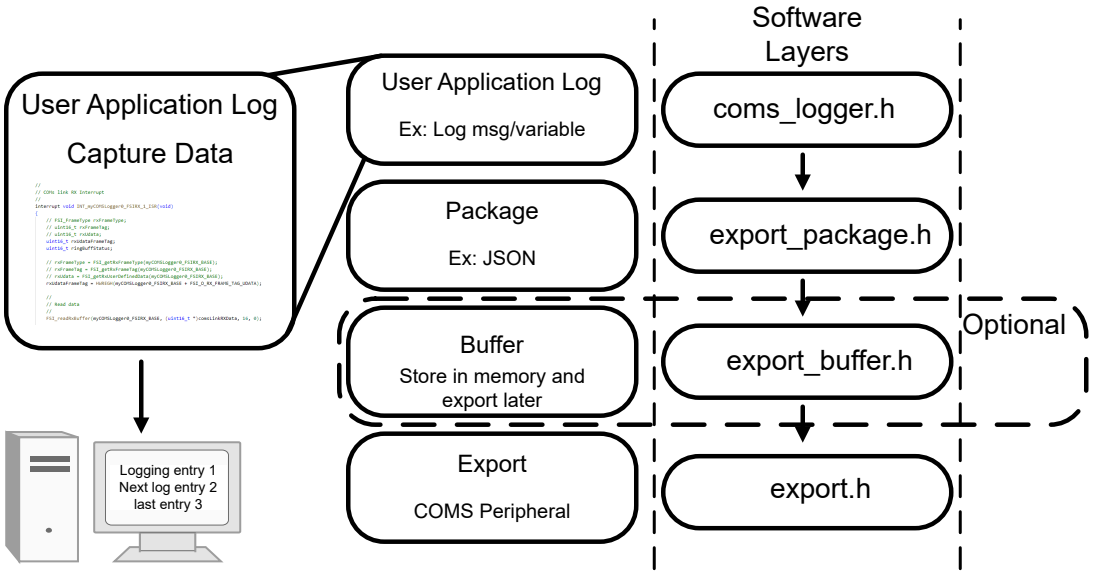


図 7-2. 通信ロガーソフトウェアレイヤ

Sysconfig 構成

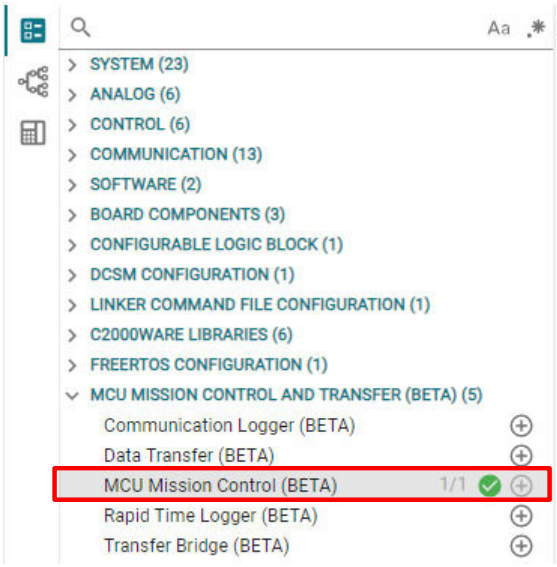


図 7-3. マイコン制御センターモジュール

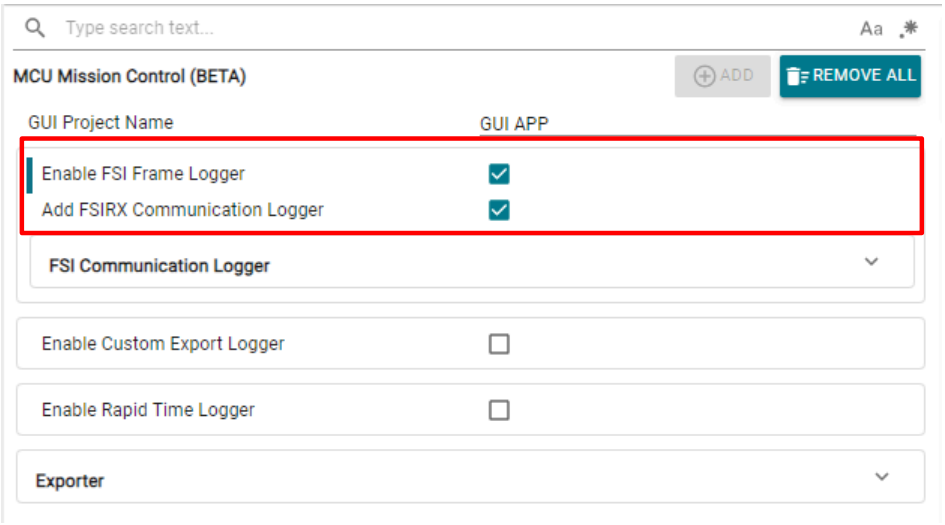


図 7-4. FSI ロガーと通信ロガーを有効にする

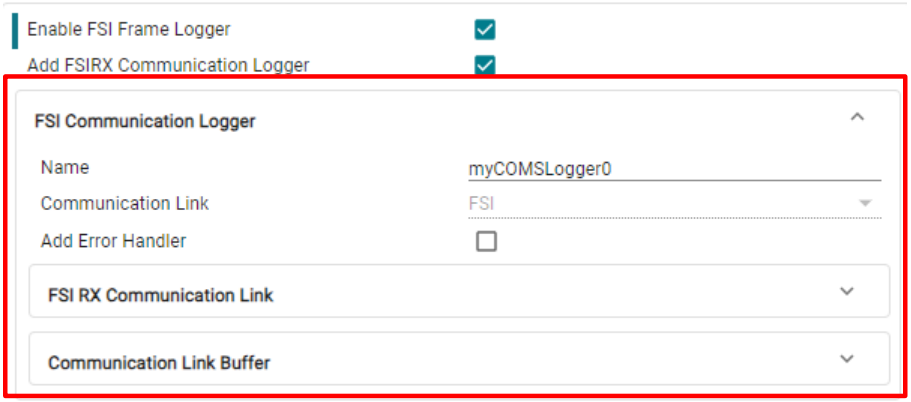


図 7-5. FSI 通信ロガー構成

インクルードファイル

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

通信ロガー初期化

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

通信ロガーのアプリケーションコード

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

通信ロガーのエラー処理 (オプション)

```
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;
}
```

プロジェクトの構築

通信ロガーのアプリケーションコードを構築し、セクション 4 の手順に従って CCS 内で GUI を生成します。

通信ロガーツールのテスト

1. 通信ロガーアプリケーションをブリッジデバイスにフラッシュ
2. ブリッジデバイスの FSIRX ピンをプライマリデバイスの FSITX ピンに接続
   a. プライマリデバイスの FSITX_CLK をブリッジデバイスの FSIRX_CLK に接続
   b. プライマリデバイスの FSITX_D0 をブリッジデバイスの FSIRX_D0 に接続
   c. プライマリデバイスの FSITX_D1 をブリッジデバイスの FSIRX_D1 に接続 (オプション - デュアルデータレーンが*フレーム構成*で構成済みの場合)
3. USB コネクタを使用してブリッジデバイスを PC に接続
4. プライマリデバイスでアプリケーションロガーアプリケーションプロジェクトを実行
5. CCS 内で生成した GUI を開く
6. 次のような最終出力が表示

