ADK-2130mPCIe Windows10

テクニカル・マニュアル

株式会社ナセル

Introduction [はじめに]

Holt のフルサイズ F2 Mini PCI カードのリファレンス・デザインは、1 つの Mini PCI カードにトランスが統合された、1 つ、または、2 つの Holt HI-2130 MIL-STD-1553 マルチチャンネル・ターミナルを備えています。このカードは、Windows10 または、Linux を搭載した PC または、シングル・ボード・コンピュータで動作するように設計されています。デモ・ソフトウェアは、Holt API ライブラリ関数を使用して、ホスト・プログラミングを大幅に簡略化する抽象化レイヤーを提供します。コンソール・メニューは、コマンドを実行する Windows コマンドシェル・ウィンドウに表示されます。
このテクニカル・マニュアルでは、ハードウェア、ソフトウェアおよび、Holt Flash Drive (FD) ファイルをユーザーの PC に転送する方法について説明します。カードの概要、アプリケーション開発キット (ADK) の内容、および、Holt ブータブル・フラッシュ・ドライブを使用してデモ・ソフトウェアを実行する方法については、クイック・スタート・ガイド (QSG-2130mPCIe_Win10) を参照してください。ソフトウェア開発に使用する Holt ソフトウェアと Microsoft Visual Studio 2019 IDE ツールをユーザーの PC にダウンロード／インストールするには、本ガイドの説明を参考にしてください。

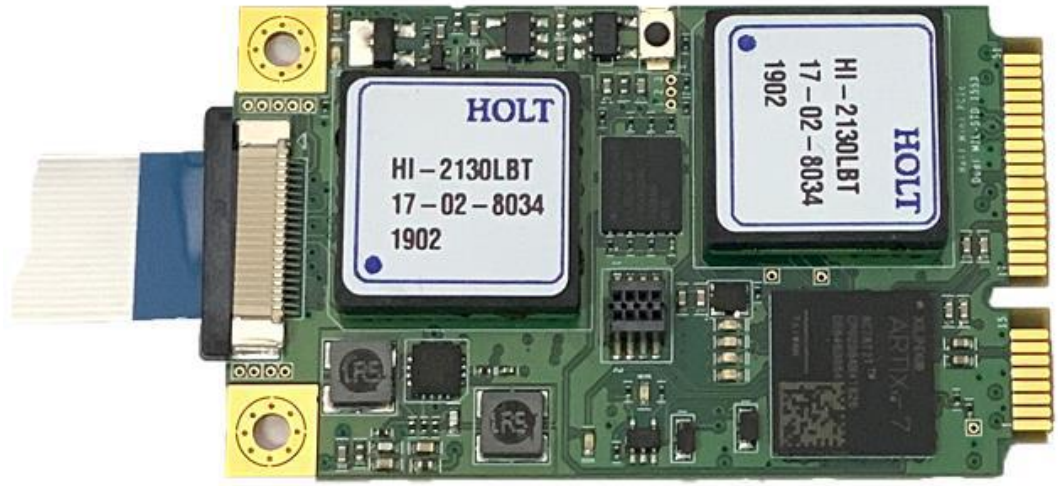


図 1　Mini PCIe カード EV-2130mPCIe-2F

Features [特徴]

- Mini PCIe F2 フルサイズ、PCIe Gen2 シングル・レーン
- シングル・チャンネル ADK-2130mPCIe-1F または、デュアル・チャンネル ADK-2130mPCIe-2F
- Mini Card Electromechanical Specification Rev.1.2
- 実績のある HI-2130 を使用した独立した 2×二重冗長 MIL-STD-1553 チャンネル
- BC、2×RT、MT の各チャンネル
- トランス結合 MIL-STD-1553 インターフェイス
- Holt API ライブラリ 3.5 のサポート
- Windows 10 OS サポート
- 動作温度：-40℃～+85℃
- カスタマイズ可能な FPGA
- BC、RT、RT2、MT の Demo ソフトウェア

PC requirements [PC 要件]

OS :	Windows10 (Visual Studio 2019 Pro または、Community エディション)
システム RAM :	VS2019 では 8GB 推奨
ハード・ドライブ容量 :	VS2019 ロジェクトでは最小 10GB
Mini PCI スロット :	スタンダード・フルサイズ F2
USB 2.0 または、3.0 ポート :	Holt Flash Drive ファイルの転送用

Mini PCIe board block diagram [Mini PCI ボード・ブロック図]

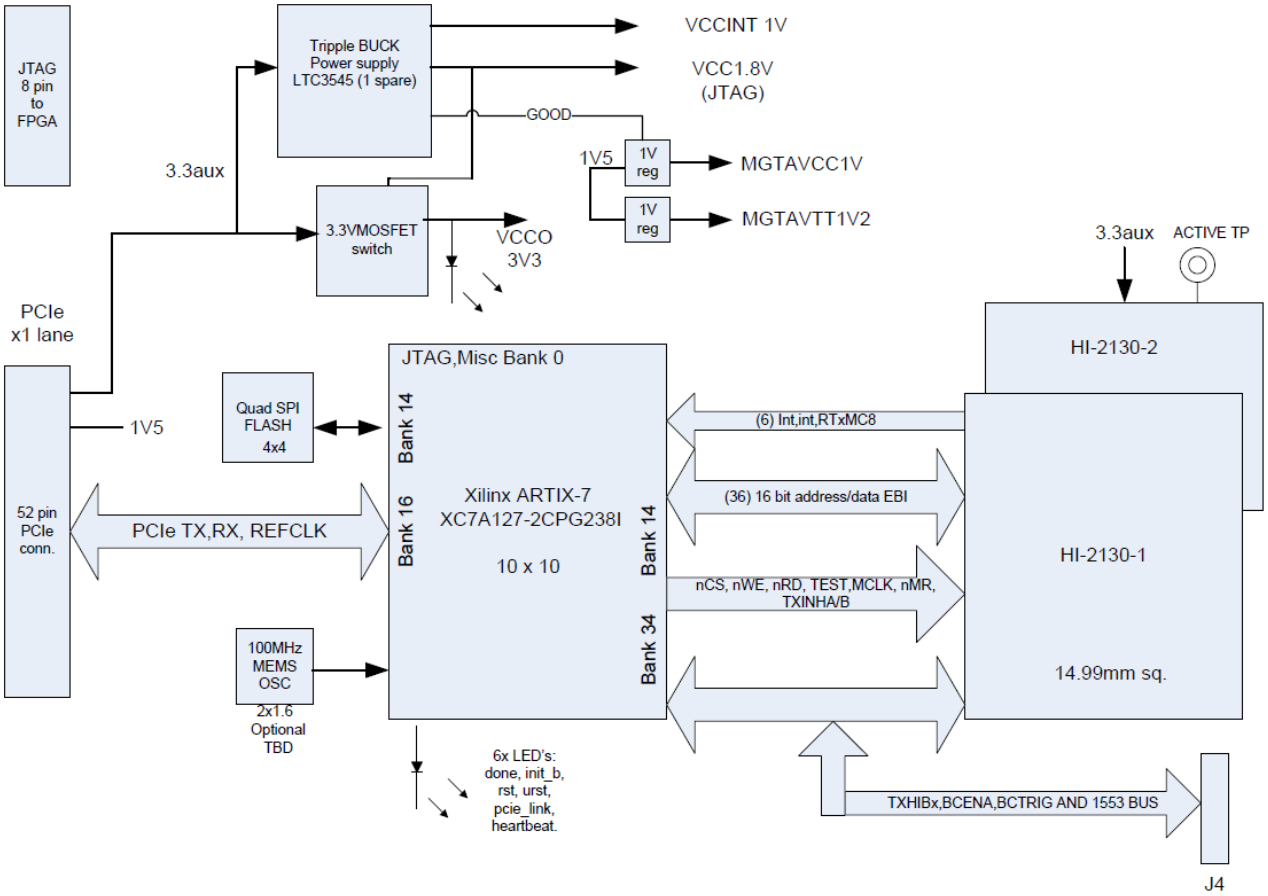


図 2　Mini PCIe ボード・ブロック図

Hardware Description［ハードウェアの説明］

Holt Mini PCIe カードは、PCI Express Mini カード電気機械仕様（下記参照）に準拠した 50.95mm×30mm のフォームファクタに、2 つの Holt HI-2130 二重冗長マルチターミナルを搭載しています。Xilinx XC7A12T FPGA が、2 つの HI-2130 と PCIe Gen2 バス間のインターフェイスとして機能します。ホストは、電源投入後に PCIe リンクが確立されると、内蔵の PCIe ドライバを使用してカードと通信します。カードはダイナミック・クロッキングを使用した PCIe 低電力モードをサポートしていません。pin-7 CLKREQ#は、カード上で接地されています。

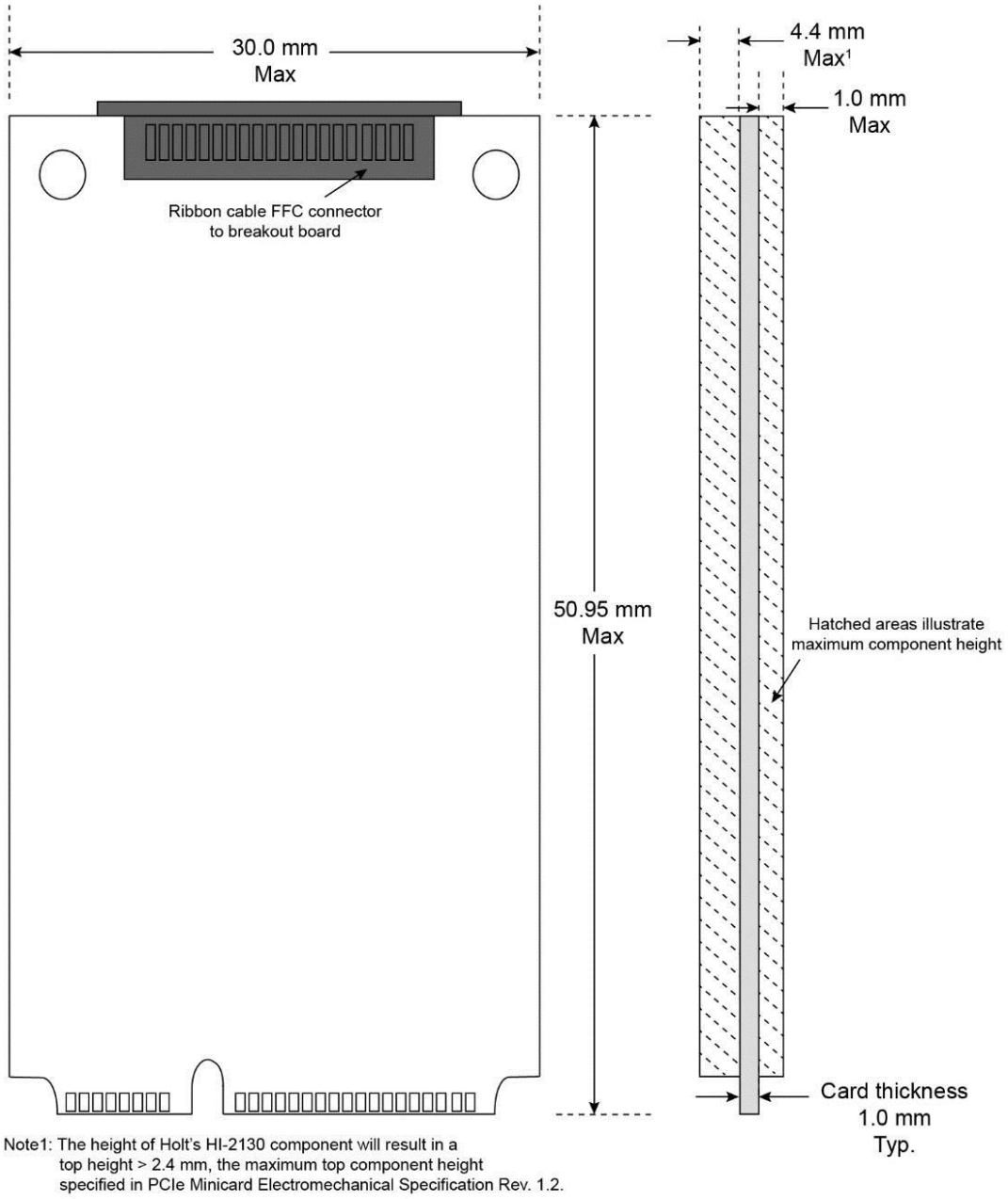


図 3　Mini PCIe カード・フォームファクタ

Break Out Board [ブレイクアウト・ボード]

ブレイクアウト・ボードは、リボン・ケーブルを介して PCIe カードに接続し、両方のチャンネル（Dev0 と Dev1)からの BusA と BusB の両方の接続をブレーク･アウトするために使用されます。Transmit Inhibits 入力は High にプルアップされており、HI-2130 デバイスに Low を提供し、ディフォルトで 1553 Bus 送信を有効にします。表 8 は、インターコネクト・リボン J4 コネクタの信号と説明のリストです。ボード・コンポーネントの位置、コネクタ、IC、および、テスト・ポイントについては、ドキュメントの最後にある図 5 ボード・リファレンスを参照してください。2 つのブレイクアウト・ボードを使用できます。

シングル・チャンネル：mPCIe_breakout-**1F**

デュアル・チャンネル：mPCIe_breakout-**2F**

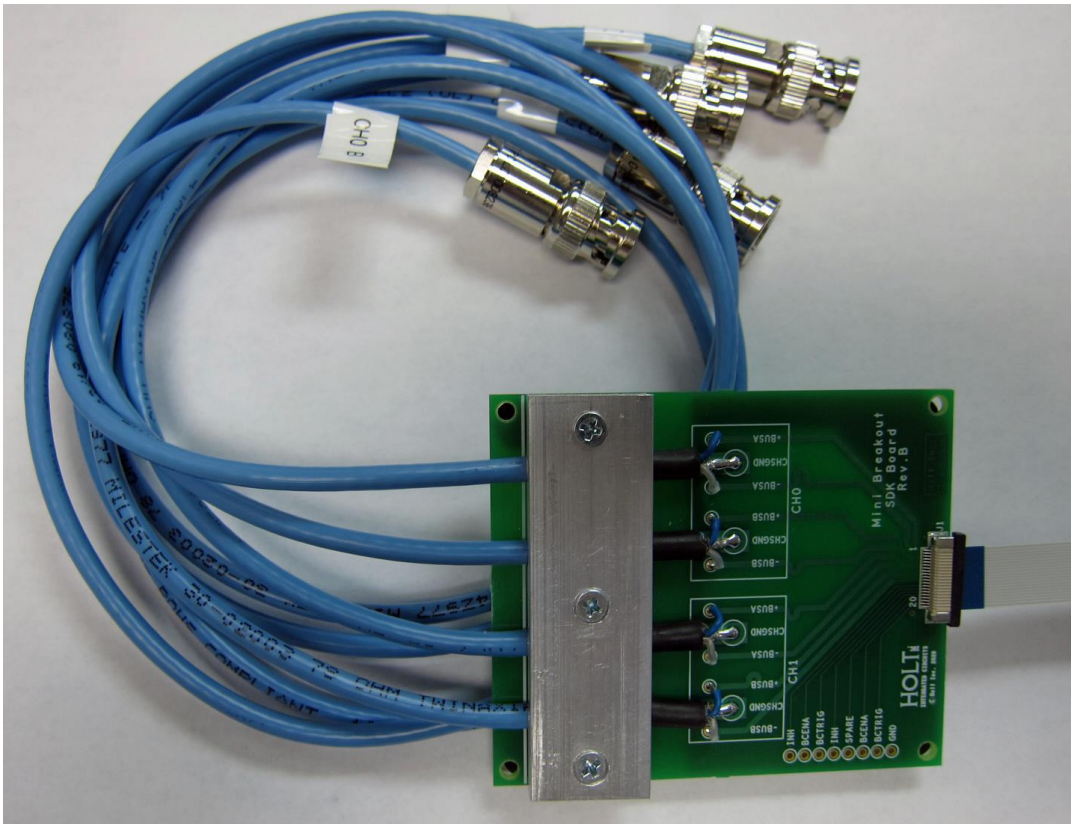


図 5　mPCIe_breakout-2F ブレイクアウト・ボード

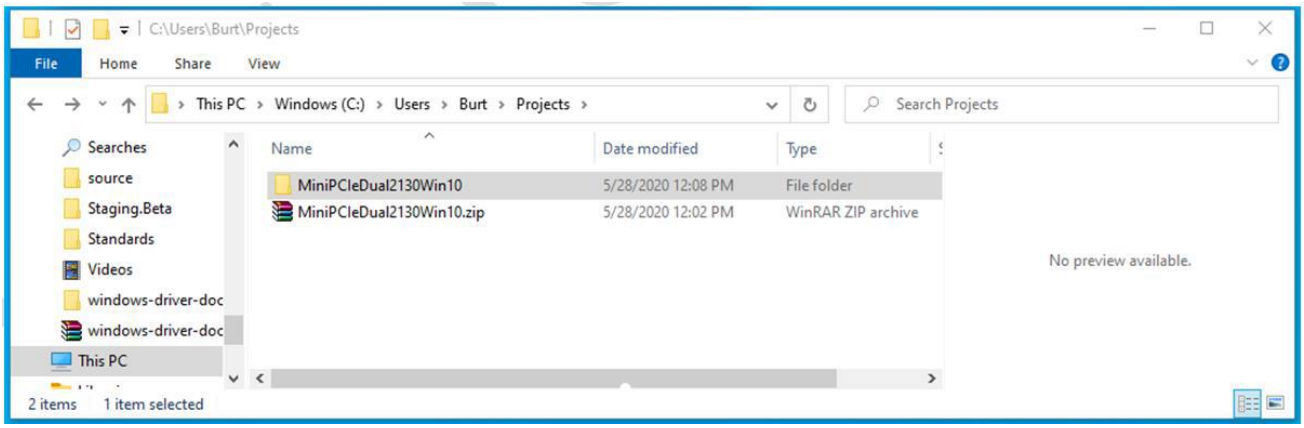
References Used in this Manual [このマニュアル中で使用される参照]

[RelNote]	Holt Mini PCIe Win10 Release note (ReleaseNotes.pdf)
[QSG]	Quick Start Guide (QSG-2130mPCIe_Win10.pdf)
[TM]	Technical Manual (AN-2130mPCIe_Win10.pdf) (本マニュアル)
[1F]	1CH ADK-2130mPCIe-1F ボード
[2F]	2CH ADK-2130mPCIe-2F ボード
[VS2019]	Microsoft Visual Studio 2019 Community または、Professional Edition

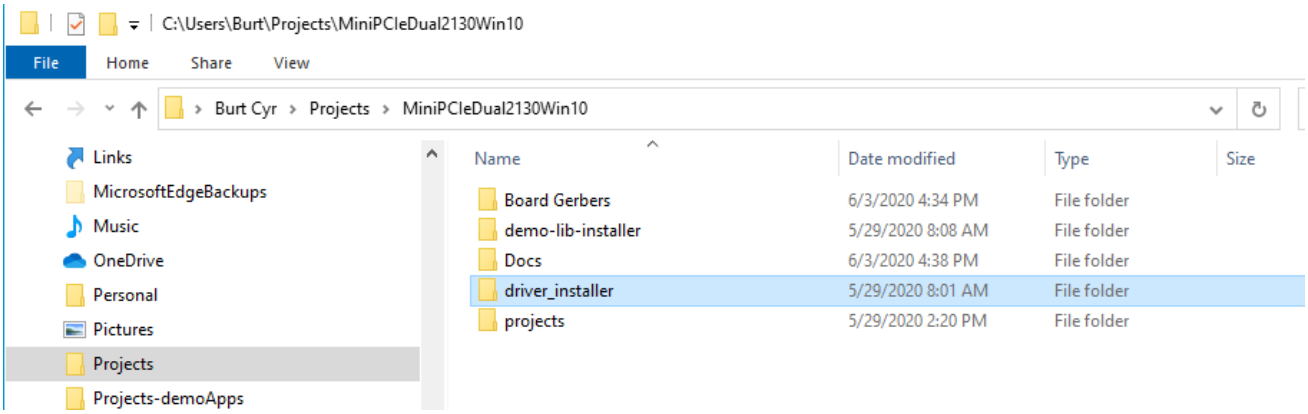
First Things First［重要なことから先に］

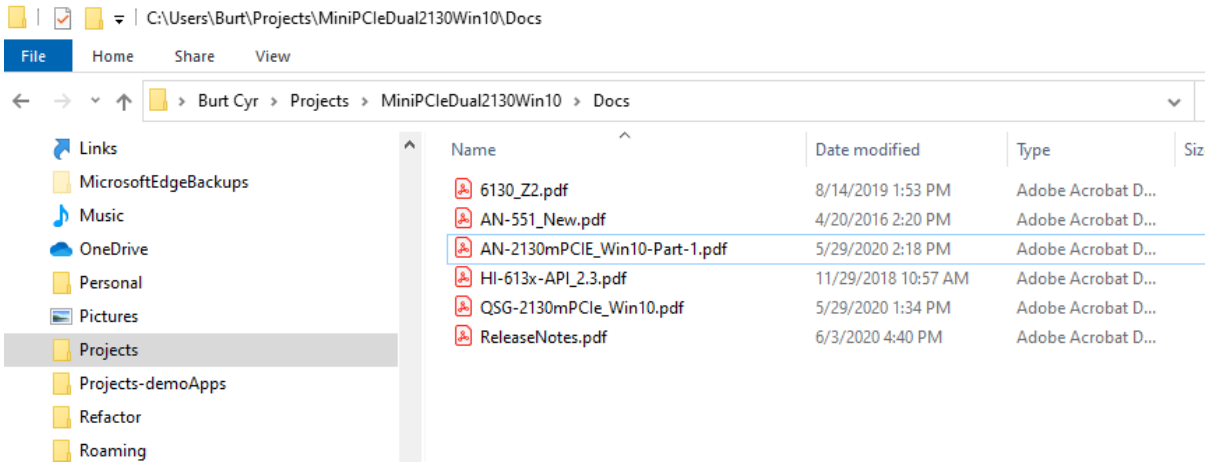
すべてのソフトウェアとドキュメントは、付属の USB フラッシュ・メモリに ZIP ファイルで提供されています。これは MiniPCIeDual2130Win10 という名前のディレクトリに解凍されます。本テクニカル・マニュアルで「トップレベル」という表現を使う場合、これは MiniPCIeDual2130Win10 ディレクトリの中を意味しています。このディストリビューションでは、トップレベルのディレクトリの中にある Projects フォルダに Visual Studio プロジェクトという名前のプロジェクトが含まれています。これらのプロジェクトは、読み書き可能なファイル・システムのどこにでも存在することができます。一般的に Projects は、「C:/Users/name/MiniPCIeDual2130Win10」のようなアクセスしやすいホームフォルダにあります。このプロジェクトを「My Documents」の下にインストールすると、一部のファイルが読み取り専用になり、プロジェクトのビルドがうまくいかなくなることがあります。

このディストリビューションは Win アーカイブ・ファイル（MiniPCIeDual2130Win10.zip）でリリースされており、解凍する必要があります。以下のトップレベルの階層が含まれています。



注記：ファイル名と日付が異なる場合があります。





本ドキュメントを読む前に、QSG-2130mPCIe_Win10 の資料をご確認いただくことをお勧めします。Holt デモ・プログラムを正常に実行するためには、PC が電源投入時にカードの PCIe インターフェイスを使用します。このセクションでは、ハードウェアのインストール手順、ドライバが Windows 10 で完全に機能することを確認するために、最初のテストを実行してください。

注意:Windows ドライバを更新するためこの手順を完了するには、システム管理者機能が必要です。

Holt IC から Beta Release 1.0 をインストールされたお客様へのご注意：このインストールを実行する前に、ベータ・リリースをアンインストールする手順については、QSG-2130mPCIe_Win10 のドキュメントを参照してください。

Mini PCIe Card Installation [Mini PCIe カード・インストール]

1. Windows 10 デバイスマネージャー・インターフェイスと Windows コマンドシェルを使用する必要があります。
2. Windows をシャットダウンするように指示してから（シャットダウンしたら）、システムの電源の電源を物理的に切るか、システムの電源への電源コードを物理的に切り離します。その理由は、システムによっては、シャットダウン時にも様々なコンポーネントに電力が残っているからです。内部ハードウェアを挿入したり取り外したりする場合は、シャットダウン後に電源が切断されていることを確認することを常にお勧めします。
3. 標準的な ESD 安全取扱方法（例：アーススストラップ、静電気制限作業領域など）を使用して、Holt Mini PCIe カードを設置してください。
4. EV-2130mPCIe-1F/2F ボードをシステムのマザーボードの Mini PCIe スロットに挿入します。Mini PCIe-to-PCIe X1 アダプタボードを使用している場合、EV-2130mPCIe-1F/2F カードをアダプタボードにマウントし、PCIe X1 スロットに挿入します。小さなリボン・ケーブルを Mini Card J4 コネクタに、もう一方の端を EV-2130mPCIe-1F/2F カードのブレイクアウト・ボードに慎重に挿入します。ケーブルのピッチは細かく、コネクタのプラスチックファスナーはデリケートなので注意が必要です。コネクタを傷つけないように注意してください。
5. システムの電源を入れ、通常通りログインしてください。
6. PC が起動します。コンピュータの電源を入れた直後に、緑色の PCIe リンク LED8 が点灯し、ハートビート LED9 が[1F]または、[2F]カードの上で点滅しているはずです。

7. LED 8 が点灯すると、コンピュータの起動シーケンスがカードの PCIe リンクを検出したことを示します。
8. ただし、Mini PCIe Windows Driver がインストールされるまでは、コンピュータが Holt デバイスを正しく検出できない場合があります。

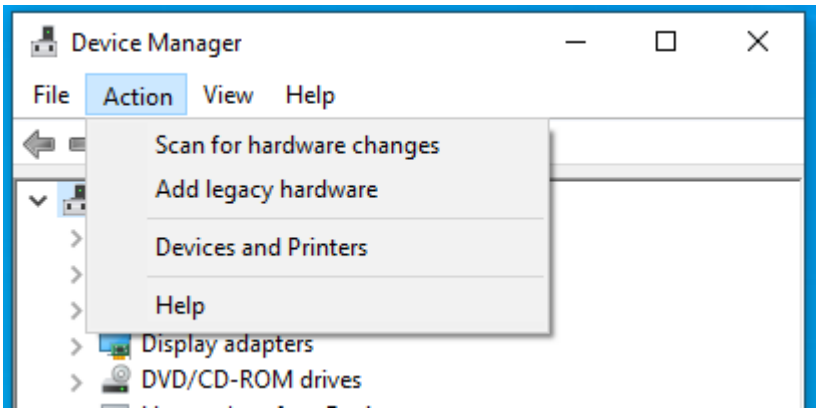
Software Installation [ソフトウェアのインストール]

今回のリリースには、ドライバソフトのインストーラと Demo+Lib の 2 つのソフトのインストーラが含まれています。最初にドライバのインストールを行います。

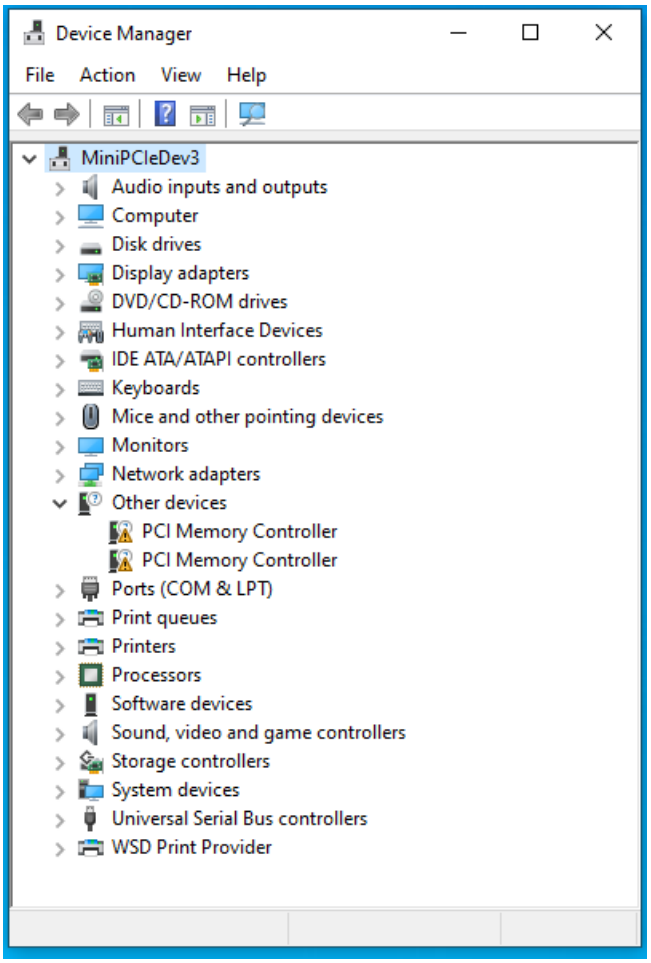
Driver Installation [ドライバのインストール]

リリースキットの最上位のディレクトリには、「*driver_installer*」というディレクトリがあり、そこには *holt_pcie.exe* ドライバのインストーラが含まれています。現時点では、Holt はドライバのソース・コードを公開していませんが、この構造はビルド手順と同じです。このステップでは、Holt の Windows ドライバをインストールします。しかし、まず、Windows が新しいハードウェアを検出することを確認しましょう。

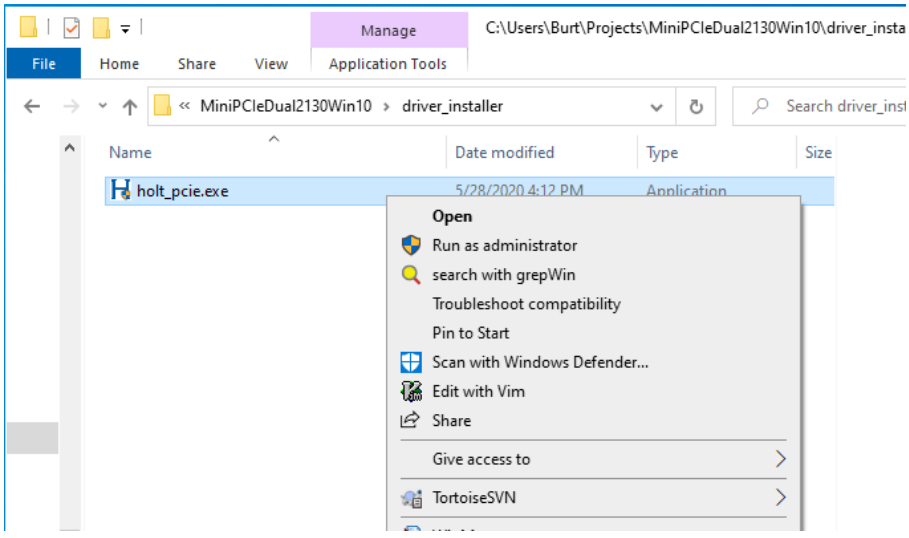
1. Windows のコントロールパネルを起動し、デバイス・マネージャをクリックします。[アクション] タブで、[ハードウェアの変更をスキャン]を選択します。



2. Mini PCIe カードが 2 枚インストールされている場合、以下のように表示されます。通常は「***その他のデバイス***」の下に表示されます。

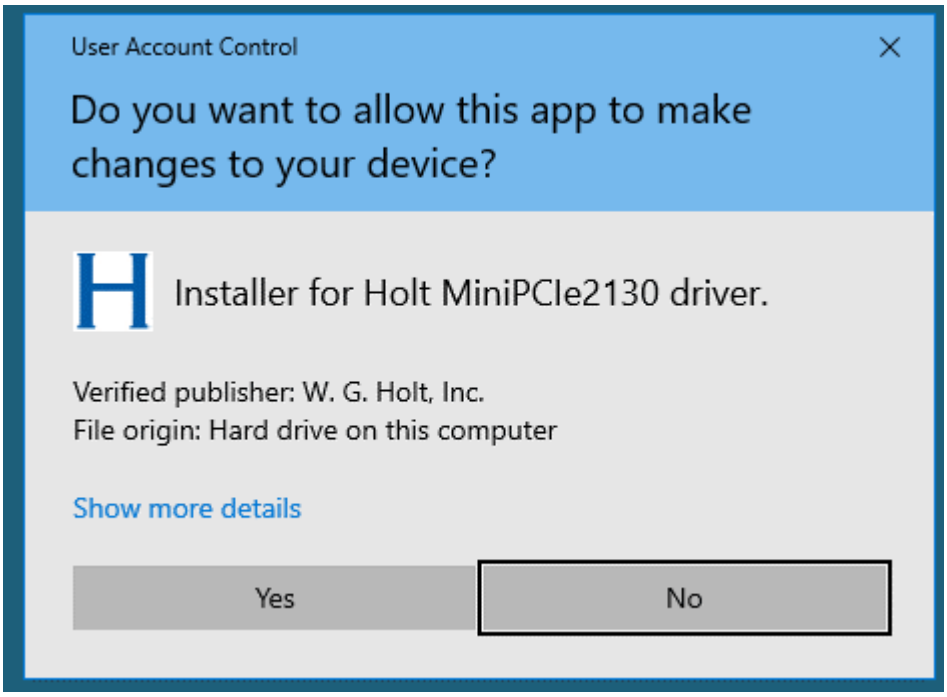


3. エクスプローラ・ウィンドウを開いて *driver_installer* に移動します。

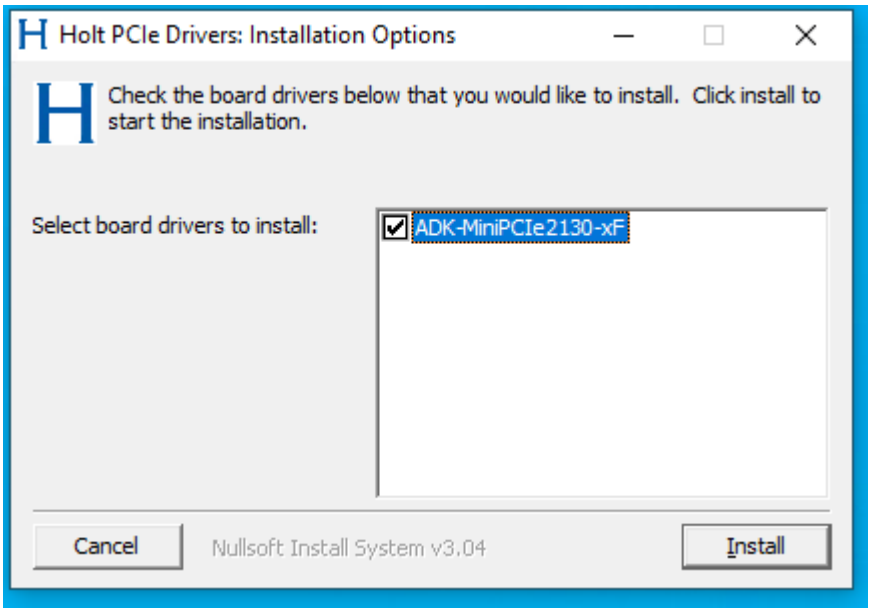


holt_pcie.exe を右クリックし、[***管理者として実行***] を選択します。

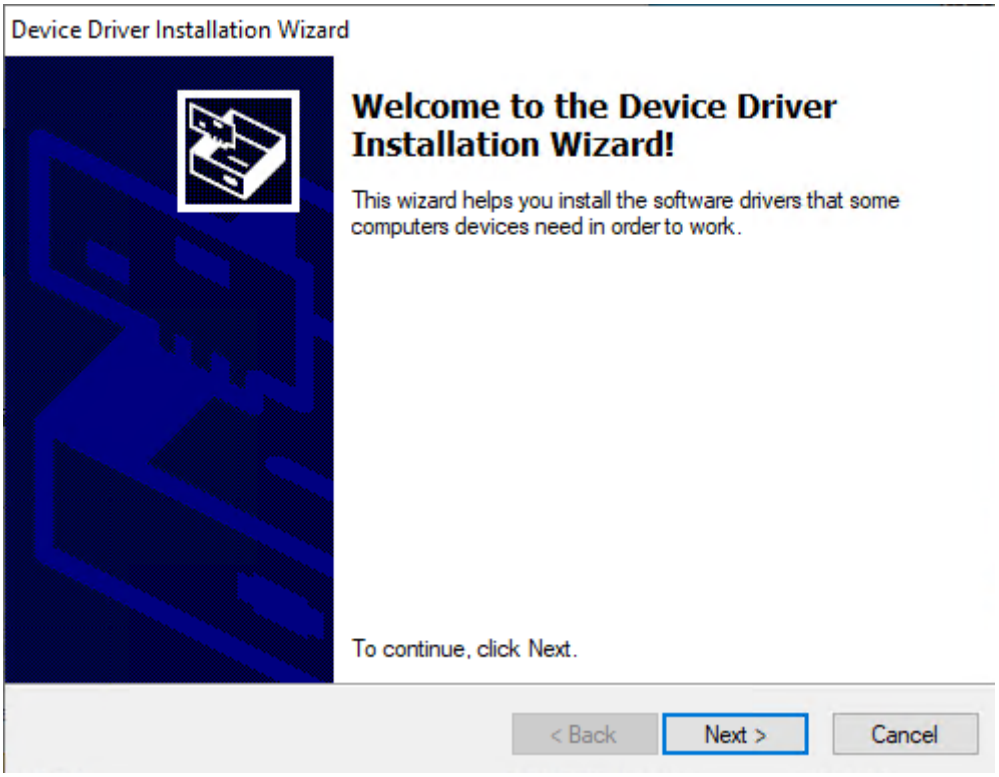
ポップアップ・ダイアログが表示されますので、「Yes」を選択して実行ファイルがシステムに変更を加えることを許可します。

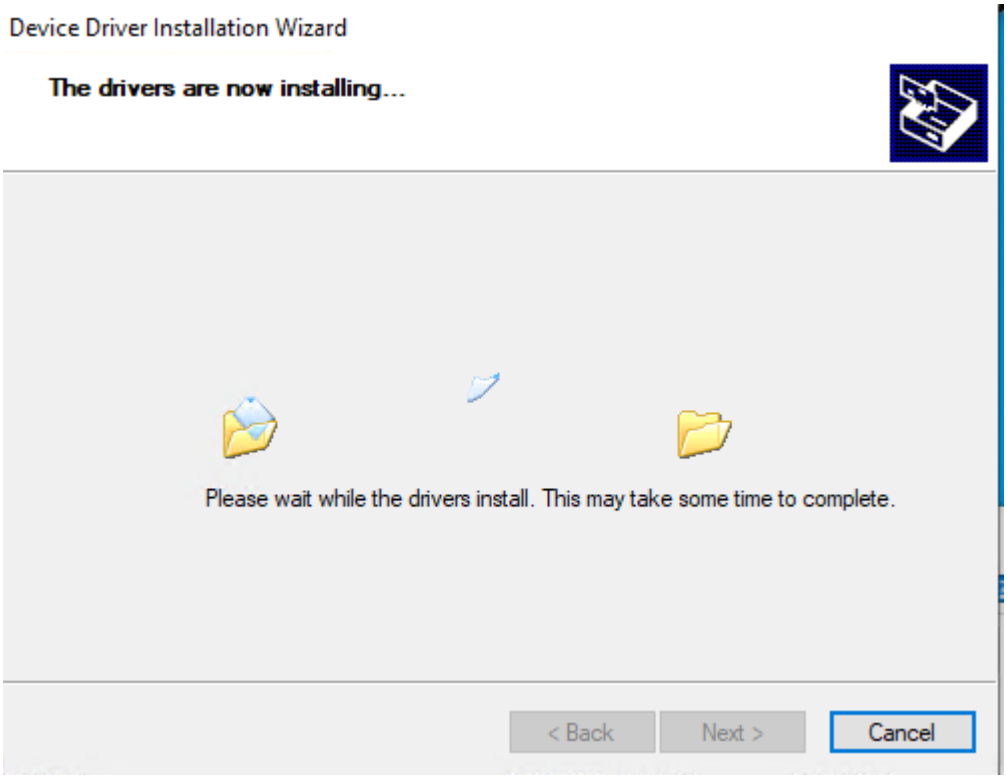


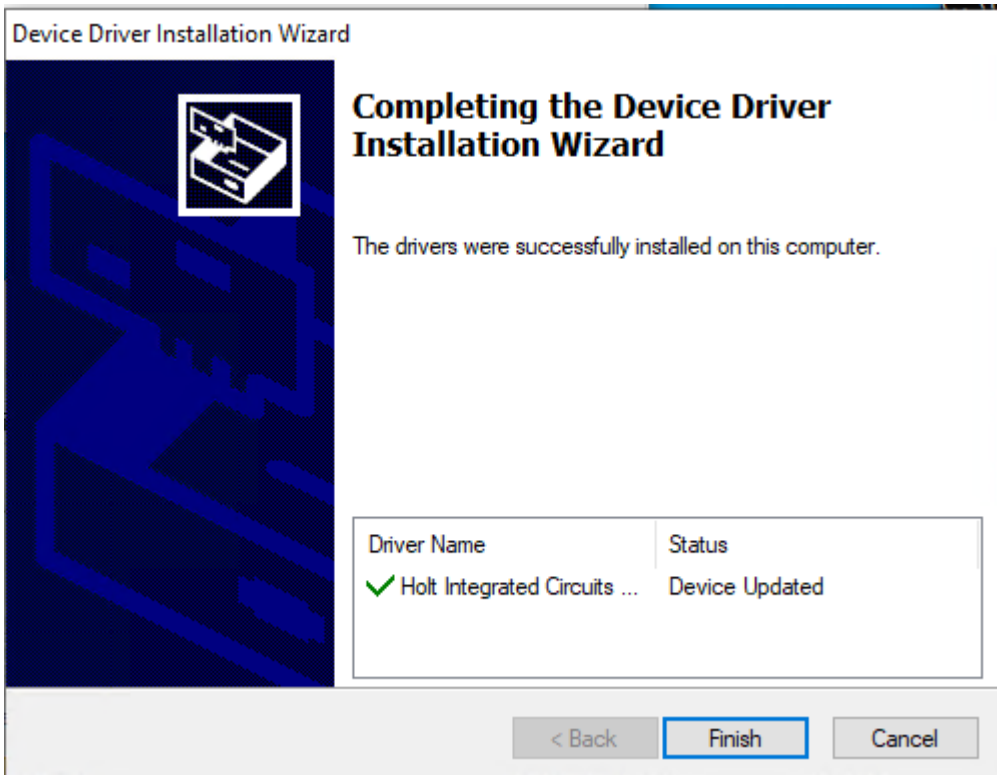
別のポップアップが表示されます。「Install」を選択します。



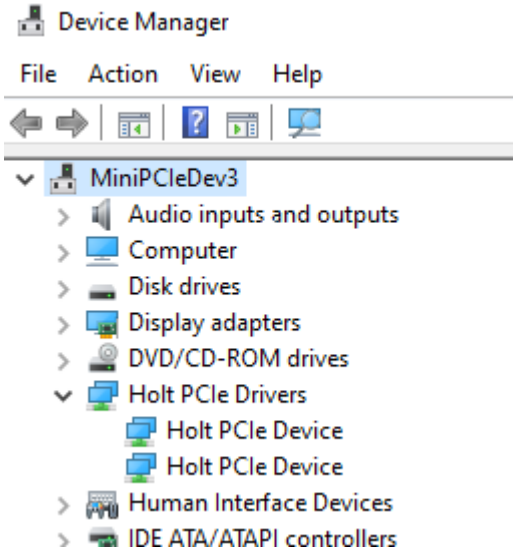
4. Windows ドライバのインストーラがポップアップ表示されます。「Next」をクリック