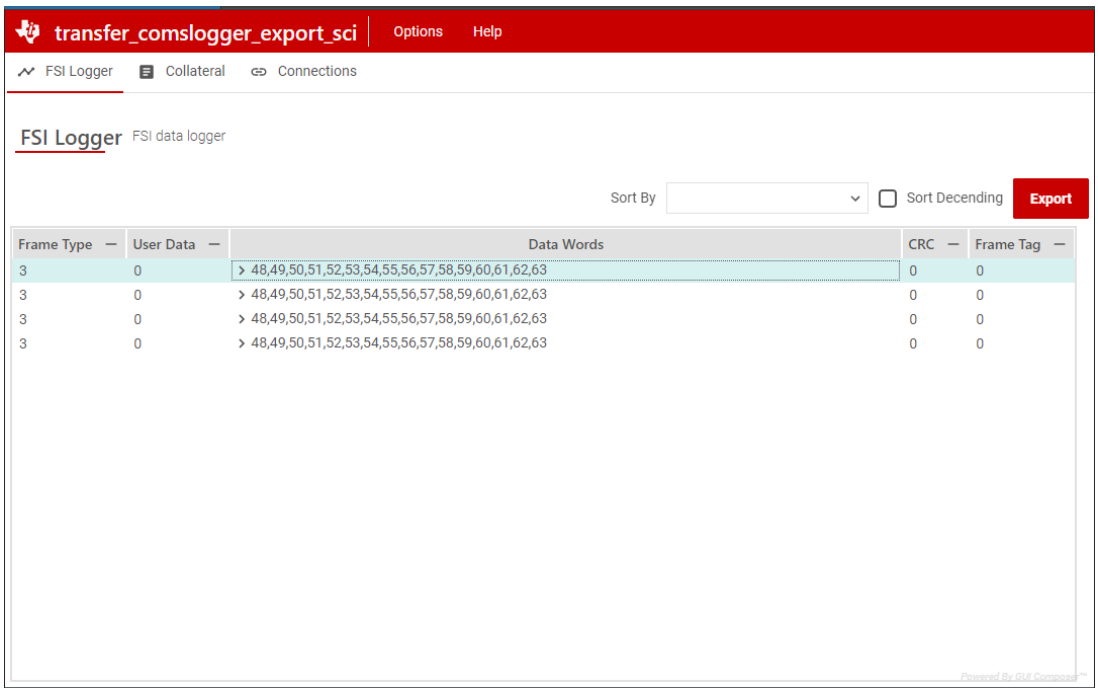


図 7-6. 最終出力

8 高速時間ロガー

以前は、通信ロガー機能により、各 FSI パケットで提供された追加データ要素の表示が有効でした。高速時間ロガーは、ユーザーデータ要素を使用してロギングメッセージのタイプをエンコードし、フレームタイプ要素を使用してアプリケーションから送信される変数を格納することで、これらの追加要素を利用します。この形式のデータロギングでは、FSI プロトコル固有の FSI パッケージ機能をアプリケーションのパッケージレイヤとして活用するため最も高速です。セクション 2.3 または セクション 2.4 に示す設定では、ブリッジデバイス上に通信ロガー機能を特別に実装しており、プライマリデバイス上で高速時間ロガー機能を使用できます。

一般的にこの機能は、プライマリアプリケーションのタイミング要件が厳しい場合に、アプリケーション変数のデバッグや表示に使用されます。プライマリデバイスで FSI プロトコル機能を最大限に活用すると、データを高速に、最適化して転送することができます。同時に、ブリッジデバイスはすべての FSI 素子からデータを抽出して PC に送信します (USB ペリフェラル経由で直接、または UART を使用したセカンダリ UART-USB ブリッジデバイス経由)。GUI アプリケーションで使用するため、各アプリケーション変数とメッセージタイプのエンコーディングを含む JSON ファイルが、高速時間ロガーの Sysconfig モジュールによって生成されます。これにより、FSI パケットで文字をさらに送信することなく、各データパケットの意味が分かるように GUI で表示できます。

8.1 高速時間ロギングの手引き

高速時間ロガー機能と 表 3-1 に関連するすべてのレイヤの概要を以下に示します。プライマリデバイスで高速時間ロガー機能を有効にする手順と、ブリッジデバイスで拡張通信ロガー機能を有効にする手順については、次の手引きで説明します。

ソフトウェアレイヤ

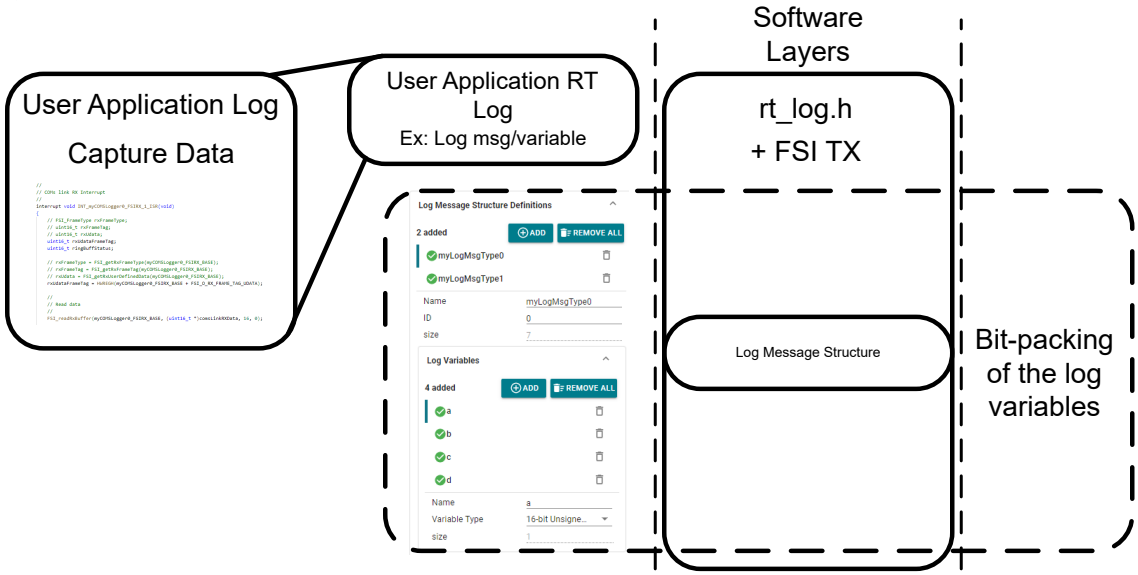


図 8-1. 高速時間ロガーソフトウェアレイヤ

Sysconfig 構成

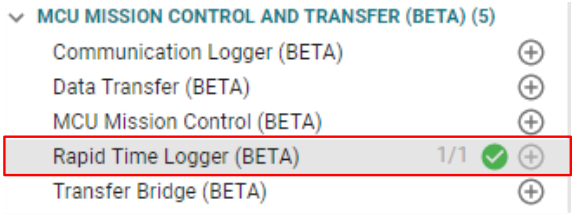
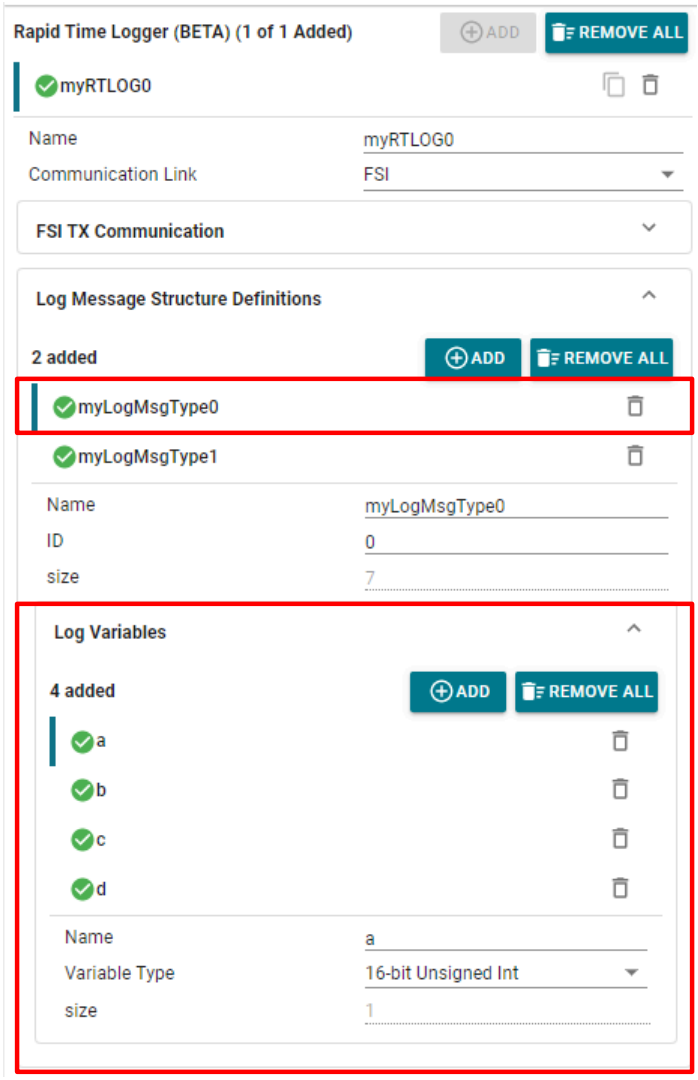


図 8-2. 高速時間ロガー SysConfig

この手引きでは、GUI に 2 つのメッセージ構造の例が追加されています。*Log Message Structure* ビューで、2 つのインスタンスを追加します。各インスタンスは、パケットで指定された特定のメッセージ構造を表します。異なるメッセージ構造の定義を、アプリケーション固有のさまざまな目的に使用できます(たとえば、メッセージタイプ 0 が一部のイベントの前に送信され、メッセージタイプ 1 がアプリケーション内の一部のイベントの後に送信)。この例では、最初のログメッセージ構造の定義に *a*、*b*、*c*、*d* という 4 つの変数を追加します。2 番目のログメッセージ構造の定義で、*e* という名前の別の変数を追加します。