5. 「Finish」をクリックして、デバイス・マネージャがその他の PCIe ドライバを Holt のものに置き換えたことを確認してください。



6. 以前に複数の Mini PCIe カードがインストールされていた場合、同じ手順で他のドライバを更新してください（この例では、2 枚のカードがインストールされていることを示しています）。

7. Holt Mini PCIe ドライバがインストールされていることを確認します。
管理者権限でコマンドシェルを開きます。dir をこのデモのディストリビューションを含むトップレベルのフォルダに変更します。トップレベル/プロジェクト/ドライバテストを選択し、x1553_find.exe を実行します。見つかったデバイスの数を正確に報告して終了します。2 枚のカードシステムの場合は、以下のようになります。

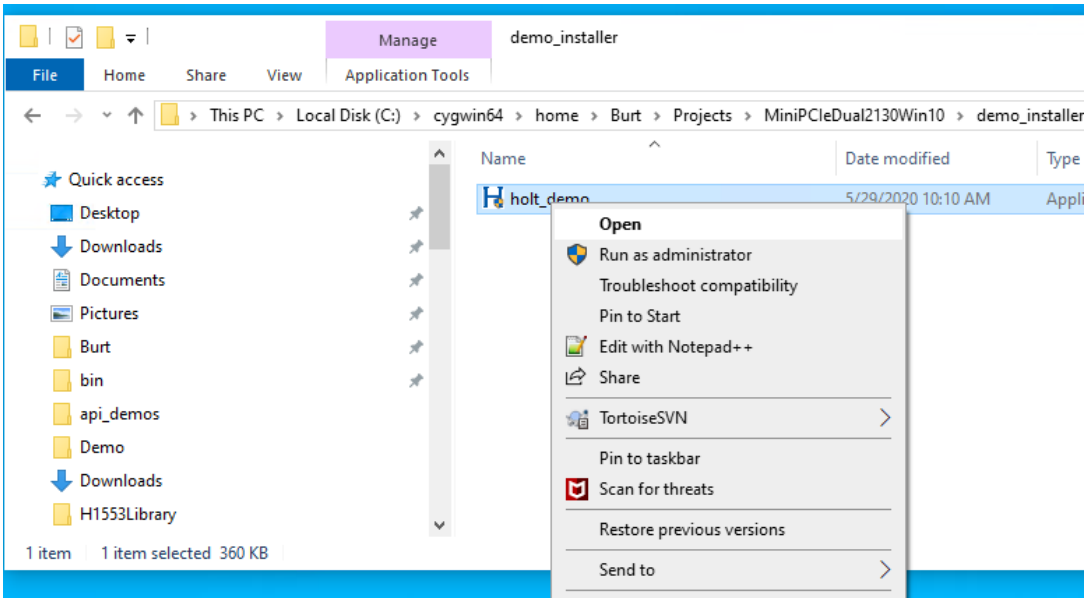
```
C:\Users\william\PcieDual2130VS\projects\driver_test>x1553_find
Devices found: 2
pcie[0]: \\?\pci#ven_10ee&dev_7011&subsys_000710ee&rev_00#0000000101000a3500#{3a3e515e-2c08-41dc-84ec-59a59966e2aa}
pcie[1]: \\?\pci#ven_10ee&dev_7011&subsys_000710ee&rev_00#6&334cb0bc&0&0000000101000a3500#{3a3e515e-2c08-41dc-84ec-59a59966e2aa}
```

もしそれが Holt PCIe カードを見つけられない場合は、読み込まれたドライバに何か問題があるということになります。
あなたの接続文字列は、文字列&rev_OO#something unique#(3a3e415...)のリビジョン・セクションが一意であるため、私たちの接続文字列と一致しないことを覚えておいてください。

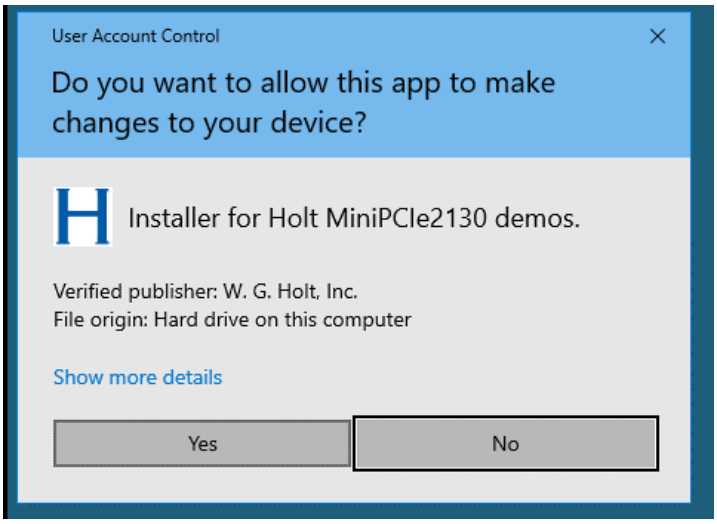
Installing the Demonstration and Holt API Library [デモンストレーションと Holt API ライブラリのインストール]

リリースキットのトップレベルのディレクトリには、*demo_installer* というディレクトリがあります。このステップでは、Holt Windows デモンストレーションとライブラリをインストールします。

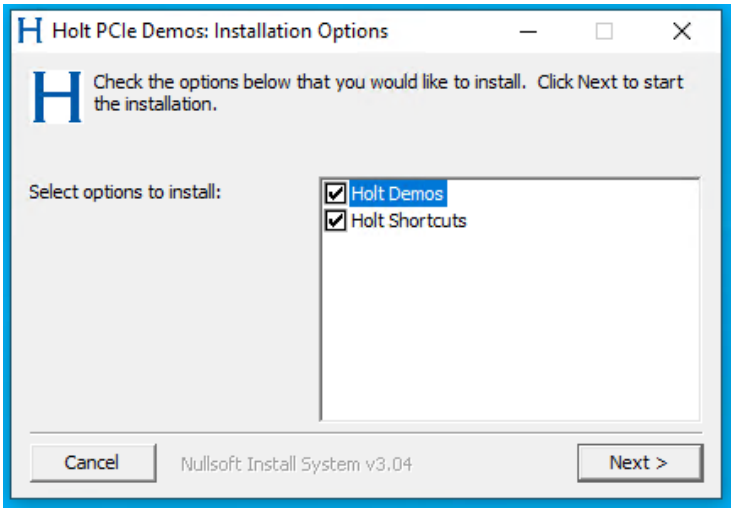
1. エクスプローラ・ウィンドウを開いて *driver_installer* に移動します。holt_demo.exe 実行ファイルを選択し、右クリックし***て管理者として実行***します。



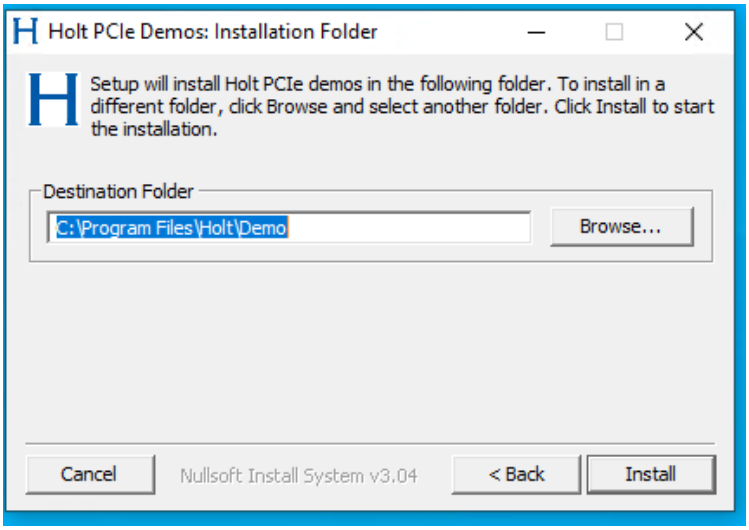
このダイアログで「Yes」をクリックします。



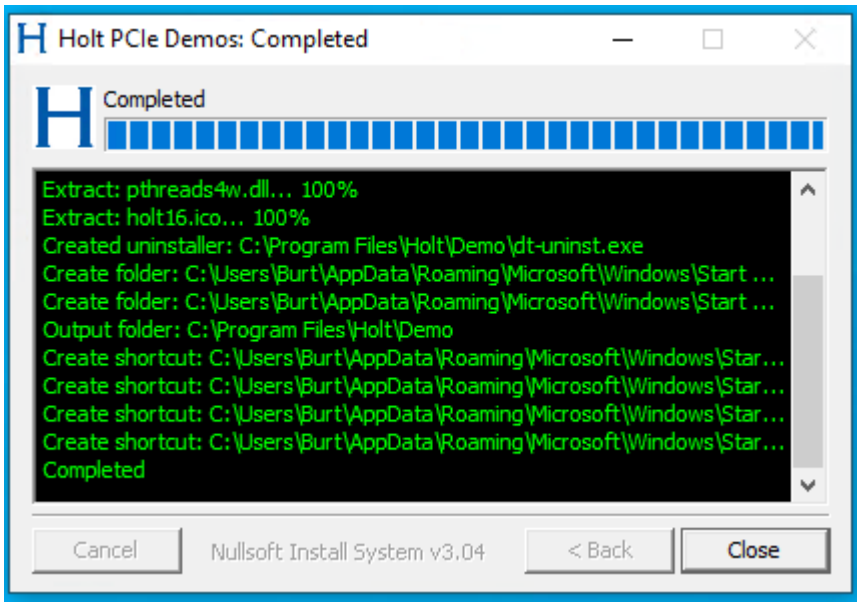
このダイアログで「Next」をクリックします。



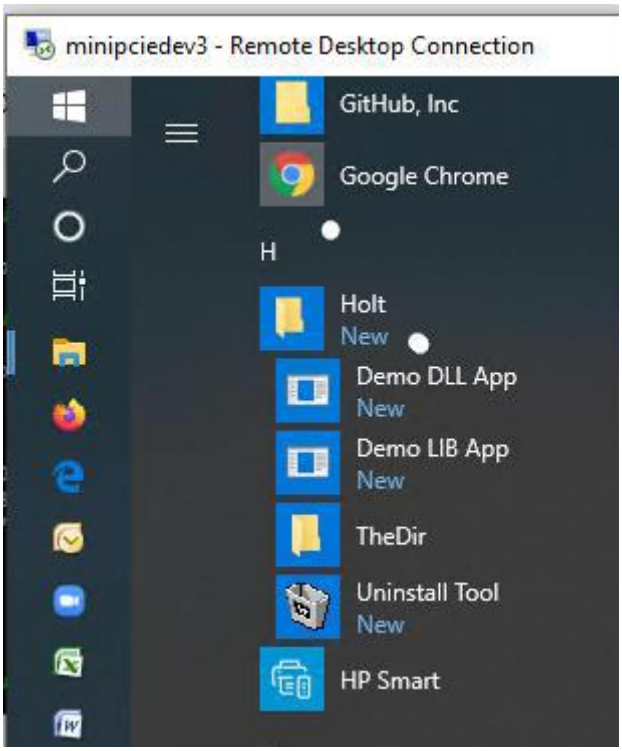
このダイアログで「Install」をクリックします。



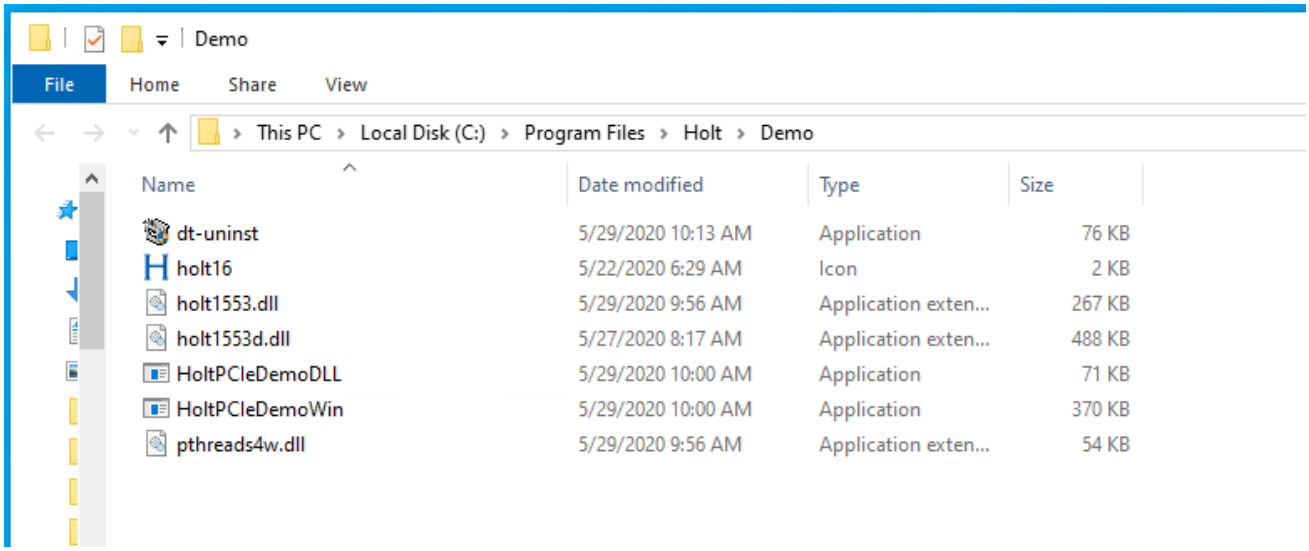
そして最後に「Close」。



2. Windows のスタートメニュー・ウィンドウを左クリックして、アプリケーション・トレイを表示します。H アプリケーションをブラウズして、Holt アプリケーションがインストールされていることを確認してください（アンインストーラーと一緒に）。



3. アイコンが表示されます。*TheDir* は、アプリケーションがインストールされた場所のエクスプローラ・ウィンドウを開きます。



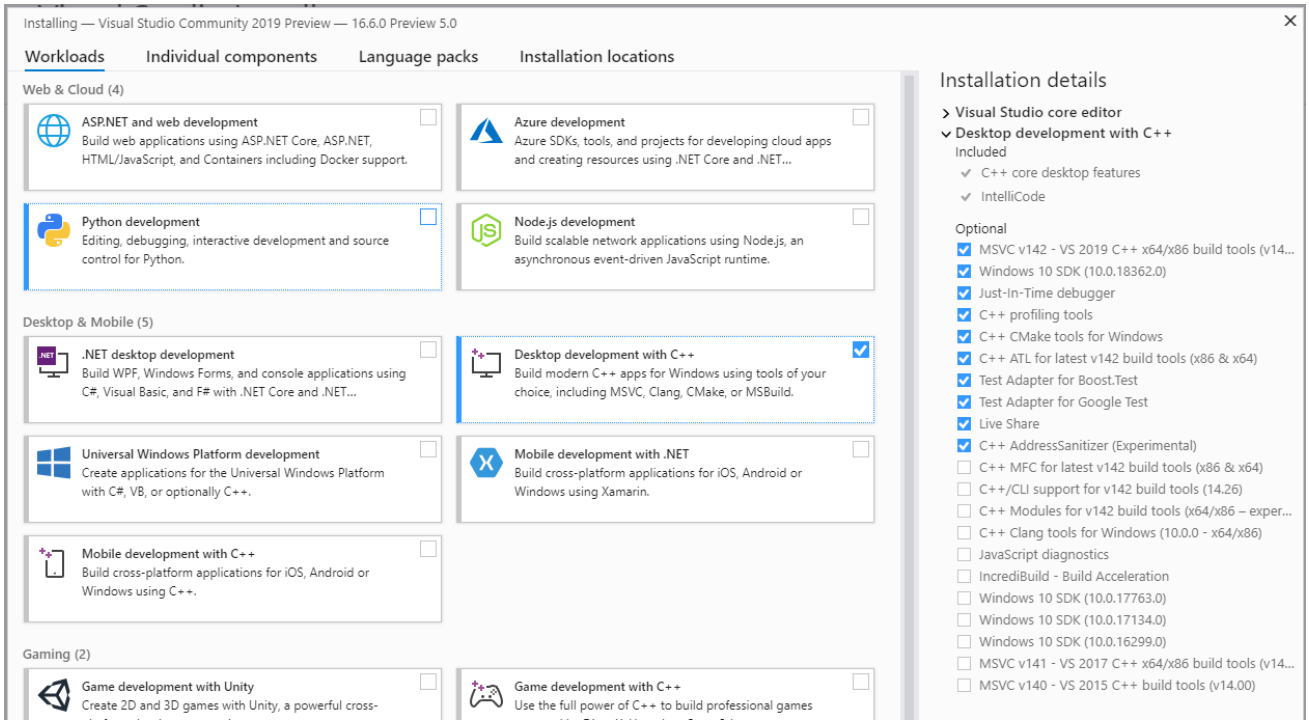
PC preparation for installing Visual Studio 2019 and Holt software [Visual Studio 2019 と Holt のソフトをインストールするための PC 準備]

Visual Studio 2019 (VS2019) Community Edition 統合開発環境は、以下から無料でご利用いただけます。Microsoft のもので、このプロジェクトで使用されています。VS2019 Community 版または、Professional 版が既にインストールされている場合、この VS2019 のインストール手順を確認して、適切なワークロードと個別コンポーネント（VS2019 でプロビジョニングされている）が適切にインストールされているかどうかを確認する必要があります。

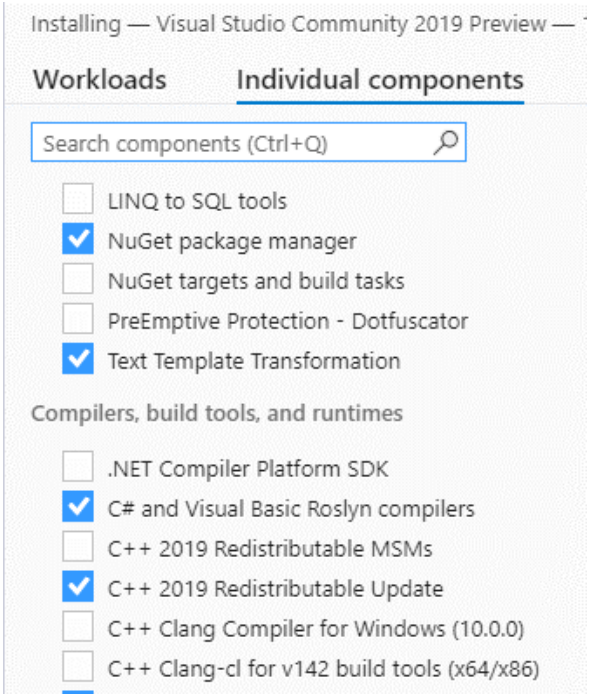
Holt Mini PCIe デモを実行するには、Windows SDK の一部である C++ Windows 再頒布可能ファイル（Visual Studio に含まれています）が必要ですが、再頒布可能ファイルは Visual Studio なしで入手できます。Holt では、VS2019 を含む道を進んできました。VS2019 が使用されていない場合は、デモを使用して、Windows の再配布可能ファイルをダウンロードするには、Microsoft から直接ダウンロードします。Visual Studio または Pro がすでにインストールされている場合は、まだ C++ワークロードでデスクトップ開発を有効にする必要があり、C++再頒布可能ファイルの担保があります。VS2019 をお持ちでない方は、こちらから Community 版をダウンロードしてください。

https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community&rel=16

C++ワークロードを使用したデスクトップ開発をインストールする場合。以下がチェックされていることを確認してください：



Visual Studio インストーラでインストールをクリックする前に、[Individual components]ドロップダウン・タブを選択して、次の要素も選択します。



および、

最後に

インストールをクリックします。インストール後、Holt の実行ファイルをビルドして実行できるようになるはずです。
Windows 再頒布可能プログラムの詳細については、こちらを参照してください。
https://docs.microsoft.com/en-us/visualstudio/releases/2019/redistribution

Building Methodology [ビルド方法]

Holt では、デモ・プロジェクトのビルド、編集、デバッグのための 3 つの主要な Visual Studio ソリューションを提供しています。Holt API ライブラリ・プロジェクトをビルドするには、API ソース・ファイルが必要ですが、標準リリースでは提供されていません。API ソース・ファイルは、署名済みの Holt ソフトウェア・ライセンス契約（SLA）を締結している場合に利用できます。デフォルト・リリースでは、必要なビルド済みのライブラリ・ファイルである h1553library.lib と holt1553.lib が提供されており、API をフルに利用してデモの 2 つのバージョンをビルド／リンクすることができます。

VS ソリューション名	目的	リンクされた API ライブラリ	ターゲット実行ファイル	Holt SLA 必要
HoltPCIeDemoWin.sln	ビルドデモと静的 API リンク	h1553library.lib	HoltPCIeDemoWin.exe	NO
HoltPCIeDemoDLL.sln	ビルドデモと API DLL リンク	holt1553.lib	HoltPCIeDemoDLL.exe	NO
		生成されたライブラリ		
h1553library.sln	API1553 静的ライブラリのビルド	h1553library.lib	N/A	YES
	API1553 DLL ライブラリのビルド	holt1553.DLL, holt1553.lib	N/A	YES

ビルド・タイプ：静的ライブラリ		
ソフトウェア バージョン	Release	Debug
SLA (API ソースを含む)	Lib .sln builds: h1553.lib pthreads4w.{lib,dll} Demo .sln builds: demoWin.exe w/no debug capabilities	Lib .sln builds: h1553d.lib pthreads4wd.{lib,dll} Demo .sln builds: demoWin.exe with complete debug capabilities
ADK	Binary dist only: h1553.lib No Lib build .sln Demo .sln builds: demoWin.exe w/no debug caps	Binary dist only: h1553.lib No Lib build .sln Demo .sln builds: demoWin.exe w/no h1553 lib debug caps

SLA には署名されたソフトウェア・ライセンス契約書が必要です。

ビルド・タイプ：DLL		
ソフトウェア バージョン	Release	Debug
SLA (API ソースを含む)	Lib .sln builds: h1553.{lib,dll} pthreads4w.{lib,dll} Demo .sln builds: demoDLL.exe w/no debug capabilities	Lib .sln builds: h1553d.{lib,dll} pthreads4wd.{lib,dll} Demo .sln builds: demoDLL.exe with complete debug capabilities
ADK	Binary dist only: h1553.lib No Lib build .sln Demo .sln builds: demoDLL.exe w/no debug capabilities	Binary dist only: h1553.lib No Lib build .sln Demo .sln builds: demoDLL.exe w/no h1553 lib debug capabilities

デモンストレーションは、2 つの方法のいずれかでビルドできます。静的にリンクされた Holt1553 ライブラリを使用する:holt1553.lib（リリース）または、holt1553d.lib（デバッグ）。

- 動的にロードされたライブラリの使用:holt1553.dll（リリース）または、holt1553d.dll（デバッグ）

デモンストレーションをビルドする際、ライブラリは最初に RELEASE ビルドまたは、DEBUG ビルドとしてビルドされます。RELEASE ビルドは実行に最適化されており、デバッグ・シンボルはありません。DEBUG ビルドは最適化されておらず、デバッグ・シンボルとデモ実行時のコンソール上の余分なログ表示が含まれています。

静的とダイナミックのどちらを使用するか？

DLL は少し複雑ですが、全体的には、リソースに制約のある環境で作業をしなければならない顧客にとっては非常に魅力的です。複数のアプリケーションが PCIe カードのリソースにアクセスしている場合（2 つのコマンドシェルが単一の DLL ベースの実行ファイルを介して同じドライバにアクセスしている場合のように）、ドライバを共有することで、全体的なメモリ消費量を削減することができます。すべてを駆動する実行ファイルを一つだけ使用する場合は、もちろん DLL オプションは必要ありませんし、静的ライブラリも同様に機能します。

ソリューション・ファイル

VS2019 では、「ソリューション」と「プロジェクト」ファイルを使用して、プロジェクトのセットを管理しています。このデモでは、3 つのソリューション・ファイルを使用しています：

HoltPCIeDemoWin.sln ： 静的な h1553library.lib（または、デバッグ用のバリアント）を使用して静的なデモだけをビルドします。

HoltPCIeDemoDLL.sln ： 動的な holt1553.dll を使用して動的にロードされたデモのみをビルドすると holt1553.lib（または、デバッグ用のバリアント）を使用しています。

PCIeDemo.sln ： 上記の両方をビルドします。

h1553library ソリューション：

H1553library.sln ： リリースまたは、デバッグ構成用の静的および、DLL ベースのライブラリと 3 つのプロジェクト・ファイルをビルドします。Holt DLL ベースの API 用、Holt 静的ライブラリ・ベースの API 用、および、pthreads4w 用の 3 つのプロジェクト・ファイルです。これら 2 つのプロジェクトをリビルドするためには、Holt API のソースファイル（SLA が必要）が必要です。

上記のソリューション・ファイルは、ビルドとクリーンプロセスを管理します。

プロジェクトのクリーン

holt1553 libs の保存先フォルダは、以下のいずれかです：

PCIeDual2130VS/H1553Library/dll（dll ベースの lib 用）または、

PCIeDual2130VS/H1553Library/lib（static lib 用）

Holt1553 ライブラリ・ソリューションがクリーンアップされてリビルドされると、これらの宛先ディレクトリはそれに応じて再配置されます。

プロジェクトのターゲットディレクトリ

デモの保存先フォルダは PCIeDual2130VS/Demo/build/x64/bin で、デモの静的 lib と dll の両方がここに配置されています。

また、このディレクトリには 3 つの dll ファイルがあります：

holt1553d.dll と holt1553.dll ： これらはソース・ディレクトリから手動でここにコピーされました。

PCIeDual2130VS/H1553Library/dll ： これらは、ライブラリがビルドされた後、ソース・ディレクトリから手動でここにコピーされました。

ucrtbased.dll ： これは外部ファイル（windows 再配布可能プログラムの一部）であり、VS2019 ワークフローではリビルドされていません。

Demo/build/x64/bin ディレクトリにある実行ファイルには以下のものが含まれています。

HoltPCIeDemoDLL.exe ： 動的にロードされたデモのバリアント

HoltPCIeDemoWIN.exe ： デモの静的にリンクされたバリアント

プロジェクト・リンク

静的リンクされた実行ファイル（HoltPCIeDemoWIN.exe）の最終リンク・ステップの間に、プロジェクト・ファイルは、プロジェクト・ディレクトリを横断して、PCIeDual2130VS/H1553Library/lib から静的ライブラリをソースにして、必要な holt1553.lib（リリース）または、holt1553d.lib（デバッグ）lib を取得します。これらのバイナリ・アーカイブは、ビルド・プロセスがソースから探し出すので、Demo サブディレクトリにコピーする必要はありません。

しかし、動的にリンクされた実行ファイル（HoltPCIeDemoDLL.exe）の最終リンク・ステップの間に、デモプロジェクト・ファイルは、最初に PCIeDual2130VS/H1553Library/dll サブディレクトリから必要なすべての DLL ファイルをコピーするように設定されています。DLL ライブラリのビルド・プロセスは、それらをデモの最終リンク・ディレクトリにコピーしません。したがって、当社のデモ方法がシステム

構築に使用される場合、最終リンクの前に DLL ファイルを同様にビルド領域にコピーすることを忘れないようにしてください。DLL 実行ファイルがリンクされているときは、最終リンクではなく、インクリメンタル・リンクであることに注意してください。OS は、プログラムの起動前の実行時に DLL 内で提供されている外部参照を動的にパッチする必要があります（後述の DLL の起動を参照）。それにもかかわらず、実行ファイルはそのランタイム DLL の対応するものに対してリンクされていなければなりません。そのため、.dll の成果物はビルド・ディレクトリに存在しなければなりません。

したがって、DLL アプリケーションをビルドするための正しい順序は以下の通りです。

1. DLL ベースの Demo をクリーンアップします（*.dll は Demo/build/x64/bin ディレクトリにまだ存在していることに注意してください。ucrtbased.dll ファイル以外はすべて削除できます）
2. DLL ベースの Lib をクリーンアップしてビルドします
3. DLL-Demo をビルドします

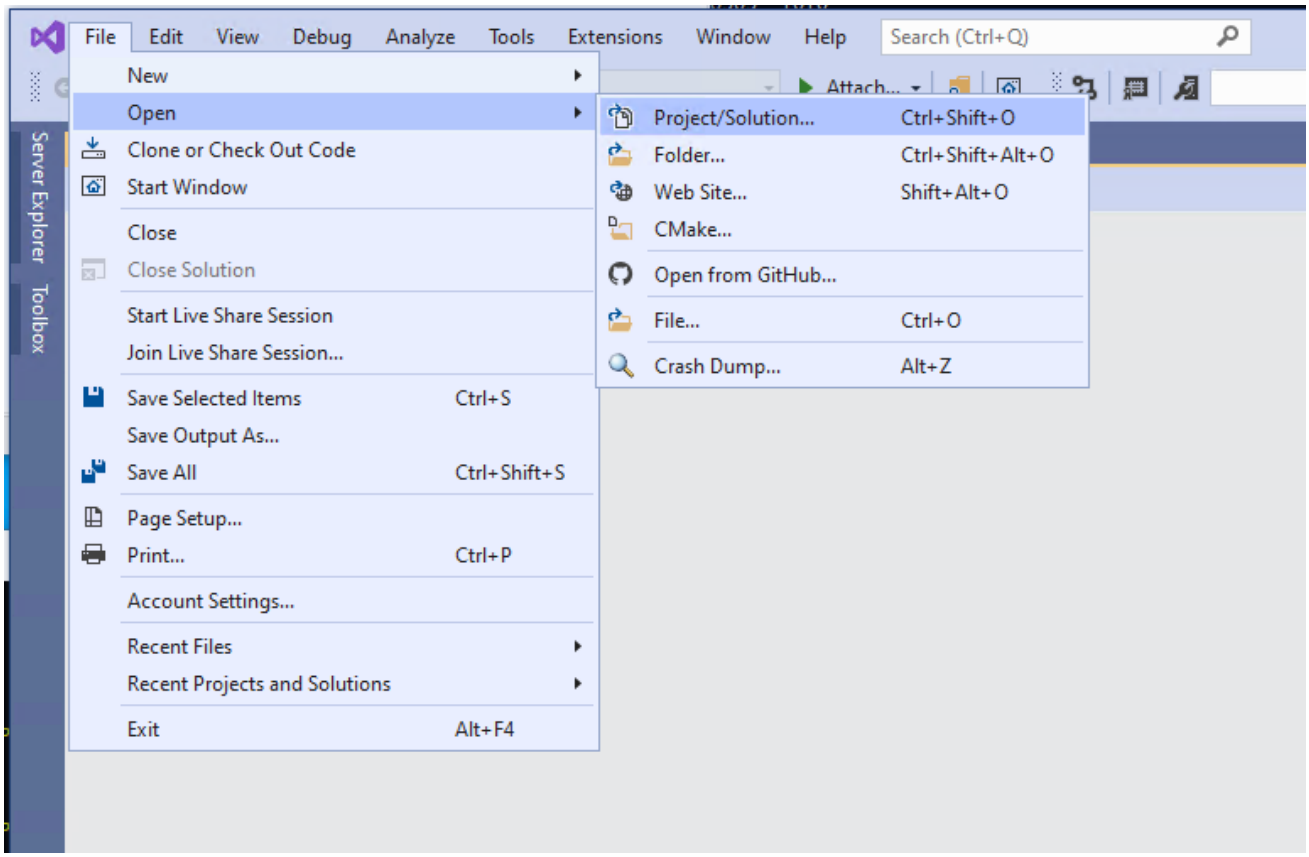
DLL の起動

上述したように、Windows はプログラムの起動前の実行時に DLL で提供されている外部参照を動的にパッチする必要があります。これを容易にするために、Windows はグローバルパス環境ディレクトリのいずれかを探して、必要な dll が見つかるかどうかを確認します。Holt は通常、これらのグローバルパスを使用しませんが、一般的に受け入れられていると思われるデフォルトパスは、.dll の成果物を /Windows/System32 ディレクトリに配置することです。あるいは、Windows は実行可能ファイルのスタートアップ・ディレクトリにある DLL アーティファクトを探します。これは Holt が使用している方法です。

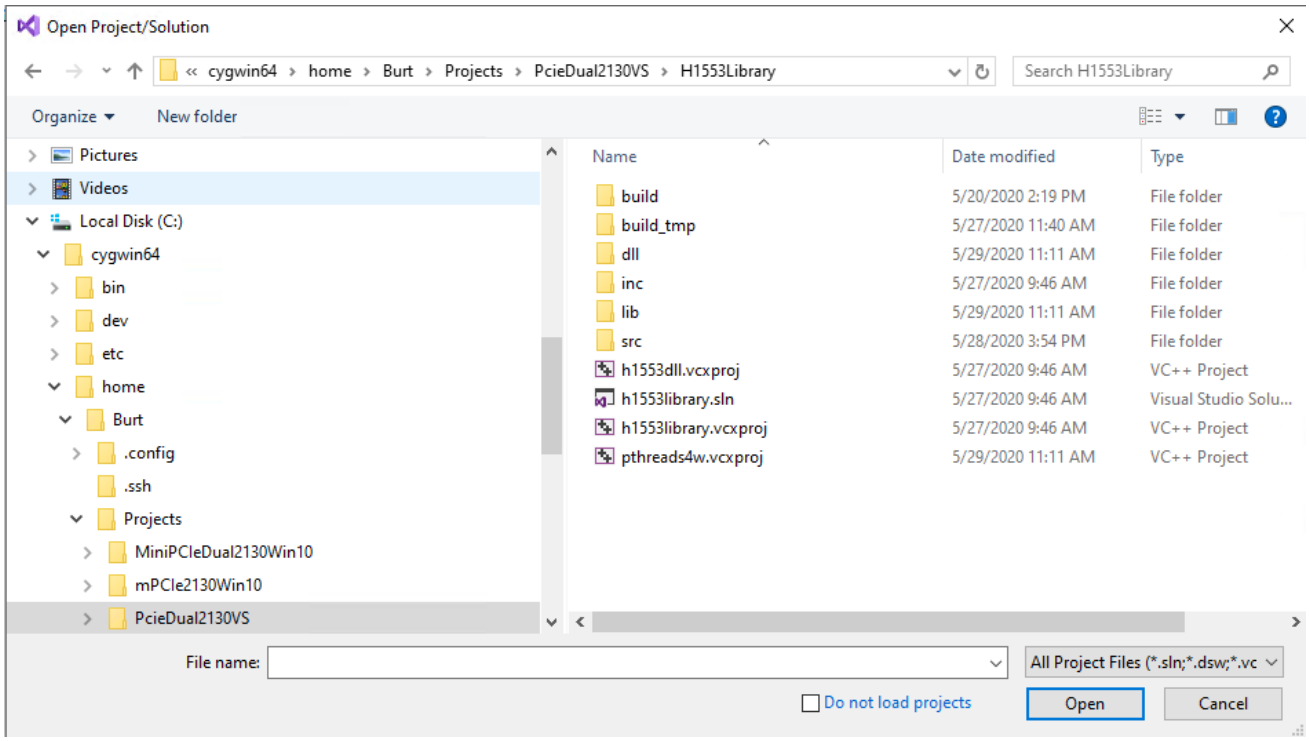
Building the two API 1553 library projects - This section requires API source files [2 つの API 1553 ライブラリ・プロジェクトのビルド : このセクションでは、API ソース・ファイルが必要]

デモをビルドするために、2 つのバリエーションをビルドします：静的ライブラリ・ベースの Holt-1553 lib デモと、動的にロードされた Holt 1553 ライブラリを使用するデモです。使用する手順は次のとおりです:最初に両方のライブラリをビルドし、最後に別のライブラリを使用するデモ用 exe ファイルをビルドします。この最初の例では、Debug 設定を使用します。

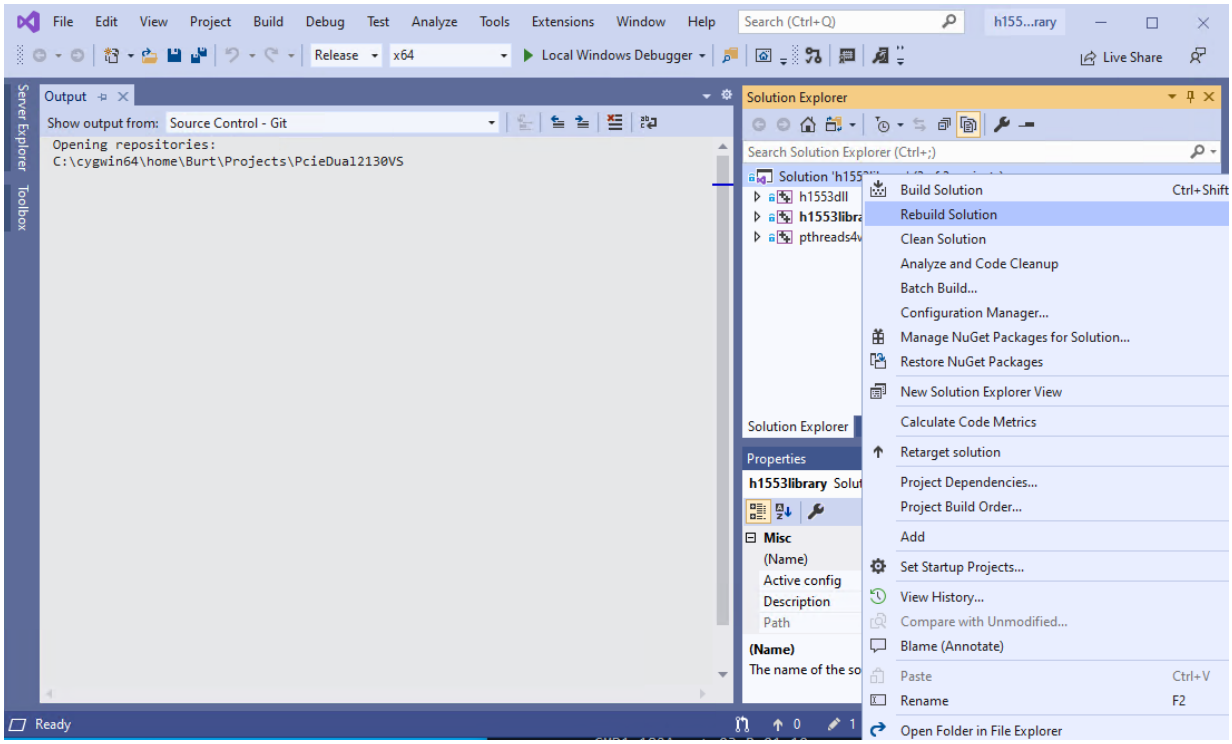
1. トップレベル/Projects/PcieDual2130VS/H1553Library に移動し、ファイル -> 開く -> プロジェクト/ソリューションダイアログを使用して、VS2019 で h1553library.sln ソリューション・ファイルを開きます。