表 8-1. ログ変数設定の例

変数	変数タイプ
a	16 ビット、符号なし整数
b	32 ビット、符号なし整数
c	32 ビット浮動小数点
d	16 ビット符号なし整数の配列 配列の長さ:2
e	32 ビット浮動小数点の配列 配列の長さ:8



上の図では、各ログ変数に一意の名前、変数タイプ、サイズを指定できます。サイズは変数タイプに基づいて自動的に計算されます。この手引きでは、各構造タイプのテーブルに従って a、b、c、d、e を構成します。

図 8-3. ログメッセージ構造 0 の定義

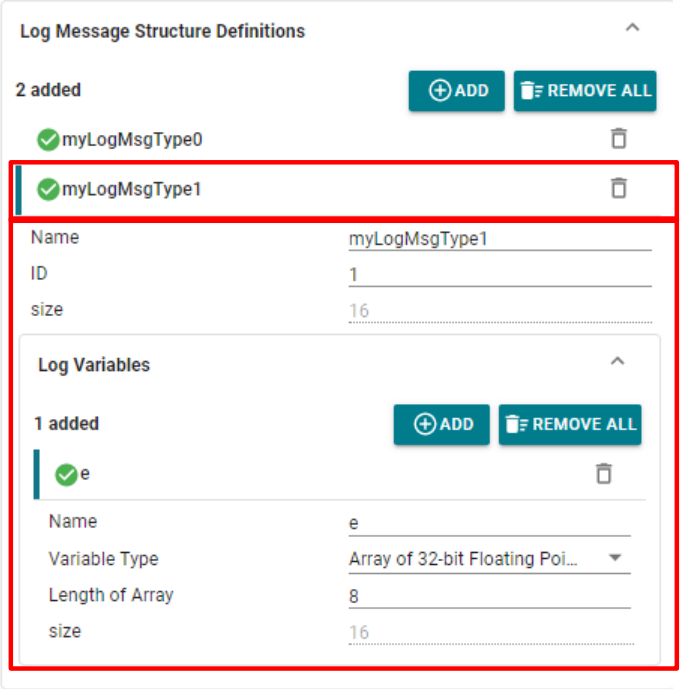


図 8-4. ログメッセージ構造 1 の定義

インクルードファイルとグローバル変数

```
//
// 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;
```

高速時間ロガーの初期化

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

アプリケーションコードに高速時間ログを追加

```
// 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;
```

通信ロガーの追加手順

残りの手順は、セクション 7.1 で説明されているように、ブリッジデバイスの通信ロガー機能を設定して、その他の手順を追加するだけです。TI は FSI TX フレームを経由して送信されるすべての変数のエンコードを含む JSON ファイルを提供しています。このファイルは通信ロガーが高速時間ロガーのメッセージをデコードするために使用します。以下の手順は、プライマリプロジェクトを構成し、ブリッジプロジェクトで セクション 5.1 の手順を実行した後に必要となります。