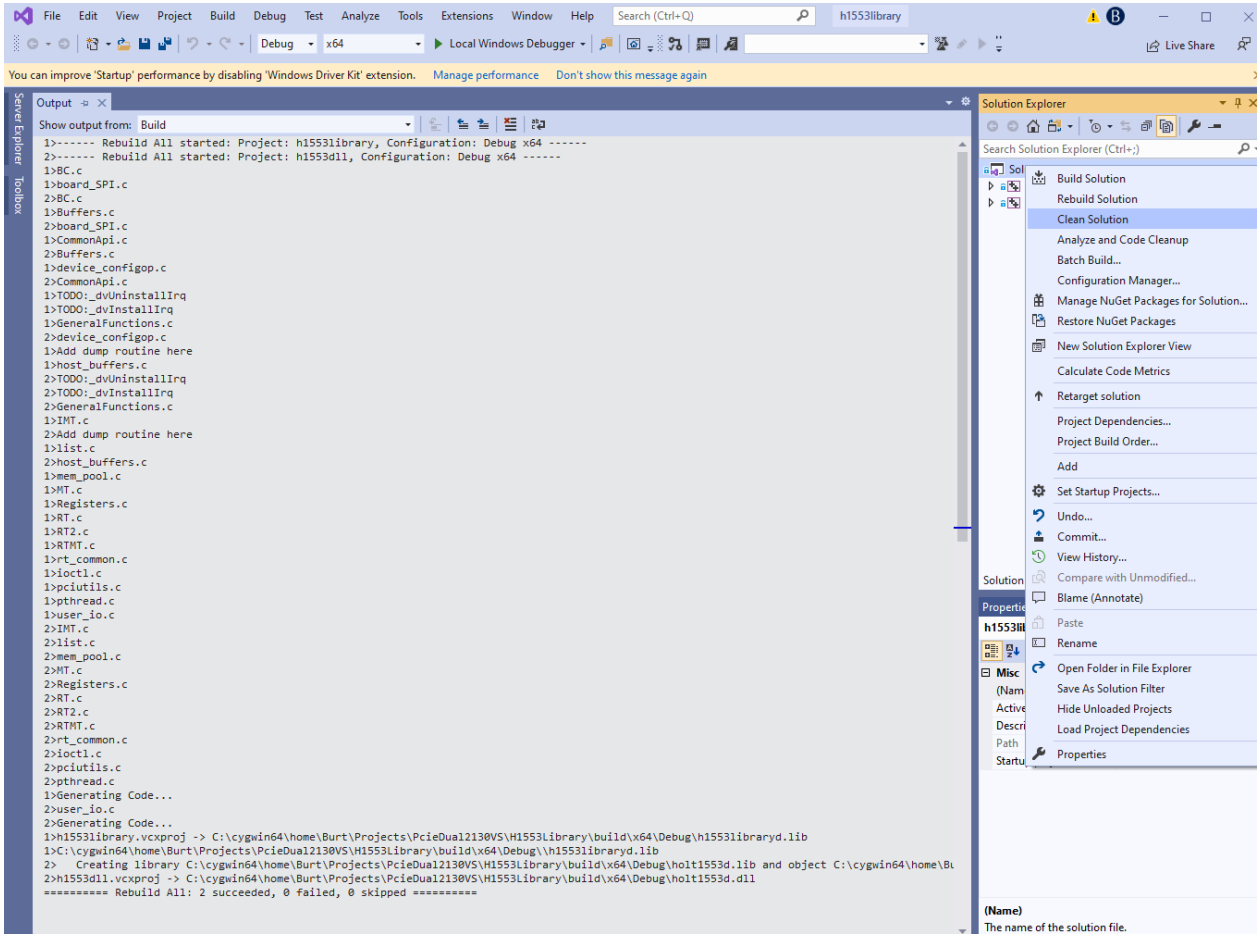


このソリューション・ファイルは、h1553.dll、h1553library、および、pthreads4w の 3 つのプロジェクトを定義しており、h1553library.sln を選択します。



2. h1553library プロジェクトを選択し(右クリック)、「Clean Solution」を選択し、「Rebuild Solution」を選択します。





3. Dll Lib も同じようにして、(PCIeDual2130VS ディレクトリから) H1553Library/dll 内のすべての *.dll を手動で Demo/build/x64/bin にコピーします。

   Copy H1553Librarydll¥*.dll Demo

4. トップレベル/Projects/PcieDual2130VS/Demo に移動し、HoltPCIeDemo.sln を開きます。「ファイル」→「開く」→「プロジェクト/ソリューション」ダイアログを使用して、VS2019 でソリューション・ファイルを開きます。このソリューション・ファイルは 2 つのプロジェクトを定義しています。HoltPCIeDemoDLL（ダイナミック・リンク・ライブラリを使用）と HoltPCIeDemoWin（static lib を使用）
5. HoltPCIeDemoWin を選択（右クリック）し、リビルドを選択します。
6. HoltPCIeemoDemoDLL についても同じことを行います。
7. ビルドが完了すると、新しくビルドされた実行ファイルはディレクトリにあります：top-level/Projects/PcieDual2130VS/Demo/build/x64/bin ディレクトリにあります。

Running the “Demo” projects [「Demo」プロジェクトの実行]

デモを実行するには複数の方法があります。

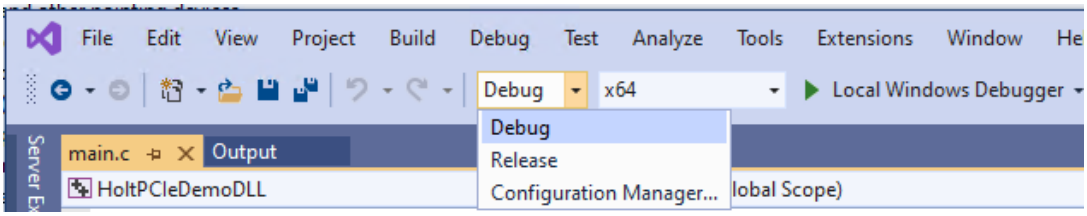
1. Win10 ファイル・エクスプローラを使用して、トップレベル/プロジェクト/PcieDual2130VS/Demo/ビルド/x64/bin を開き、HoltPCIeDemoWin.exe または、HoltPCIeDemoDLL.exe を選択します。
2. setup.exe が使用されていて、リビルドしないことを選択した場合は、Holt1553PCIe デモを Windows のスタート・メニューから（または、デスクトップ上の Holt1553Demo アイコンをダブルクリックしてください）
3. cd を使って同じディレクトリに移動し、HoltPCIeDemoWin.exe を実行します（この方法は QSG-2130mPCIe_Win10 ドキュメント)を参照してください。
4. VS2019 ポストビルドを使用して、統合デバッガを使用してブレーク・ポイントを設定してデバッグ制御下でデモをデバッグします（このオプションにはデバッグ設定を使用します）

Debugging the Demo [デモのデバッグ]

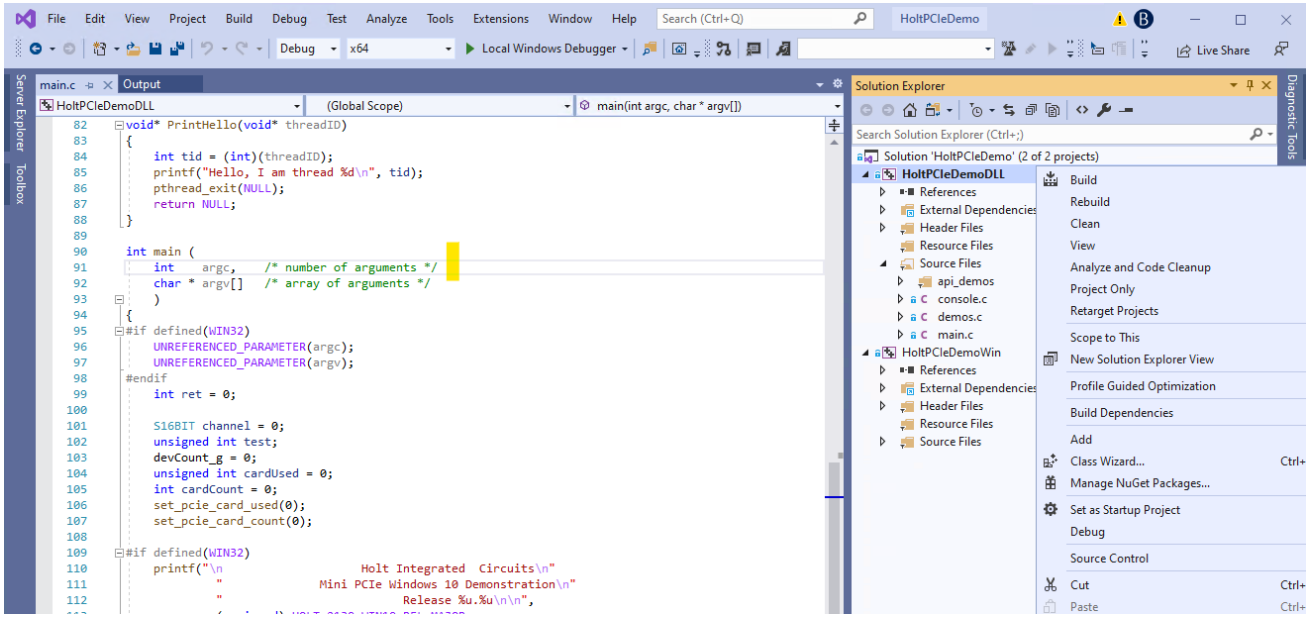
現時点では、Holt は Windows 10 カーネル PCIe ドライバをリリースしていないため、Windows 10 ユーザースペースから制御が渡されるとドライバに踏み込む方法はありません。ただし、ソフトウェア・ライセンス契約(SLA)をお持ちのお客様は、Holt1553 ライブラリ(静的および、DLL)をリビルドする際にデバッグ設定を使用して、デバッグ・ワークフロー用のライブラリ・コードにステップ・アップするために、シンボリック・デバッグ・シンボルを追加したり、最適化を無効にしたりすることができます。また、SLA を取得しておらず、Holt ライブラリのバイナリファイルのみを受け取っているお客様は、Holt ライブラリ API コードにはアクセスできませんが、ライブラリ外の Holt デモコードと同様に、ご自身のコードをデバッグすることができます。

このワークフローは、HoltPCIDemoDLL 実行ファイルのデバッグセッションを示しています。

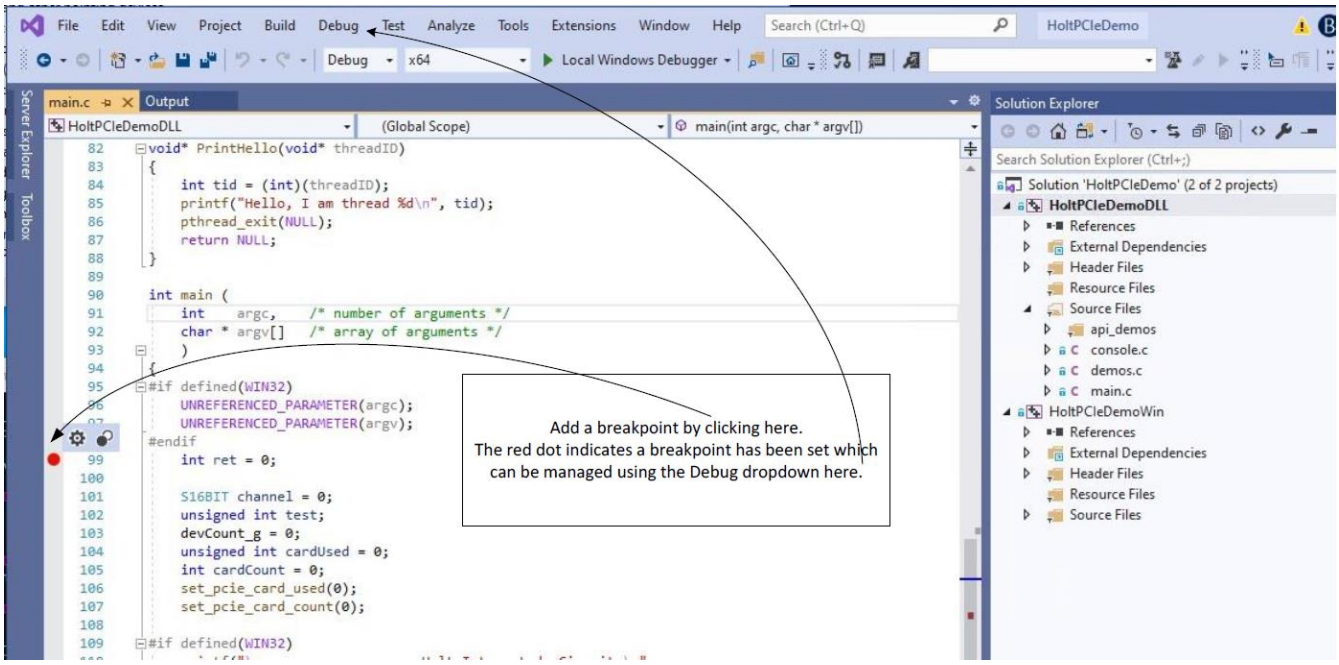
1. HoltPCIeDemo.sln ファイルを開き、選択した構成が Debug と x64 であることを確認します。



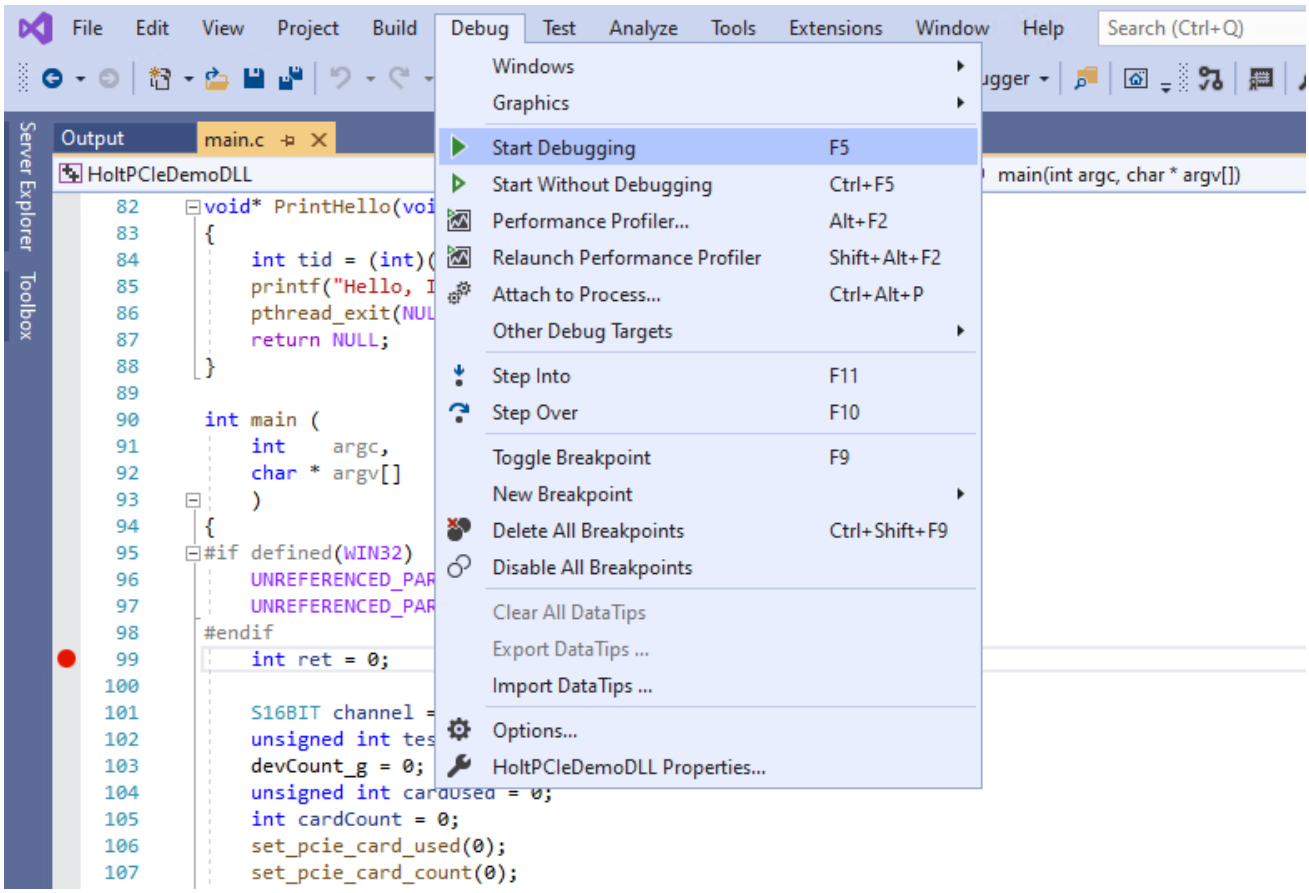
2. HoltPCIeemoDemoDLL を右クリックして Clean を選択し、次に Rebuild を選択します。



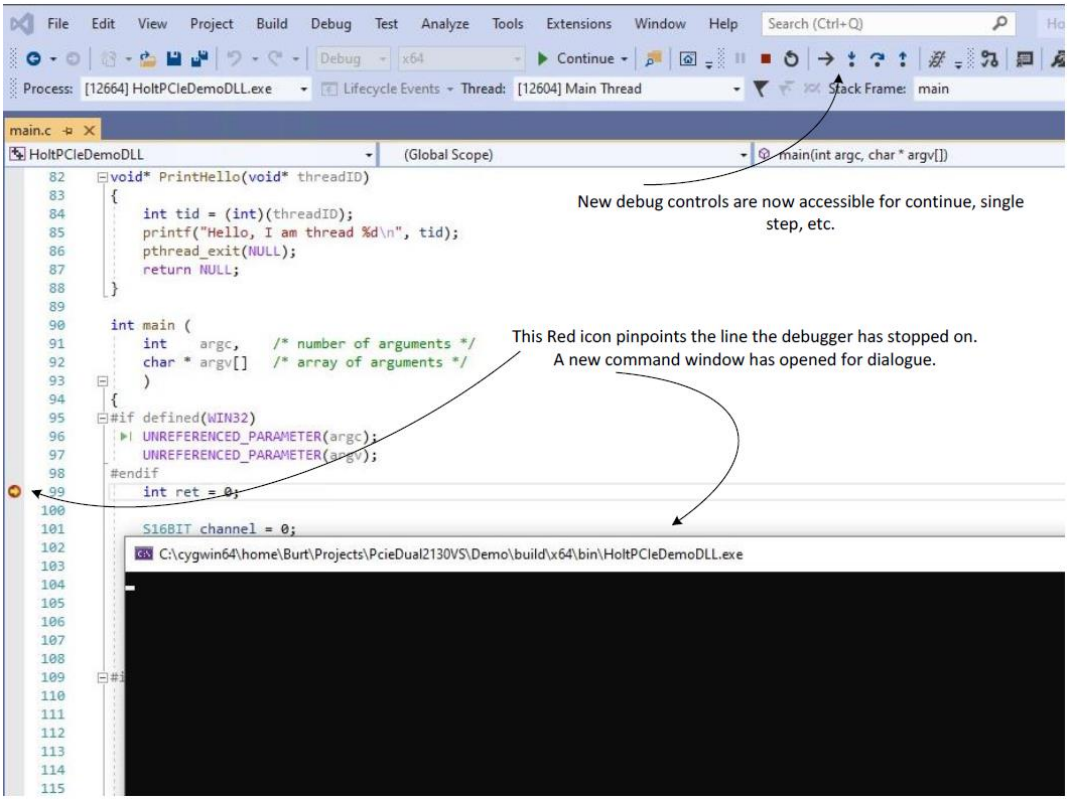
3. デバッグ制御下でデモを開始した後に制御を得るために、便利な場所にブレーク・ポイントを設定します。



4. ドロップダウン・タブを選択して「Start Debugging」を選択してデバッグを開始します。（または、単に F5 ファンクション・キーを押します）



5. 新しいコマンド・ウィンドウが開き、デバッガはあなたのコントロール・ポイントで停止します。



6. [続行]アイコンを押してプログラムを実行したり、[デバッグ]ドロップダウンを使用して条件コード付きの代替デバッグブレークポイントを設定することができるようになりました。

   a. デバッガでは、ブレーク・ポイント、シングル・ステップ、および、統合されたソース・レベルのデバッガに典型的な他のデバッガ機能を使用できます。デバッガの使用方法については、VS2019 ヘルプを参照してください。

   b. 黒いコンソール・ウィンドウにデモコンソール・メニューが表示されます。コードの変更を行う場合は、VS2019 IDE を使用してください。しかし、ターミナル・ウィンドウ（VS2019 を使用していない）から Demo を実行すると、少し良いビューと経験を提示します。

Demo project description [Demo プロジェクトの説明]

Demo ソフトウェアは、Holt 1553 Windows PCIe ドライバを初期化し、ホストが PCIe インターフェイスを介して HI-2130 デバイスと通信できるようにします。これは、main.c の HoltInitialize()で実行されます。Xilinx FPGA は、PCIe バスと両方の HI-2130 間のハードウェア・インターフェイスとして機能します。

メニューがコンソールに表示され、プログラムはユーザーがコンソール・コマンドを押すのを待ちます。コマンドは chk_key_input()関数によって受け入れられます。ユーティリティのコードの一部は console.c に含まれています。

モジュール Demo.c に含まれるすべての 1553 BC、RT、および、SMT デモは、Holt API ライブラリ関数を使用して、ターミナルを初期化および、制御します。BC はメッセージを送信するように初期化され、RT と SMT は 1553 データ・ワードを読み書きできるように初期化されます。Holt HI-613x マニュアルは、変更を加える前にコードがどのように機能するかを全体的に理解するために、デモコードとともに検討する必要があります。

Demo Preparations [Demo の準備]

デモを実行する前に、PCIe カードとブレイクアウト・ボードの間にリボン・ケーブルを接続します。

小さなリボン・ケーブルを Mini PCIe カード J4 コネクタに慎重に挿入し、もう一方の端をブレイクアウト・ボードに挿入します。ケーブルのピッチは細かく、コネクタのプラスチック製ファスナーはデリケートなので、コネクタの損傷を防ぐために注意が必要です。両手でプラスチックのファスナーを慎重に引き出し、ケーブルを挿入して、両側のファスナーを押し込みます。

ブレイクアウト・ボードを接続せずにデモを実行することは可能ですが、これは推奨されておらず、1553 バス終端がないため必ずしも機能しない場合があります。

Demo のレビューについては、いくつかの例とコンソール画面が表示された QSG を参照してください。QSG ですでにカバーされているコンソール・コマンドの一部は、ここではカバーされません。

QSG で述べたように、両方のデバイスの 2 つの RT アドレスは同じです。RT1=3 および、RT2=1。BC デモはこれら 2 つの RT アドレスにのみメッセージ・コマンドを送信するため、これによりデモが簡素化されます。そして RT は同じ RT アドレスでプログラムされます。ユーザーはコマンド「9」を使用して、他の目的で RT アドレスを変更できます。

Running the Demonstration [Demo の実行]

静的ライブラリと動的ライブラリのバリアントはどちらも同じように動作します。ここでは静的ライブラリを使用します。
プログラムは 4 つの RT を以下の RT アドレスで初期化します（これらはコマンド'9'を使って後で変更することができます）。いくつかのデモやソフトウェアでは、RT1 を RT と呼ぶこともあります。RT2 は常に RT2 と呼ばれます。

DEVICE/IC	RT	RT ADDRESS	ADK Board	Command 9 channel
Dev 0 / U7	RT (RT1)	3	ADK-2130mPCIe-1F	0
Dev 0 / U7	RT2	1	ADK-2130mPCIe-1F	1
Dev 1 / U8	RT (RT1)	3	ADK-2130mPCIe-1F ADK-2130mPCIe-2F	2
Dev 1 / U8	RT2	1	ADK-2130mPCIe-1F ADK-2130mPCIe-2F	3

1. Windows のコマンドシェルを開く：<Win ロゴキー>を押し、コマンドプロンプト・オプションが表示されていることを確認して cmd と入力します。
2. トップレベル/Projects/PcieDual2130VS/Demo/build/x64/bin に Cd し、HoltPCIeDemoWIN.exe を実行します。
3. Cmd シェルでは、複数のカードがインストールされている場合、プログラムはカードを選択するように要求します ： select 1

```
C:¥Users¥home¥Burt¥Projects¥PcieDual2130VS¥Demo¥build¥x64¥bin> HoltPCIeDemoWIN.exe
                        Holt Integrated Circuits
                    Mini PCIe Windows 10 Demonstration
                            Release 1.0

Windows found 2 PCIe Cards

Choose PCIe Card <1..2>: 1
Card Selected = 1

Opening Device: …#4&171d4e2e&0&00e3#...¥ch0
Success: chan 0
Opening Device: …#4&171d4e2e&0&00e3#...¥ch1
```

```
Success: chan 1

Number of Devices found: 2

Initial default RT addresses:
 DEV0:RT1=3 DEV0:RT2=1 DEV1:RT1=3 DEV1:RT1
 Optionally use console command '9' to change these RT addresses BEFORE
RUNNING RT

***********************************************
 Holt Integrated Circuits
 Mini PCIe Dual HI-2130 API Demo
 Demo Rev: 1.0 Compiled: May 4 2020 17:26:04
 API Lib Rev: 03-5-0
***********************************************

  BC On    SMT On    RT1 On    RT2 On

 Press 'a' or 'A' to run Dev0 or Dev1 BC Async demo.
 Press 'b' or 'B' to run Dev0 or Dev1 RT demo.
 Press 'c' or 'C' to run Dev0 or Dev1 RT2 demo.
 Press 'k' or 'K' to Enable Dev0 or Dev1 RTMT.
 Press 'l' or 'L' to send high priority BC message.
 Press 'h' or 'H' to send low priority BC message.
 Press 'n' or 'N' to run Dev0 or Dev1 BC Major Minor Frame demo.
 Press 'x' or 'X' to stop Dev0 or Dev1 BC transmissions.
 Press 'S' to run SMT demo.
 Press 't' to display RT Traffic Toggle.

 ------------ Utilities -------------
 Press 'r' or 'R' to Display Dev0 or Dev1 HI-2130 Registers.
 Press 'w' for Memory Watch window
 Press 'f' Reads J4 connector and FPGA control signals
 Press '1' for Register Write
 Press '2' for Memory Write
 Press '3' RT Mode Code data word reads
 Press '4' Master Reset and reinitializes terminals
 Press '5' Toggle Dev0 BCENA on/off
 Press '6' Toggle Dev1 BCENA on/off
 Press '9' Set RT addresses
 Press '0' Toggle between User and Demo(default) modes

Press 'M' for menu, or press any valid menu key. >>
```

メニュー・コマンドは、ADK-6138、ADK-6130-2 または、ADK-6131 のような他の Holt ADK と非常に似ています。小文字'a'、'n'、'b'、'c'、'r'は(Dev0)のコマンドを実行し、'A'、'N'、'B'、'C'および、'R'は Dev1 のコマンドを実行します。なお、-1F のカード（HI-2130 は 1 枚のみ）では、大文字のコマンドは表示されません。

以下の説明では、'r'のようないくつかのコマンドは、大文字の'R'と同等の意味を持ちます。コマンドを使用します。Dev0 の場合の例をいくつか示します。

コマンド'r'と'R'は、システム・レジスタの名前と値を画面に表示するために使用されます。

コマンド'w'は、最大 256 ワードまでのデバイス・メモリのメモリ・ダンプを表示します。これは以下のような場合に便利である。イリーガリゼーション・テーブル、コントロール・ブロック、データ・バッファなどのメモリの他の領域を表示することができます。

4. 手順 1～3 を繰り返します。新しい(2 番目の)コマンドシェルを開き、HoltPCIeDemoWIN.exe と今回はカード 2 を選択します。
5. この時点で、2 つのウィンドウが開き、それぞれが静的実行ファイルを実行しています。コマンド入力のためにここからデモ演習が始まります。
6. コマンド'r'はデバイスのシステム・レジスタを名前と値で表示します。これはレジスタの設定を変更します。
7. コマンド'w'を実行すると、メモリ・ウォッチ・ウィンドウは、それぞれのデバイスを使用しています。これは、すべてのシステム・レジスタを一目で確認するのに便利です。サブコマンドを使用することでメモリ空間を上下に移動します。これは、RT 制御のような大きなメモリ領域を見るのに便利です。ブロック、BC メッセージリスト、割り込みログテーブルのいずれかを使用します。
8. コマンド「1」、デバイス別システム・レジスタ 0x0000-0x004F に書き込む。
9. コマンド「2」、デバイスによって任意のレジスタ／メモリ 0x0000-0x7FFF に書き込みます。
10. コマンド「n」は、BC に 15 のメッセージを送信するように命令する。

Demo exercises using Device 0 [デバイス 0 を使ったデモ演習]

コマンドシェル 1 で以下を実行します。

1. これは、BC が 15 メッセージを送信し、RT1 がトラフィックデータをキャプチャして表示することを示しています。コマンド'n'または、'N'は、15 メッセージを送信するために BC を初期化します。RT データ・トラフィックを表示するためには、いくつかの追加コマンドを最初に実行する必要があります。3 つのリピート・メッセージを 5 セット送信する。SA30 はデータ・ループバック用に設定されています。つまり、最初の Receive コマンドから RT に送信されたデータは、BC にデータを送信する際に RT が参照するのと同じバッファの位置にロードされる。この 3 つのコマンドだけで、A バスと B バスの両方が BC と適切に通信していることが確認できます。データ・ワードの値は、最初の受信コマンドからのデータと一致します。

```
      BC > RT Receive Cmd, SA30, 32 words, BusA: 03-R-30-00
      RT > BC Transmit Cmd, SA30, 32 words, BusB 03-T-30-00
      RT > BC Transmit Cmd, SA30, 32 words, BusA 03-T-30-00
a) Enter  'b'  to enable Dev0 RT1
b) Enter  'k'  to enable Dev0 RT1 with SMT
c) Enter  't'  to enable RT traffic data displayed on the screen.
d) Enter  'n'  to transmit 15 messages and stops.
```

ディスプレイは以下のように表示されます。
M'を押してメニューを表示するか、有効なメニューキーを押します。

```
> b

>k

RTMT Demo
>t

Traffic Enabled
>n
>
Dev0 MSG #0000. TIME = 00040628us   BUS A   TYPE0: BC to RT
     CMD1 1BC0 --> 03-R-30-00
     DATA 0101  0202  0303  0404  0505  0606  0707  0808
          0909  1010  1111  1212  1313  1414  1515  1616
          1717  1818  1919  2020  2121  2222  2323  2424
          2525  2626  2727  2828  2929  3030  3131  3232

     STA1 1800

Dev0 MSG #0001. TIME = 00041324us  BUS B   TYPE1: RT to BC
     CMD1 1FC0 --> 03-T-30-00
     STA1 1800
     DATA 0101  0202  0303  0404  0505  0606  0707  0808
          0909  1010  1111  1212  1313  1414  1515  1616
          1717  1818  1919  2020  2121  2222  2323  2424
          2525  2626  2727  2828  2929  3030  3131  3232


Dev0 MSG #0002. TIME = 00042020us  BUS A   TYPE1: RT to BC
```

```
     CMD1 1FC0 --> 03-T-30-00
     STA1 1800
     DATA 0101  0202  0303  0404  0505  0606  0707  0808
          0909  1010  1111  1212  1313  1414  1515  1616
          1717  1818  1919  2020  2121  2222  2323  2424
          2525  2626  2727  2828  2929  3030  3131  3232


Dev0 MSG #0003. TIME = 00042720us BUS A TYPE0: BC to RT
     CMD1 1BC0 --> 03-R-30-00
     DATA 0101  0202  0303  0404  0505  0606  0707  0808
          0909  1010  1111  1212  1313  1414  1515  1616
          1717  1818  1919  2020  2121  2222  2323  2424
          2525  2626  2727  2828  2929  3030  3131  3232

     STA1 1800

Dev0 MSG #0004. TIME = 00043416us BUS B TYPE1: RT to BC
     CMD1 1FC0 --> 03-T-30-00
     STA1 1800
     DATA 0101  0202  0303  0404  0505  0606  0707  0808
          0909  1010  1111  1212  1313  1414  1515  1616
          1717  1818  1919  2020  2121  2222  2323  2424
          2525  2626  2727  2828  2929  3030  3131  3232
```

2. BC メジャーフレームおよび、マイナーフレームを送信

コマンド「a」は、BC が Major/Minor フレーム形式で連続したメッセージを送信していることを示す。送信を停止するには、'x'の後に'return'を押す。ここでは最初の 7 つのメッセージのみを表示しています。前回のデモで RT、SMT、トラフィックコマンドはすでに有効になっていたので、「a」を押してこのデモを実行します。RT から RT へのメッセージは'no response'エラーを持っていることに注意してください - これは RT2 が有効にされていないためです。

```
> a
 Dev0 MSG #0015. TIME = 00004238us  BUS A   TYPE0: BC to RT
       CMD1 1822 --> 03-R-01-02
       DATA 0005 0002
       STA1 1800

 Dev0 MSG #0016. TIME = 00004286us BUS A TYPE2: RT to RT
       CMD1 182A --> 03-R-01-10
       CMD2 0C2A --> 01-T-01-10
 ERROR: NORES

 Dev0 MSG #0017. TIME = 00004352us BUS A TYPE2: RT to RT
       CMD1 182A --> 03-R-01-10
       CMD2 0C2A --> 01-T-01-10
 ERROR: NORES

 Dev0 MSG #0018. TIME = 00004416us BUS B TYPE2: RT to RT
       CMD1 182A --> 03-R-01-10
```

```
    CMD2 0C2A --> 01-T-01-10
ERROR: NORES

Dev0 MSG #0019. TIME = 00104176us BUS A TYPE0: BC to RT
    CMD1 1822 --> 03-R-01-02
    DATA 0005 0002
    STA1 1800

Dev0 MSG #0020. TIME = 00104224us BUS A TYPE2: RT to RT
    CMD1 182A --> 03-R-01-10
    CMD2 0C2A --> 01-T-01-10
ERROR: NORES

Dev0 MSG #0021. TIME = 00104290us BUS A TYPE2: RT to RT
    CMD1 182A --> 03-R-01-10
    CMD2 0C2A --> 01-T-01-10
ERROR: NORES
```

3. RT2 を有効にします。
   前の BC メッセージが送信中の場合は、「x」キーを押した後、「return」キーを押して BC を停止させます。c'コマンドを押すと RT アドレス 1 に設定されている RT2 が有効になります。
4. 'a'をもう一度押すと、今度は RT から RT へのメッセージが「応答なし」エラーなしで適切に応答します。' x'を押して戻り、BC を停止してメッセージを表示します。以下に一部のメッセージのみを表示します。

```
>c
>a
>
 Dev0 MSG #0043. TIME = 00127134us BUS A TYPE0: BC to RT
     CMD1 1822 --> 03-R-01-02
     DATA 0005 0002
     STA1 1800

 Dev0 MSG #0044. TIME = 00127436us BUS A TYPE2: RT to RT
     CMD1 182A --> 03-R-01-10
     CMD2 0C2A --> 01-T-01-10
     STA1 0800
     DATA BBBB  0202  1414  0404  0505  0606  0707  0808
          0909  1010
     STA2 1800

 Dev0 MSG #0045. TIME = 00096012us BUS A TYPE0: BC to RT
     CMD1 1822 --> 03-R-01-02
     DATA 0005  0002
     STA1 1800

 Dev0 MSG #0046. TIME = 00096316us BUS A TYPE2: RT to RT
     CMD1 182A --> 03-R-01-10
     CMD2 0C2A --> 01-T-01-10
     STA1 0800
```

```
       DATA BBBB  0202  1414  0404  0505  0606  0707  0808
            0909  1010
       STA2 1800

 Dev0 MSG #0047. TIME = 00064942us BUS A TYPE0: BC to RT
       CMD1 1822 --> 03-R-01-02
       DATA 0005  0002
       STA1 1800
```

コマンド「a」: 100ms ごとに繰り返しメッセージのセットを継続的に送信するように BC に命令します。部分的なリストについては、QSG を参照してください。コマンド「x」: メッセージを停止します。

コマンド「l」: BC Async デモ（コマンド「a」）の実行中に、Bus B 上に 3 つ事前定義された優先度の低い BC メッセージをメッセージ・シーケンスに挿入します。これは、1 度だけ行われます。

```
Dev0 MSG #1694. TIME = 00070698us   BUS B   TYPE0: BC to RT
       CMD1 0822 --> 01-R-01-02
       DATA DEAD BEEF
       STA1 0800

Dev0 MSG #1695. TIME = 00071054us   BUS B   TYPE1: RT to BC
       CMD1 0C2F --> 01-T-01-15
       STA1 0800
       DATA BBBB  0202  1414  0404  0505  0606  0707  0808
            0909  1010  1111  1212  1313  1414  1515

Dev0 MSG #1696. TIME = 00071210us   BUS B   TYPE0: BC to RT
       CMD1 0825 --> 01-R-01-05
       DATA CAFE  CODE  0303  0404  0505
       STA1 0800
```

コマンド「h」: コマンド「l」と同様の BC Async デモの実行中に、事前定義された高優先度 BC メッセージをメッセージ・シーケンスに挿入しますが、繰り返し可能です。これは、他の 2 つのメッセージの間に挿入されたメッセージ#1553 を示しています。

```
Dev0 MSG #1532. TIME = 00129598us   BUS A   TYPE2: RT to RT
       CMD1 182A --> 03-R-01-10
       CMD2 0C2A --> 01-T-01-10
       STA1 0800
       DATA BBBB  0202  1414  0404  0505  0606  0707  0808
            0909  1010
       STA2 1800

HoltBCSendAsyncMsgHP returns 0
>
Dev0 MSG #1533. TIME = 00042566us BUS B TYPE0: BC to RT
       CMD1 0822 --> 01-R-01-02
       DATA DEAD  BEEF
       STA1 0800

Dev0 MSG #1534. TIME = 00098222us BUS A TYPE0: BC to RT
       CMD1 1822 --> 03-R-01-02
       DATA 0005  0002
       STA1 1800
```

Demo exercises using Device 1: ADK-2130mPCIe-2F [デバイス 1 : ADK-2130mPCIe-2F を使用したデモ演習]

上記のデモが完了したら、コマンドシェル 1 で以下を実行してください:

2 枚目の HI-2130(Dev1)をデモするためには、HI-2130 を 2 枚搭載した Holt IC カードが必要です。
2130mPCIe-2F です。同じコマンドを大文字で実行します。
B'、'K'、'N'を全て大文字で押して、2 回目の HI-2130(Dev1)BC と RT1 を実行します。Dev1」は
と表示されています。n コマンドは同じ 15 のメッセージを送信します。
(3 は以下の通り)ですが、今回は 2 台目の HI-2130(Dev1)の BC を使用しています。

```
> B

> K

RTMT Demo
>N

>
Dev1 MSG #0075. TIME = 00124310us  BUS A   TYPE0: BC to RT
           CMD1 1BC0 --> 03-R-30-00
           DATA 0101  0202  0303  0404  0505  0606  0707  0808
                0909  1010  1111  1212  1313  1414  1515  1616
                1717  1818  1919  2020  2121  2222  2323  2424
                2525  2626  2727  2828  2929  3030  3131  3232

           STA1 1800

Dev1 MSG #0076. TIME = 00125006us  BUS B  TYPE1: RT to BC
           CMD1 1FC0 --> 03-T-30-00
           STA1 1800
           DATA 0101  0202  0303  0404  0505  0606  0707  0808
                0909  1010  1111  1212  1313  1414  1515  1616
                1717  1818  1919  2020  2121  2222  2323  2424
                2525  2626  2727  2828  2929  3030  3131  3232

Dev1 MSG #0077. TIME = 00125714us  BUS A  TYPE1: RT to BC
           CMD1 1FC0 --> 03-T-30-00
           STA1 1800
           DATA 0101  0202  0303  0404  0505  0606  0707  0808
                0909  1010  1111  1212  1313  1414  1515  1616
                1717  1818  1919  2020  2121  2222  2323  2424
                2525  2626  2727  2828  2929  3030  3131  3232
```

Demo exercises using Card 2: Devices 0 and 1: ADK-2130mPCIe-2F [カード 2：デバイス 0 と 1：ADK-2130mPCIe-2F を使用したデモ演習]