1. マイコン制御センターの Sysconfig モジュールで *Enable Rapid Time Logger* を選択します。

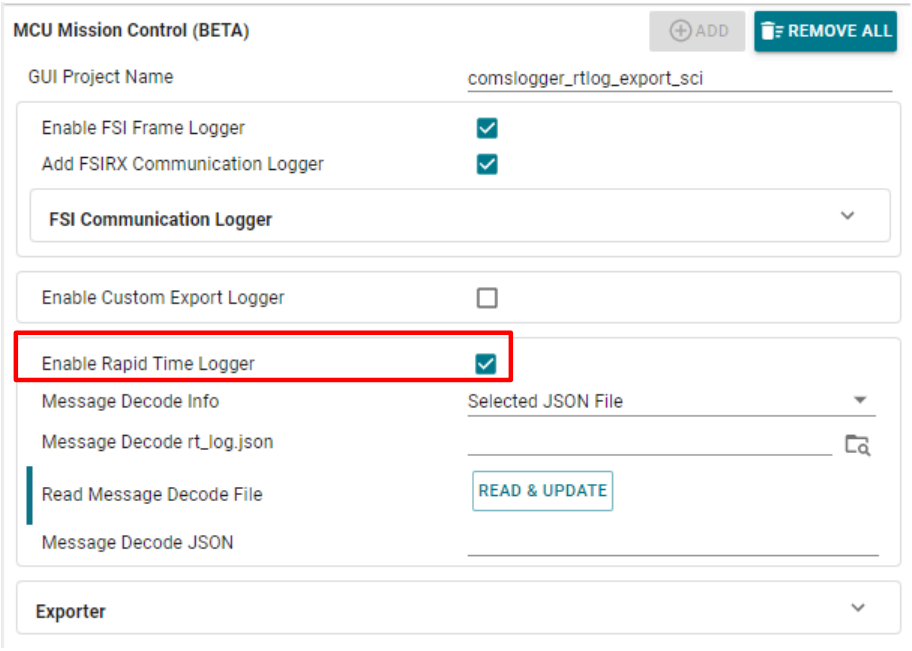


図 8-5. 高速時間ロガーを有効にする

2. 高速時間ロガープロジェクトの build フォルダにある rt_log.json ファイルを検索します。
   a. たとえば、<workspace_ccs>/<name_of_project>/<build_folder>/syscfg/logger/rt_log.json

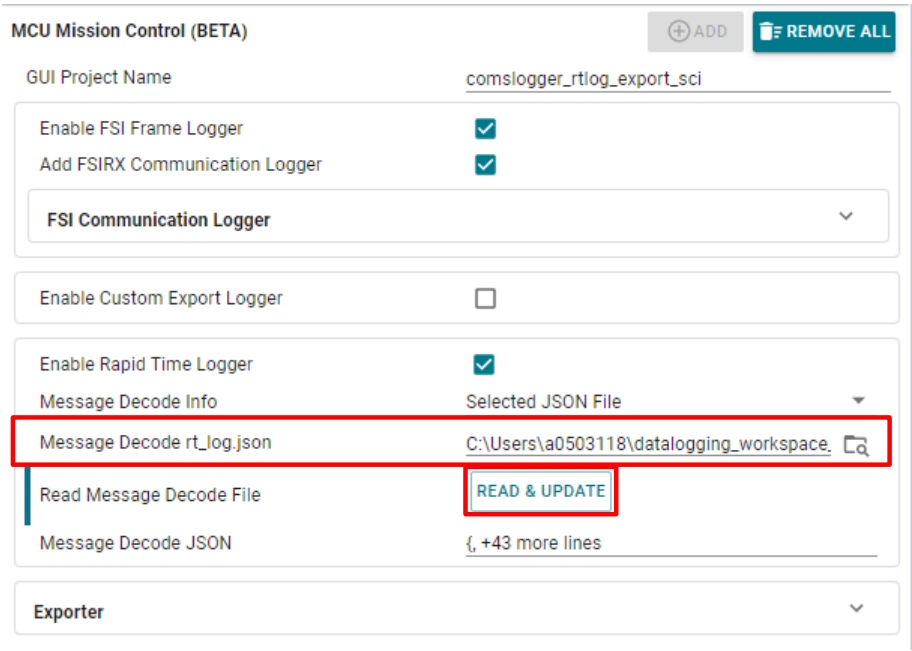


図 8-6. JSON rt_log.json ファイルに移動

プロジェクトの構築

通信ロガーのアプリケーションコードを構築し、セクション 4 の手順に従って CCS 内で GUI を生成します。

リアルタイムロガー機能のテスト

1. 通信ロガーアプリケーションプロジェクトをブリッジデバイスにフラッシュ
2. 通信ロギングデバイスの FSIRX ピンをリアルタイムロギングデバイスの FSITX ピンに接続
   a. プライマリデバイスの FSITX_CLK をブリッジデバイスの FSIRX_CLK に接続
   b. プライマリデバイスの FSITX_D0 をブリッジデバイスの FSIRX_D0 に接続
   c. プライマリデバイスの FSITX_D1 をブリッジデバイスの FSIRX_D1 に接続 (オプション - デュアルデータレーンが*フレーム構成*で構成済みの場合)
3. USB コネクタを使用してブリッジデバイスを PC に接続
4. プライマリデバイスで高速時間ロガーアプリケーションプロジェクトを実行
5. CCS 内で生成した GUI を開く
6. 次のような最終出力が表示

最終出力

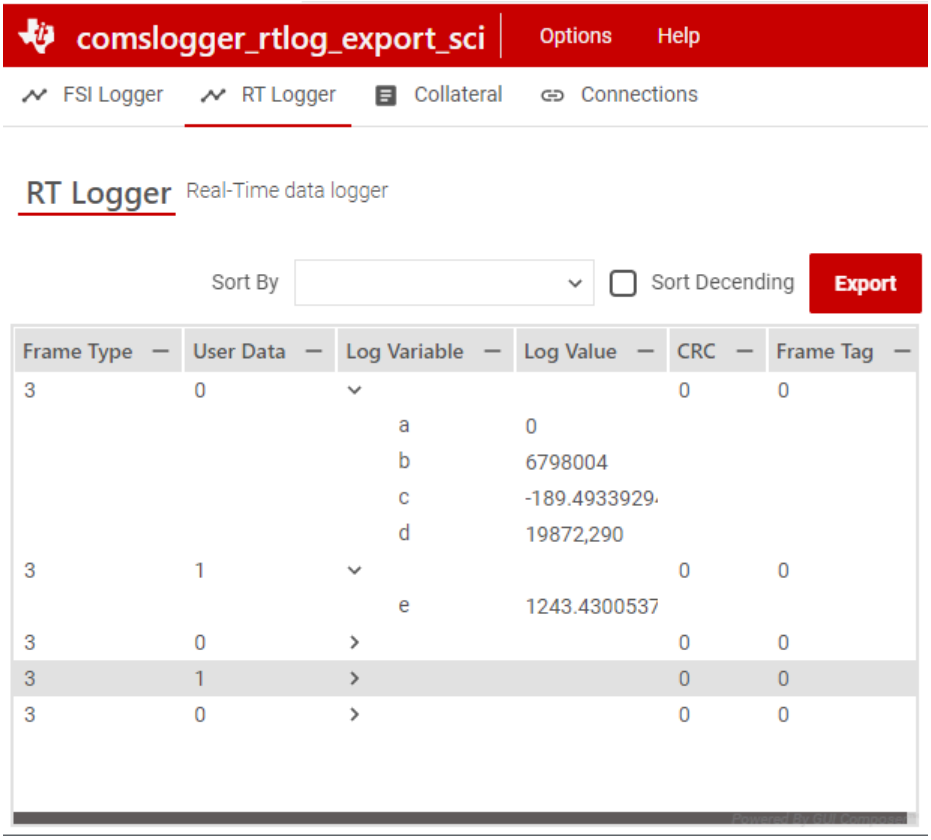


図 8-7. PC の GUI で高速時間ログデータを表示

9 転送例の概要

このドキュメントで前述した手引きでは、データロギングサポートを既存の CCS プロジェクトに追加する方法を説明しました。さらに、マイコン制御センターの各機能の C2000 サンプルコードは、以下のパスの C2000Ware SDK で参照できます。*C2000Ware_VERSION#/driverlib/[DEVICE_GPN]/examples/[CORE_IF_MULTICORE]/transfer/*. 各例の説明と、エクスポートデバイスおよびブリッジデバイスでこれらの例を一緒に使用する方法を以下に示します。

プライマリデバイスの例	ブリッジデバイスの例	説明	ハードウェア設定
transfer_customlog_export_usb	該当なし	• *機能*:アプリケーションロギング • *プライマリ通信プロトコル*:USB • *プライマリパッケージフォーマット*:JSON • 簡単なテキストベースのメッセージを 1 秒ごとに GUI に記録します (基本的には USB を使用した printf)。	設定 #1
transfer_customlog_export_sci	(1)	• *機能*:アプリケーションロギング • *プライマリ通信プロトコル*:UART • *プライマリパッケージフォーマット*:JSON • 簡単なテキストベースのメッセージを 1 秒ごとに GUI に記録します (基本的には UART を使用した printf)。	設定 #2
transfer_customlog_export_sci_buffer	(1)	• *機能*:アプリケーションロギング • *プライマリ通信プロトコル*:UART • *プライマリパッケージフォーマット*:JSON • 個別のプロセスを使用して、1 秒ごとに簡単なテキストベースのメッセージを GUI にバッファしてログに記録します (基本的には、タイミングの狭いシステムでは UART を使用する printf)。	設定 #2
transfer_customlog_export_sci_logArrays	(1)	• *機能*:アプリケーションロギング • *プライマリ通信プロトコル*:UART • *プライマリパッケージフォーマット*:JSON • テキストベースのデータ配列と数値データ配列を 1 秒ごとに GUI に記録します。	設定 #2
transfer_raw_fsi_tx	transfer_comslogger_export_sci	• *機能*:通信ロガー • *プライマリ通信プロトコル*:FSI • *プライマリパッケージフォーマット*:なし (FSI フレームのみ) • FSI を使用して未加工データを記録 - すべての未加工 FSI データを抽出して SCI を介し送信し GUI に表示	設定 #4
transfer_customlog_export_fsi	transfer_bridge_sci	• *機能*:アプリケーションロガーと転送ブリッジ • *プライマリ通信プロトコル*:FSI • *プライマリパッケージフォーマット*:JSON • FSI (JSON 形式) を使用してデータを記録 - JSON 形式の FSI データを受信して SCI を介しペイロードを送信し GUI に表示	設定 #4