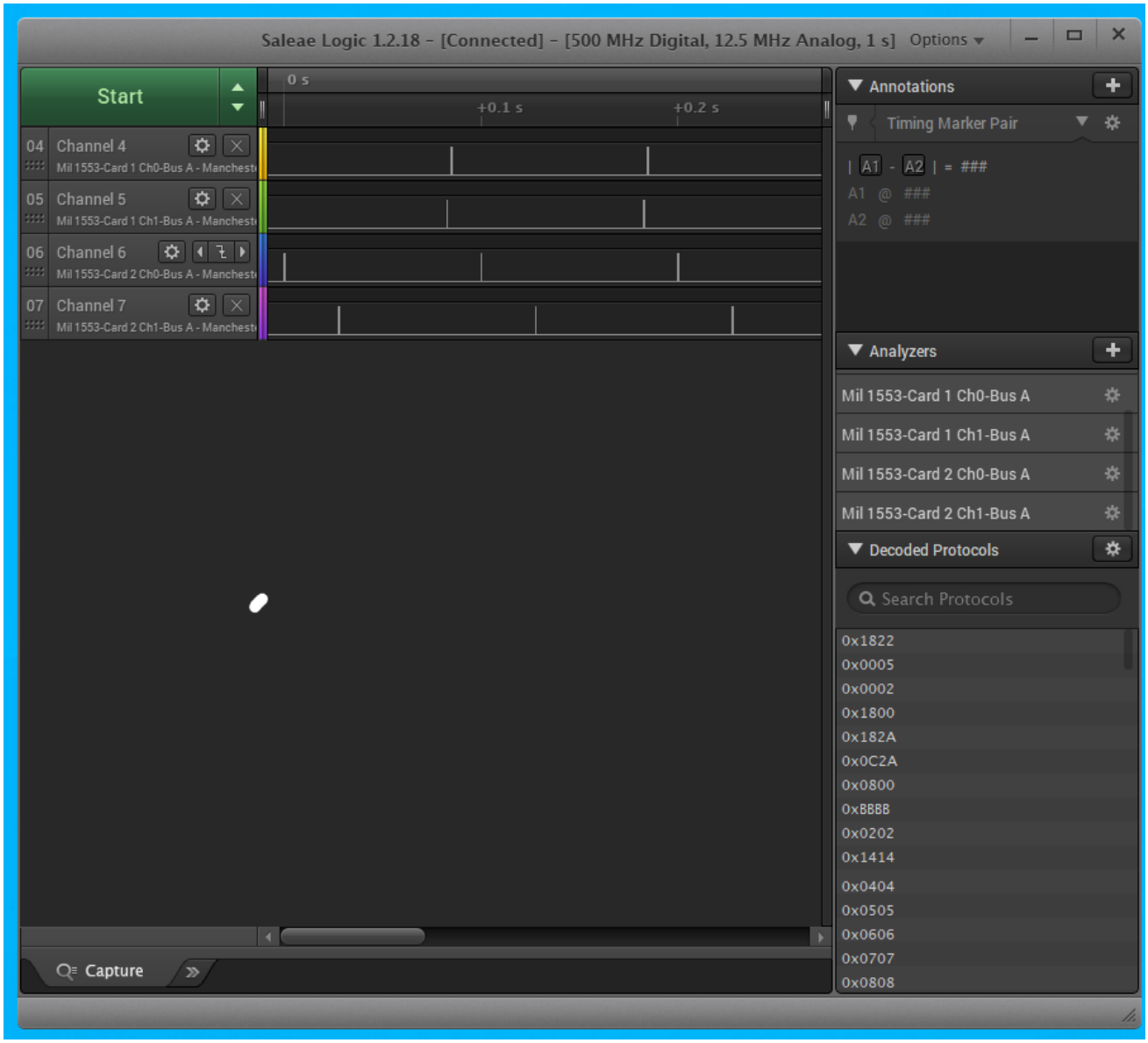
2 枚目のカードが利用可能な場合

カード 1 が現在、上記の b, k, t, n, B, K, N, c, C, a, A のデモを実行している間に、2 番目のコマンドシェルで同じシーケンスを正確に繰り返す。両方のコマンドシェルでチャネル 0 と 1 でトラフィックが進行していることを示していることに注意してください。

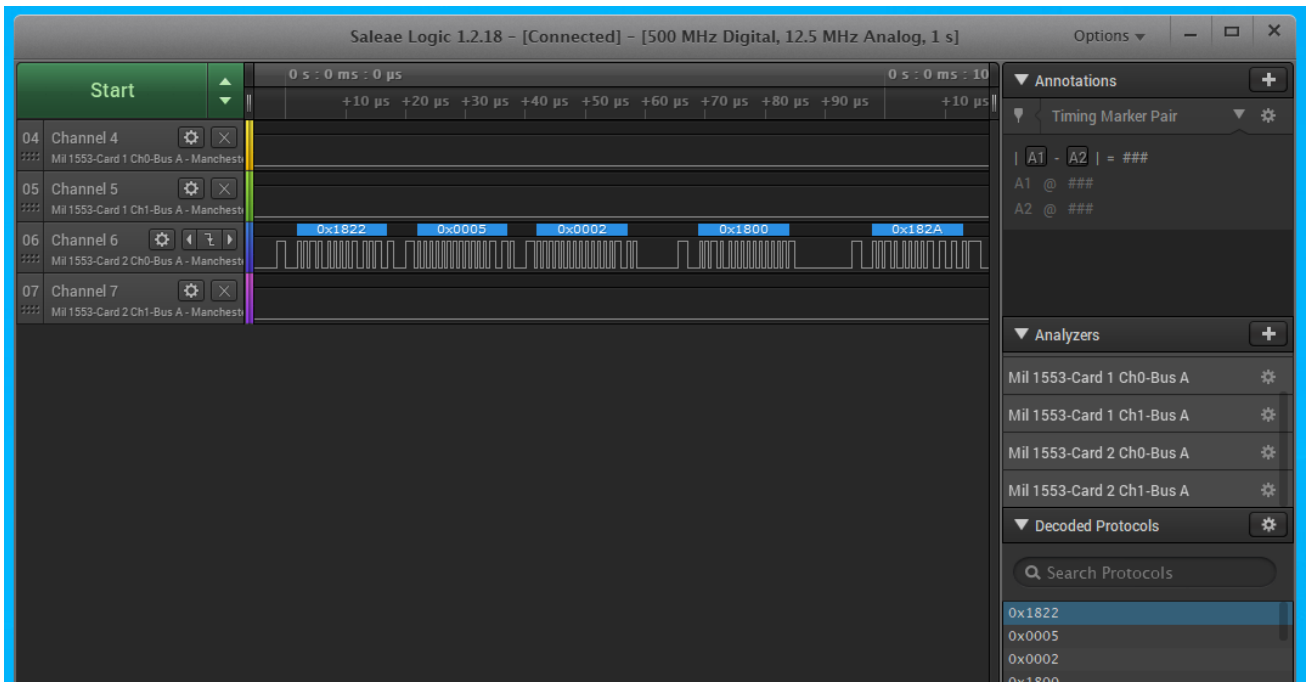
Using a Saleae Logic Pro 16 to view 1553 transmissions: [Saleae Logic Pro 16 を使用して 1553 トランザクションを観測]

Saleae Logic Pro 16 USB アナライザを使用して 1553 信号をキャプチャすることは可能ですが、接続を行うためには、ケーブルブレークアウトボードの裏側にハンダ付けする必要があります。カードが 1553 バスカプラと一緒に使用されている場合、これを行うには制限があります。電圧減衰のため、バスカプラを通過する外部端子からデータを受信できない場合があります。このドキュメントで説明されているほとんどのデモでは、バスカプラがなくても正常に動作します。外部端子があれば、Holt デバイスがデータを送信するときにサレアは十分な信号を受信できます。より良いツールは、Ballard USB 1553 デバイスのような 1553 ツールを使用することです。以下は、4 つの Saleae チャンネル（#4-7）を 2 枚の別々の-2f カードに接続したスクリーンショットです：

Channel 4: Card 1, 2130 #1, Bus A
Channel 5: Card 1, 2130 #2, Bus A
Channel 6: Card 2, 2130 #1, Bus A
Channel 7: Card 2, 2130 #2, Bus A



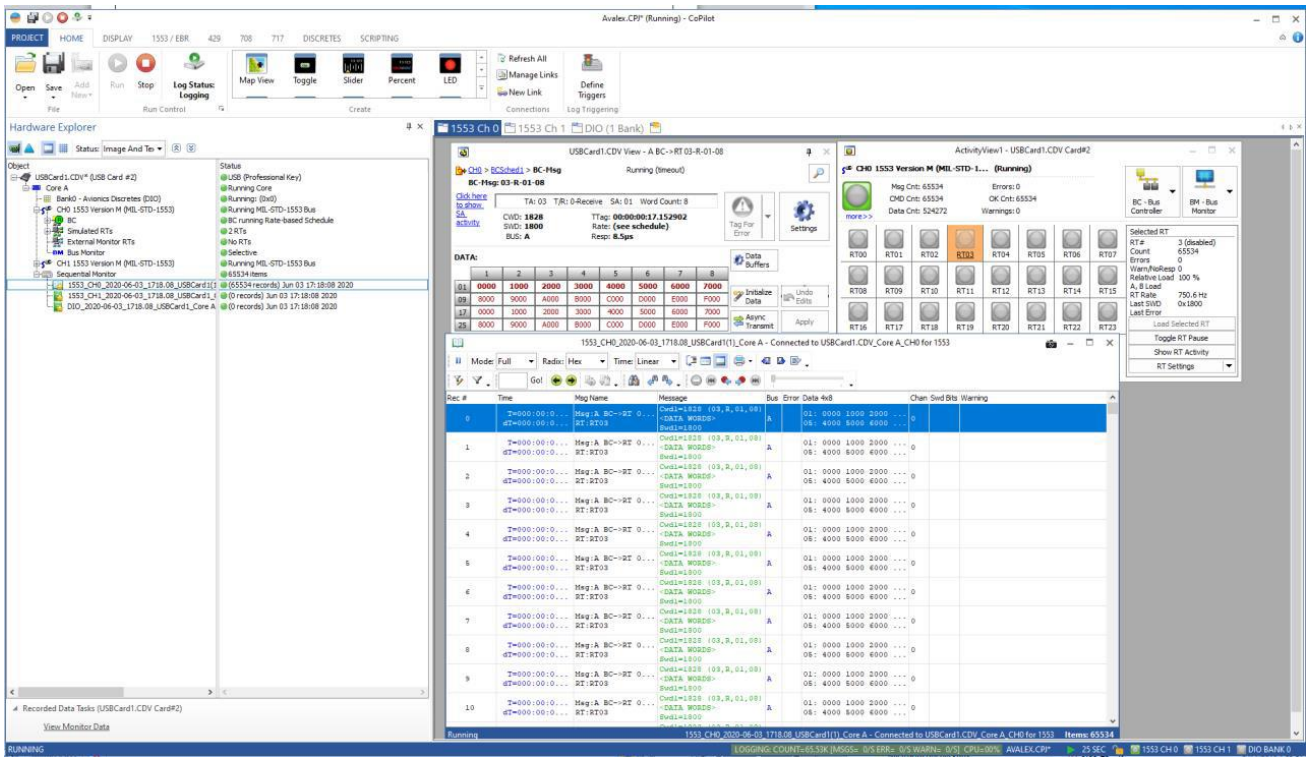
最初のトランザクションを拡大すると、見つかるはずでしょう：


Saleae Logic 1.2.18 - [Connected] - [500 MHz Digital, 12.5 MHz Analog, 1 s]
Options
Start
0 s : 0 ms : 0 µs
0 s : 0 ms : 10
+10 µs +20 µs +30 µs +40 µs +50 µs +60 µs +70 µs +80 µs +90 µs
+10 µs
04 Channel 4
Mil 1553-Card 1 Ch0-Bus A - Manchesto
05 Channel 5
Mil 1553-Card 1 Ch1-Bus A - Manchesto
06 Channel 6
Mil 1553-Card 2 Ch0-Bus A - Manchesto
07 Channel 7
Mil 1553-Card 2 Ch1-Bus A - Manchesto
0x1822
0x0005
0x0002
0x1800
0x182A
Annotations
Timing Marker Pair
| A1 - A2 | = ###
A1 @ ###
A2 @ ###
Analyzers
Mil 1553-Card 1 Ch0-Bus A
Mil 1553-Card 1 Ch1-Bus A
Mil 1553-Card 2 Ch0-Bus A
Mil 1553-Card 2 Ch1-Bus A
Decoded Protocols
Search Protocols
0x1822
0x0005
0x0002

Using an external 1553 tester tool to view 1553 transmissions such as Ballard USB 1553 device. [外部の 1553 テスターツールを使用して、Ballard USB 1553 デバイスなどの 1553 伝送を表示]

Ballard 1553 アナライザツール（または、他の 1553 テスタ）を 1553 バスカプラに接続して、外部 BC 送信、RT 受信、または、MT メッセージのモニタリングを可能にすることもできます。このデモでは、MIL-STD-1553 バス上のバスアクティビティを監視するために、Ballard USB-1533 ポータブル Avionic Interface を使用しています。PC には Ballard CoPilot Avionics Databus Tool が実行されています。(ソフトウェアリビジョンは、Ballard USB CPUA1133: Copilot:Version 7.30.71315)セットアップは以下の通りです：

Holt Mini PCIe Dual MIL-STD-1553 ブレイクアウト・ボードは、デモで駆動される-1F または、-2F 2130 Mini PCIe PCB に接続する必要があります。ブレイクアウト・ボードは、MIL-STD 1553 信号を遮断する細ピッチのリボン・ケーブルで接続されています。Excalibur ESI-410 を使用して適合 MIL-1553 バスを作成し、ブレイクアウト・ボードの A バスの A-Bus 接続を Excalibur のスタブ J4 と J6 に接続します。セットアップを完了するには、Ballard USB Copilot Tool を Excalibur のスタブ J4 に接続します。Excalibur の Bus-J1 と Bus-J2 コネクタには、適切なバスターミネータが必要です。2f 2130 カードの A バス、または、-1f カードのシングル A バスの両方にトラフィックを生成するには、上記の 2 つのデモンストレーションを実行します。これら 2 つのデモンストレーションのキーシーケンスは次のとおりです：b、k、t、n、B、K、N、c、C、a、A（この順）。このセットアップが完了すると、Ballard Co-pilot ソフトウェアは、以下のスクリーンキャプチャのように、Holt Mini PCIe デモによって生成されたバストランザクションをモニタすることができます。



上記を見てもわかるように、Saleae でキャプチャしたものと同じようなトランザクションを行っています。

Demonstrating the Holt Mini PCIe RT with an external BC [Holt Mini PCIe RT を外部 BC でデモ]

4 つの RT のうちの 1 つは、外部の BC テスターで使用することができます。希望のチャンネルを選択し、コマンド'9'を使用して BC メッセージに合わせて RT アドレスを設定します。外部 BC をバスカプラに接続し、バスカプラのスタブをオプションの Holt ブレイクアウト・ボードの BNC コネクタに接続します。バスの正しい接続方法は AN-551 を参照してください。

RT ターミナル・アドレスを変更するには、RT を有効にする前に'9'を押してください。' b'、'B'、'c'、'C'コマンドで RT を有効にした場合は、コマンド'4'を使用して最初にカードをリセットしてください。プログラムは最初にチャンネル番号 0-4 の後に RT アドレスの値を要求します。

```
Press  'M'  for menu, or press any valid menu key. >>
>9
Enter RT channel 0-3 for Dev0 RT1, Dev0 RT2, Dev1 RT1 or Dev1 RT2
0
Enter RT address: 5
 0 5
```

使用する RT とチャンネルに応じて、対応する'c'、'C'、'b'または、'B'コマンドで RT を有効にします。

'k'または、'K'と't'を有効にします。RT は BC コマンド受信の準備ができています。外部 BC メッセージテスターが対応する RT アドレスに設定されていることを確認してください。t'コマンドを使用したトラフィックデータの表示は任意であるが, カードを最初に学習する際には推奨される. BC メッセージのトランザクションは、't'コマンドが使用されているかどうかに関わらず発生するが、't'は単にコンソールにデータを表示する機能を有効にするだけである。

Demonstrating the Holt Mini PCIe BC With an External RT. [Holt Mini PCIe BC を外部 RT でデモ]

BC デモは RT アドレス 3 と 1 にメッセージを送信します。これらの RT アドレスを変更するには、プログラムを修正して再コンパイルする必要があります。

外部 RT をブレイクアウト・ボードの 3 軸ジャックの 1 つに接続します。外部 RT とカード BC の接続にはバスカプラを使用してください。適切なバス接続については AN-551 を参照してください。

カード上の RT のいずれかが同じ RT アドレスを持っている場合、バスコンフリクトが発生する可能性があります。この場合は、コマンド'9'で RT アドレスを異なる値に変更するか、コマンド'4'でカードをリセットして端末を再初期化してください。この時間は、カードの RT のいずれかを有効にしないでください。

コマンド'n'または、'N'を使用して、メッセージを送信するために BC をコマンドします。RT アドレスが BC コマンドワードの RT アドレスと一致した場合、外部 RT はメッセージを受信する必要があります。

BC が外部 RT または、RT とモニタに送信しているため、Demo コンソールの RT トラフィックデータは表示されません。外部 Ballard USB 1553 ツールを使用している場合、メッセージはそこに表示されるはずです。

コマンド「S」は、現在の SMT コマンドとデータ・バッファのアドレスを表示します。
この例では、両方のデバイスで BC と両方の RT を有効にし、コマンド'a'と'A'を使用して両方の BC から送信を開始するために必要な手順を示しています。両方のデバイスは-2F カードでのみ使用可能なので、-1F カードでは大文字のコマンドを入力する必要はなく、以下の出力には Dev1 メッセージは表示されません。Dev0 からのメッセージと Dev1 からのメッセージがあることに注意してください。

```
>> b
>c
>k (RTMT Demo)
>t (Traffic Enabled)
>B
>C
>K
RTMT Demo
>a
>A (this will be seen much later inter-mixed in the messages below)
With a -2F card a mixed of messages from Dev0 and Dev1 are displayed.
Dev0 MSG #0283. TIME = 00130428us BUS A TYPE2: RT to RT
        CMD1 182A --> 03-R-01-10
        CMD2 0C2A --> 01-T-01-10
        STA1 0800
        DATA BBBB  0202  1414  0404  0505  0606  0707  0808
             0909  1010
        STA2 1800

Dev1 MSG #0284. TIME = 00090538us BUS A TYPE0: BC to RT
        CMD1 1822 --> 03-R-01-02
        DATA 0005  0002
        STA1 1800

Dev1 MSG #0285. TIME = 00090842us BUS A TYPE2: RT to RT
        CMD1 182A --> 03-R-01-10
        CMD2 0C2A --> 01-T-01-10
        STA1 0800
        DATA BBBB  0202  1414  0404  0505  0606  0707  0808
             0909  1010
        STA2 1800

Dev0 MSG #0286. TIME = 00099052us BUS A TYPE0: BC to RT
        CMD1 1822 --> 03-R-01-02
        DATA 0005  0002
        STA1 1800

Dev0 MSG #0287. TIME = 00099356us BUS A TYPE2: RT to RT
        CMD1 182A --> 03-R-01-10
        CMD2 0C2A --> 01-T-01-10
        STA1 0800
        DATA BBBB  0202  1414  0404  0505  0606  0707  0808
             0909  1010
        STA2 1800
```

注：コマンド「9」（リセット後のみ）は、4 つの RT の RT アドレスを変更するために使用することができます。BC デモ、'n'と'a'は RT アドレス 1 と 3 にメッセージを送信します。BC メッセージの RT アドレスは、デモコードファイル「demo.c」の BC メッセージ関数：bcAsync()と MajorMinorframe()に設定されています。これらの RT アドレスを BC コマンドワードで変更するには、コードの中でアドレスを変更する必

要があります。

コマンド'0'はデモ・ボードとユーザ・ボードを切り換えます。ディフォルトは「Demo」モードです。コマンド「f」は、Dev0 と Dev1 の両方に送られる FPGA 制御信号の状態を読み込んで表示します。信号は次のセクションの表 1 に記載されています。デモおよび、ユーザー・モードの説明は次のセクションを参照してください。

```
Press 'M' for menu, or press any valid menu key. >> f

Dev 0 values:
 P_BC_TRIG = 0
 P_NMR = 1
 P_TEST = 0
 P_BCENA = 1
 P_NTRUM = 1
 P_RT1ENA = 1
 P_RT2ENA = 1
 P_INHIBIT = 0
 P_INPUTCONTROL demo/user mode = 0 0=demo, 1=user
 P_SPAREINPUT = 0
 P_MTPKTRDY = 1
 P_ACTIVE = 0
 P_RT1MC8 = 1
 P_RT2MC8 = 1
 P_READY = 1
Dev 1 values:
 P_BC_TRIG = 0
 P_NMR = 1
 P_TEST = 0
 P_BCENA = 1
 P_NTRUM = 1
 P_RT1ENA = 1
 P_RT2ENA = 1
 P_INHIBIT = 0
 P_INPUTCONTROL demo/user mode= 0 0=demo, 1=user
 P_SPAREINPUT = 0
 P_MTPKTRDY = 1
 P_ACTIVE = 0
 P_RT1MC8 = 1
 P_RT2MC8 = 1
 P_READY = 1
>
```

Trouble shooting [トラブル・シューティング]

特定のエラーや問題が発生した場合のトラブル・シューティングのヒントは、QSG のドキュメントを参照してください。

Project Customizations [プロジェクト・カスタマイズ]

Holt プロジェクトに慣れてきた後のプロジェクトのカスタマイズの検討。

- プロジェクトをデモに適合させるが、Holt ライブラリのビルドサブアーキテクチャはそのままにしておく（デモをユーザーアプリケーションに置き換える）
- ビルド環境のワークフローに API ライブラリ・ソリューションを追加するだけです。

FPGA HI-2130 Dev0 and Dev1 control signals [FPGA HI-2130 Dev0 および Dev1 制御信号]

2 つの HI-2130 デバイス（Dev0/Dev1）と通信するために、共通のパラレル 16Bit アドレスおよび、データ・バスが FPGA に実装されています。これらの信号は回路図の 2 ページに記載されています。

両方のデバイスのターミナル制御信号（HI-2130）は、I/O レジスタ 0x8000～0x8013 にアクセスするためのメモリマップソフトウェア書き込み機能を使用してホストによって制御される FPGA I/O に接続されます。書き込み可能な信号は、1 を書き込んで High、または、0 を書き込んで Low に設定することができます。例えば、Dev0 の BCENA を High に設定する場合：「Chan」パラメータは、パラメータを 1 または、0 に設定することにより、Dev0 または、Dev1 を指定します。

```
HoltRegWrite(Chan, P_BCENA, 1); // Chan=0 for Dev0.
```

Demo Mode vs. User Mode [Demo モードと User モード]

ソフトウェア・デモンストレーションを簡単にするために、INPUT_CONTROL (0x800D)が Low に設定されている場合は、Demo モードが選択され、High の場合は User モードが選択されます。電源投入時のディフォルトは Demo モードです。

Demo モードは、一部の制御信号はディフォルトで有効な状態にプリセットされているため、ターミナルはすべて有効になっています（BCENA=1、RTENA=1、MTENA=1 など）。

ユーザー・モードでは、ユーザーは BC、RT、または、MT のイネーブル制御信号の任意の組み合わせを設定して、各デバイスのこれらのターミナルを有効または、無効にできます。User モードでは、2 つの制御信号、BCENA および、BCTRIG が J4 コネクタ・ピンから供給されます。

Demo モードまたは、User モードを選択するには、Write 関数を使用します。

```
HoltRegWrite(Chan, INPUT_CONTROL, 0); // 0=demo mode (default)
HoltRegWrite(Chan, INPUT_CONTROL, 1); // 1=user mode
```

デモ・ソフトウェアが「Demo mode -default」に設定されている場合、FPGA からデバイスへの BCENA0 および、BCENA1 ピンはソフトウェアで設定できます。「User」モードでは、両方の BC が J4 コネクタの状態に従います。ディフォルトでは、両方の BC の有効化が無効になっています。BC を有効にするには、ユー

ザーはコネクタの BEEnable ピンを接地する必要があります。BCTRIG0 および、BCTRIG1 入力が提供されていますが、現在デモでは使用できません。

コンソール・メニュー・コマンド「0」を使用して、User モードと Demo モードを選択できます。コンソール・メニュー・コマンド「f」は、各デバイスからのすべての信号を読み取り、各信号の状態を表示します。

表 1　メモリ・マップ I/O デバイス制御信号

デバイス信号	ソース	機能	アドレス	ソフトウェア 読込／書込み	用途
BC_TRIG	FPGA->Device	BC トリガ・パルス	0x8000	Read/Write	High、Low の順にセット (1/0)
NMR	FPGA	デバイス・リセット	0x8002	Read/Write	Low、High の順にセット
TEST	FPGA	Test ピン状態	0x8003	Read/Write	High : テスト・モード テスト・モードについては 6130 データシート参照
BCENA	FPGA	BC 有効	0x8004	Read/Write	BC 有効にするには High にセット
MTRUN	FPGA -> Device	MT 有効	0x8009	Read/Write	MT 有効にするには High にセット
RT1ENA	FPGA -> Device	RT1 有効	0x800A	Read/Write	RT1 有効にするには High にセット
RT2ENA	FPGA -> Device	RT2 有効	0x800B	Read/Write	RT2 有効にするには High にセット
TXINHA/TXINHB	J4 Connector	バスの禁止	0x800C	Read only	J4 へのハード・ワイヤ
INPUT_CONTROL	FPGA	Demo モードまたは、ユーザー・モードの設定	0x800E	Read/Write	0=Demo モード(ディフォルト) 1=ユーザー・モード
SPARE_INPUT	J4 Connector	Spare 入力	0x800E	Read only	リード可能な Spare 入力
MTPKTRDY	Device -> FPGA	Monitor Packet 準備完了デバイス出力	0x800F	Read only	オプション・リード
ACTIVE	Device -> FPGA	アクティブ状態	0x8010	Read only	オプション・リード
RT1MC8	Device->FPGA	RT1 モード・コード 8	0x8011	Read only	オプション・リード
RT2MC8	Device->FPGA	RT2 モード・コード 8	0x8012	Read only	オプション・リード
READY	Device->FPGA	Ready 状態を示す	0x8013	Read only	オプション・リード デモ・ソフトウェアで、MR 後にデバイスの準備ができているかどうかを判断するために使用されます

J-TAG & FPGA Boot [J-TAG & FPGA ブート]

ブート SPI フラッシュ・メモリは、出荷時にプログラムされていますが、Xilinx Vivado ツールを使用して、DIGILENT JTAG-HS2 USB プログラミング・ケーブル（付属しません）でユーザーが再プログラム可能です。

FPGA が事前にプログラムされた SPI フラッシュから正常に起動すると、D2 LED が点灯します。再プログラミング後に FPGA コンフィグレーションを再起動するには、SW1 ボタンを押します。通常の動作では、このスイッチは無視できます。

FPGA は 3 つの信号を使用します：コンフィグレーション・オプションの M0、M1 および、M2 入力。M0 と M1 はハードワイヤードなので、この設計では M2 のみ変更できます。しかし、ディフォルトでは、プルダウン抵抗が M2 を Low に保持し、FPGA が SPI フラッシュから起動するようにコンフィグレーションします。再プログラミングするには、J2 ピン 8（図 3 ボード・リファレンスを参照）を High にプルアップし、M2 を High に駆動する必要があります。通常の使用では、ユーザーが Verilog 設計をカスタマイズし、SPI ブートフラッシュを再プログラミングしたい場合以外は、これらの接続をユーザーが変更する必要はありません。再プログラミングの手順については、このドキュメントの後半で説明します。

表 2　FPGA ブート・オプション

機能	M2 - M1 - M0	注記
ダイレクト FPGA プログラミング	0 - 0 - 1	FPGA をプログラムする（揮発性）
SPI ブートフラッシュのプログラム （ディフォルト構成）	1 - 0 - 1	フラッシュから SPI ブートフラッシュ（不揮発性）FPGA ブートをプログラムします

初期化ステータスを提供するために、FPGA によって駆動される 2 つの緑色の LED があります。FPGA の初期化シーケンスが完了すると LED D2 が点灯し、DONE_0 信号を High にアサートします。FPGA がコンフィグレーションリセット状態のときに LED D10 が点灯し、初期化が完了すると消灯します。Xilinx は、FPGA コンフィグレーション・シーケンスに関する完全なテクニカル・ガイドを提供しています。Xilinx UG470 を参照してください。

HI-2130 Parallel Interfaces [HI-2130 パラレル・インターフェイス]

FPGA に実装された 16Bit パラレル・インターフェイスは、両方の HI-2130 デバイスとのインターフェイスに使用されます。アドレスおよび、データ・バスといくつかの他の信号は、両方のデバイス間で共通です。固有のチップセレクトライン nCE0 と nCE1 を使用してそれらの間で選択します。

表 3　FPGA への HI-2130 共通インターフェイス信号

HI-2130	FPGA バンク	プライマリ コネクタ	コメント
Address pins: [A15:A0]	14	-	アドレス・バス
Data bus pins: [D15:D0]	14	-	双方向データ・バス
nRE	14	-	リード・ストローブ(Intel mode)
nWE	14	-	ライト・ストローブ(Intel mode)
MCLK50	34	-	FPGA からのマスター50MHz 入力クロック
MTTCLK	34	-	オプション・クロック、TBD
TTCLK	34	-	オプション・クロック、TBD

表 4　HI-2130 Device 0

HI-2130	FPGA バンク	J4 コネクタ	コメント
nCS0	14		チップ・セレクト 0
nMR0	34		マスター・リセット 0
nIRQ0	34		割り込み出力
TEST0	34		High にアサートされたときのテスト・モード
RT1ENA0	34		FPGA によって RT1 有効に制御
RT2ENA0	34		FPGA によって RT2 有効に制御
RT1MC80	34		RT1MC8 出力、FPGA への入力
RT2MC80	34		RT2MC8 出力、FPGA への入力
CH0BCENAB0 (BCENA)	34	YES	ディフォルトは、プルアップ抵抗により有効です。コネクタによって Low に設定するか、FPGA で上書きします。
CH0BCTRIG (BCTRIG)	34	YES	通常 Low。コネクタでパルスを入力するか、FPGA で上書きします。
MTRUN0	34		FPGA によって制御される入力
READY0, ACTIVE0	34		FPGA への出力、入力。機能は未定
MTPKTRDY0	34		出力、未使用または未定
TXINHA,TXINHB		YES	どちらも、コネクタの CH0INHIBIT0 に接続された入力を禁止します。ディフォルトは有効です。両方のトランスミッタを無効にするには、Low をアサートします

表 5 HI-2130 Device 1

HI-2130	FPGA バンク	J4 コネクタ	コメント
nCS1	14		チップ・セレクト 1
nMR1	34		マスター・リセット 1
nIRQ1	34		割り込み出力
TEST1	34		High にアサートされたときのテスト・モード
RT1ENA1	34		FPGA によって RT1 有効に制御
RT2ENA1	34		FPGA によって RT2 有効に制御
RT1MC81	34		RT1MC8 出力、FPGA への入力
RT2MC81	34		RT2MC8 出力、FPGA への入力
CHOBCENAB1 (BCENA)	34	YES	ディフォルトは、プルアップ抵抗により有効です。コネクタによって Low に設定するか、FPGA で上書きします。
CHOBCTRIG (BCTRIG)	34	YES	通常 Low。コネクタでパルスを入力するか、FPGA で上書きします。
MTRUN1	34		FPGA によって制御される入力
READY1, ACTIVE1	34		FPGA への出力、入力。機能は未定
MTPKTRDY1	34		出力、未使用または未定
TXINHA,TXINHB		YES	どちらも、コネクタの CH1INHIBIT1 に接続された入力を禁止します。ディフォルトは有効です。両方のトランスミッタを無効にするには、Low をアサートします

J4 コネクタ入力ピン：Transmit Inhibit、BCENAx、BCTRIG は、ESF 保護されていますが、DC 定常状態入力電圧は 3.6V を超えてはなりません。

Power supply［電源］

PCIe コネクタからの 3.3aux 電力は、FPGA の 1V および、1V8 電源レールを生成するダブルバック・コンバータに電力を供給します。「power good1」信号を使用して、FPGA GBT PCIe トランシーバ・レールに 1V および、1V2 を供給する 2 つのリニア電圧レギュレータを有効にします。1V8 は、SPI フラッシュ、MEMS オシレータ、および、FPGA バンク 14、15、34 レールに電力を共有する Xilinx の電源シーケンスの推奨を満たすために別の 3V3 電源をオンにするために使用される MOSFET スイッチをオンにするために使用されます。

表 6　電源供給

供給電圧	回路図名	機能	最大電流	消費電流
1V	VCCINT_1V	FPGA ロジック	800mA	200mA
1.8V	VCCAUX_1V8	FPGA 補助機能(JTAG)	800mA	100mA
3.3V	VCCO3V3	FPGA I/O レール、フラッシュ、OSC、その他	1000mA	200mA
3.3V (PCIe conn.)	3V3aux	2×HI-2130	1600mA	1553 メッセージのデューティ・サイクルに依存
1.5V (PCIe conn.)	MGTAVCC1V MGTAVCTT1V2	GBT(PCIe bus)にクリーンな電力を供給します	150mA 150mA	<100mA <100mA

表 7　69Ω に送信される電力とサーマル・ベンチ測定

アクティブチャンネル 0=U7 1=U8	TX デューティ・サイクル %	ICC 電流 (3.3V) 69Ω 負荷	U7 (2130) C° 温度。 自然空冷	U8 (2130) C° 自然空冷
0	100	1.07	70	56
0 & 1	100	1.86	82	78
0 & 1	100	1.86	50 ファンあり	45 ファンあり
0 & 1	100	1.78	81	78
0 & 1	50	1.02	65	63
0	50	.655	59	52
0	10	.34	49	45
0 & 1	10	.42		
0 & 1	IDLE	.26	46	44

注記：ICC は、通常の 75Ω バスカプラーインピーダンスでは低くなります。

表 8　J4 コネクタ・ピン

ピン	名称	説明
1	CHANNEL 0 APOS	MIL-STD-1553 CH0 A+ (BUSA)
2	CHANNEL 0 ANEG	MIL-STD-1553 CH0 A- (nBUSA)
3	Chassis GND	取付ネジ- 他の接続はありません
4	CHANNEL 0 BPOS	MIL-STD-1553 CH0 B+ (BUSB)
5	CHANNEL 0 BNEG	MIL-STD-1553 CH0 B- (nBUSB)
6	Chassis GND	取付ネジ- 他の接続はありません
7	CHANNEL1 APOS	MIL-STD-1553 CH1 A+
8	CHANNEL 1 ANEG	MIL-STD-1553 CH1 A-
9	Chassis GND	取付ネジ- 他の接続はありません
10	CHANNEL 1 BPOS	MIL-STD-1553 CH1 B+
11	CHANNEL 1 BNEG	MIL-STD-1553 CH1 B-
12	Chassis GND	取付ネジ- 他の接続はありません
13	CH0INHBIT0	チャンネル 0 送信禁止。10K プルアップ。2130 Inhibit ピンへの反転直接接続であるインバータに接続します。
14	CH0 BCENAB	チャンネル 0 BC 有効。10K プルアップ
15	CH0 BCTRIG	チャンネル 0 BC トリガ。10K プルアップ
16	SPARE INPUT	未使用
17	CH1INHBIT1	チャンネル 1 送信禁止。10K プルアップ。2130 Inhibit ピンへの反転直接接続であるインバータに接続します。
18	CH1 BCENAB	チャンネル 1 BC 有効。10K プルアップ
19	CH1 BCTRIG	チャンネル 1 BC トリガ。10K プルアップ
20	Logic GND	

表 9　カードテスト・ポイント

テスト ポイント	信号	回路図ページ	機能
TP1	ACTIVE0, Device0 HI-2130	3	BC/RT/SMT コマンド中に High をアサートします
TP2	Device 0 APOS	3	APOS(BUSA)
TP3	Device 0 ANEG	3	ANEG(nBUSA)
TP4	Device 0 BPOS	3	BPOS(BUSB)
TP5	Device B BNEG	3	BNEG(nBUSB)
TP6	ACTIVE1, Device1 HI-2130	4	BC/RT/SMT コマンド中に High をアサートします
TP7	Device 1 APOS	4	APOS(BUSA)
TP8	Device 1 ANEG	4	ANEG(nBUSA)
TP9	Device 1 BPOS	4	BPOS(BUSB)
TP10	Device 1 BNEG	4	BNEG(nBUSB)
TP11	FPGA GPIO	5	ユーザー実装に利用可能
TP12	ADC 12-bit	5	オプションの ADC、ユーザー実装

TP1 と TP6（フィードスルーのみ）は、HI-2130 の ACTIVE 出力で、新しいソフトウェアやその他の設計の変更をチェックするときに役立ちます。ACTIVE は、BC、RT、または、SMT コマンドの実行中に High をアサーとし、ソフトウェアまたは、FPGA 設計の変更をデバッグするときにチェックするための良い出発点として機能します。テスト・ポイント・ボードの配置についてはボード・リファレンスを参照してください。

System Clocks [システム・クロック]

100MHz PCIe システム・クロックは、PCIe コネクタから FPGA に入力されます。このクロック信号は、PCIe に必要な GPT クロック入力ペアに入力され、FPGA ロジックの汎用クロックとしても機能します。

Mini PCIe 低電力モードは、このデザインではサポートされておらず、信号 CLKREQ#は PCIe Mini コネクタのピン 7 に接地されています。

オプションの 100MHz クロックは、U13 MEMS 発信器モジュールによってオンボードで提供されます。子のクロックは現在使用されていません。

GPIO

プライマリコネクタから単一の GPIO ピンは、抵抗で絶縁され、ESD 保護され、FPGA ピンにルーティングされます。これは入力または、出力であり、ユーザー定義です。

ADC 12Bit converter [ADC 12 Bit コンバータ]

ADC 入力はテスト・ポイントで提供されますが、この機能は現在の設計では実装されていません。要求がある場合、ユーザーはこれを実装するオプションがあります。

FPGA Spare Pins [FPGA スペア・ピン]

回路図の Spare pin を参照してください。

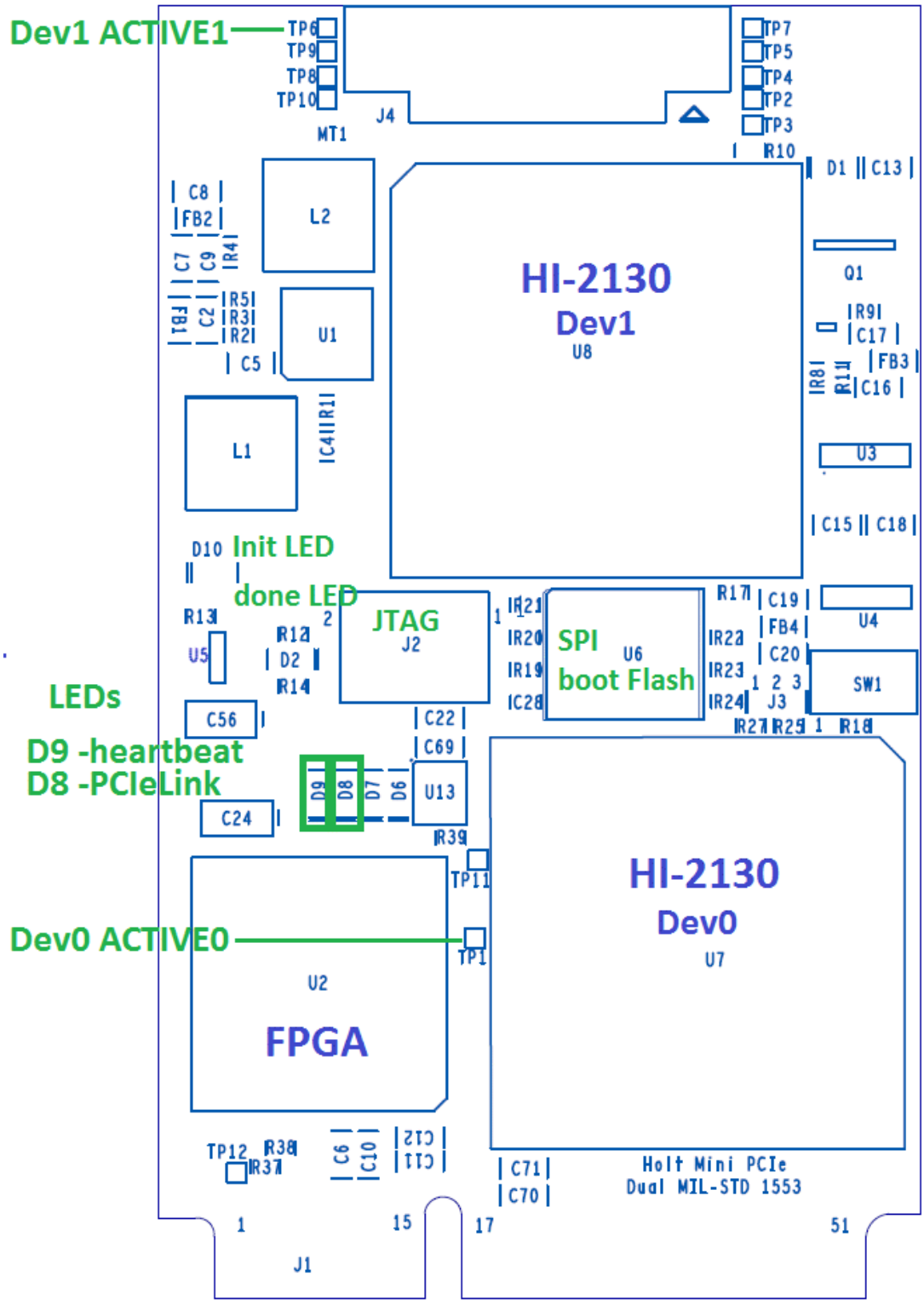


図 5　ボード・リファレンス

Summary [まとめ]

このテクニカル・ガイドでは、Windows 10 環境で Holt の Mini PCIe 2130-1f と-2f カードをデモするために、このリリースを使用する方法について簡単な手順を説明しています。このテクニカル・ガイドでは、Holt Mini PCIe Visual Studio 2019 のプロジェクトとソリューション・ファイルをインストールし、プロジェクトをビルドし、コマンドラインまたは、Visual Studio デバッガ内からデモを実行するために必要な手順をまとめました。また、検証だけでなく、ドライバのインストールに関する小さなワークフローも紹介しました。