プライマリデバイスの例	ブリッジデバイスの例	説明	ハードウェア設定
transfer_rtlog	transfer_comslogger_rtlog_export_sci	• *機能*:リアルタイムロガーおよび通信ロガー • *プライマリ通信プロトコル*:FSI • *プライマリパッケージフォーマット*:カスタム (フレーム構造による) • カスタムバイトパッキング形式の FSI を使用してリアルタイムデータを記録 - 受信した FSI データ要素を抽出して SCI で送信し GUI にインテリジェントに表示します。	設定 #4

(1) この例では、UART-USB ブリッジデバイスが必要です。すべての C2000 LaunchPAD と controlCARD には、このブリッジデバイスがボードに組み込まれており、PC 上でデータを表示するためにハードウェアを追加する必要はありません。ロギングデバイスにカスタムボードを使用する場合は、外部の UART-USB ブリッジデバイスが必要です。

10 まとめ

マイコン制御センターは、ログデータをキャプチャおよび分析するためのカスタム GUI アプリケーションを使用して、データロギングを実装します。このツールで使用できる機能は、任意のプロジェクトに追加して、さまざまな使用事例を実装できます。マイコン制御センターツールを使用すると、自動生成ターゲットコードサポート、直観的なグラフィカルユーザーインターフェイス GUI を使用、および特定の C2000 デバイスで利用可能な通信周辺装置を効率的に使用することにより、C2000 デバイスからデータをシームレスに収集できます。

11 参考資料

- テキサス インスツルメンツ、『MCU Signal Sight Tool Developer's Guide (マイコン信号観測ツール開発者ガイド)』、アプリケーションノート
- テキサス インスツルメンツ、『C2000 リアルタイムマイクロコントローラ周辺装置』、ユーザーガイド
- テキサス インスツルメンツ、『C2000 Sysconfig』、アプリケーションノート
- テキサス インスツルメンツ、『DLT Developer's Guide with Tooling (DLT 開発者ガイドとツール)』、アプリケーションノート

重要なお知らせと免責事項

テキサス・インスツルメンツは、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、テキサス・インスツルメンツ製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した テキサス・インスツルメンツ製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとします。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている テキサス・インスツルメンツ製品を使用するアプリケーションの開発の目的でのみ、テキサス・インスツルメンツはその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。テキサス・インスツルメンツや第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、テキサス・インスツルメンツおよびその代理人を完全に補償するものとし、テキサス・インスツルメンツは一切の責任を拒否します。

テキサス・インスツルメンツの製品は、テキサス・インスツルメンツの販売条件、または ti.com やかかる テキサス・インスツルメンツ製品の関連資料などのいずれかを通じて提供する適用可能な条項の下で提供されています。テキサス・インスツルメンツがこれらのリソースを提供することは、適用されるテキサス・インスツルメンツの保証または他の保証の放棄の拡大や変更を意味するものではありません。

お客様がいかなる追加条項または代替条項を提案した場合でも、テキサス・インスツルメンツはそれらに異議を唱え、拒否します。

郵送先住所:Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

重要なお知らせと免責事項

テキサス・インスツルメンツは、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、 テキサス・インスツルメンツ製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した テキサス・インスツルメンツ製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとします。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている テキサス・インスツルメンツ製品を使用するアプリケーションの開発の目的でのみ、 テキサス・インスツルメンツはその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。 テキサス・インスツルメンツや第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、 テキサス・インスツルメンツおよびその代理人を完全に補償するものとし、 テキサス・インスツルメンツは一切の責任を拒否します。

テキサス・インスツルメンツの製品は、 テキサス・インスツルメンツの販売条件、または ti.com やかかる テキサス・インスツルメンツ製品の関連資料などのいずれかを通じて提供する適用可能な条項の下で提供されています。 テキサス・インスツルメンツがこれらのリソースを提供することは、適用される テキサス・インスツルメンツの保証または他の保証の放棄の拡大や変更を意味するものではありません。

お客様がいかなる追加条項または代替条項を提案した場合でも、 テキサス・インスツルメンツはそれらに異議を唱え、拒否します。

郵送先住所：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265