HI-3220 ARINC 429

データ・マネージメント・エンジン 16×RX、8×TX

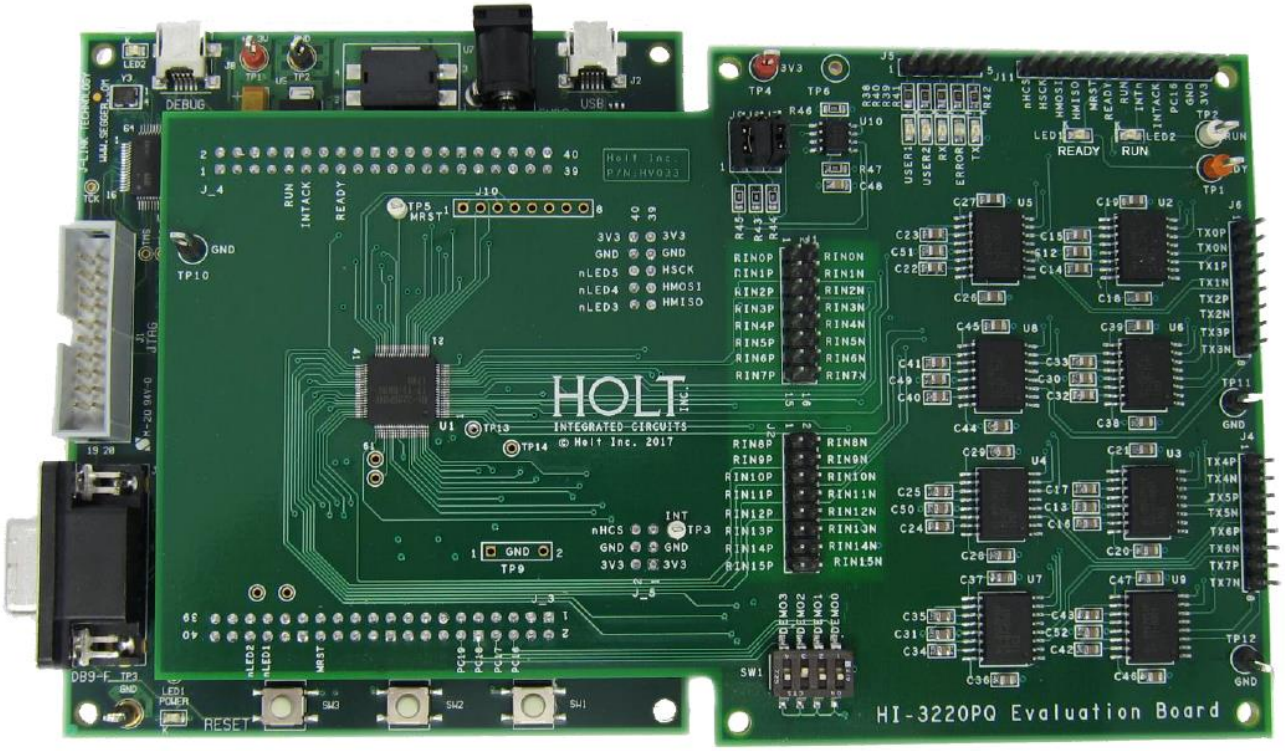
ADK-3220 Application Development Kit ユーザー・ガイド

株式会社ナセル

Introduction【はじめに】

Holt ADK-3220 評価ボードは、Holt の ARINC 429 データ管理エンジンの幅広い機能セットを示し、16 個の ARINC429 レシーバーと 8 個のデジタル・トランスミッターが 1 つの IC に統合されています。HI-3220 は、HI-3200 および、HI-3210 の後継であり、チャンネルが拡張され、ホストの通信効率を高めるように設計された新機能が強化されています。HI-3220 レシーバー・チャンネルは、DO-160G 雷保護を備えています。HI-3200 の最大 SPI レートは 40MHz です。

このガイドでは、ボードをセットアップして実行する方法について説明します。追加のサポート資料と必要なすべてのプロジェクト・ソフトウェアは、付属の Holt USB メモリ・スティックにあります。デモンストレーション・ソフトウェアのバージョンは、マイクロコントローラ・フラッシュに事前にプログラムされています。ボードは、提供されているソフトウェア開発ツールをインストール、または、実行しなくても、箱から出してすぐに動作できます。

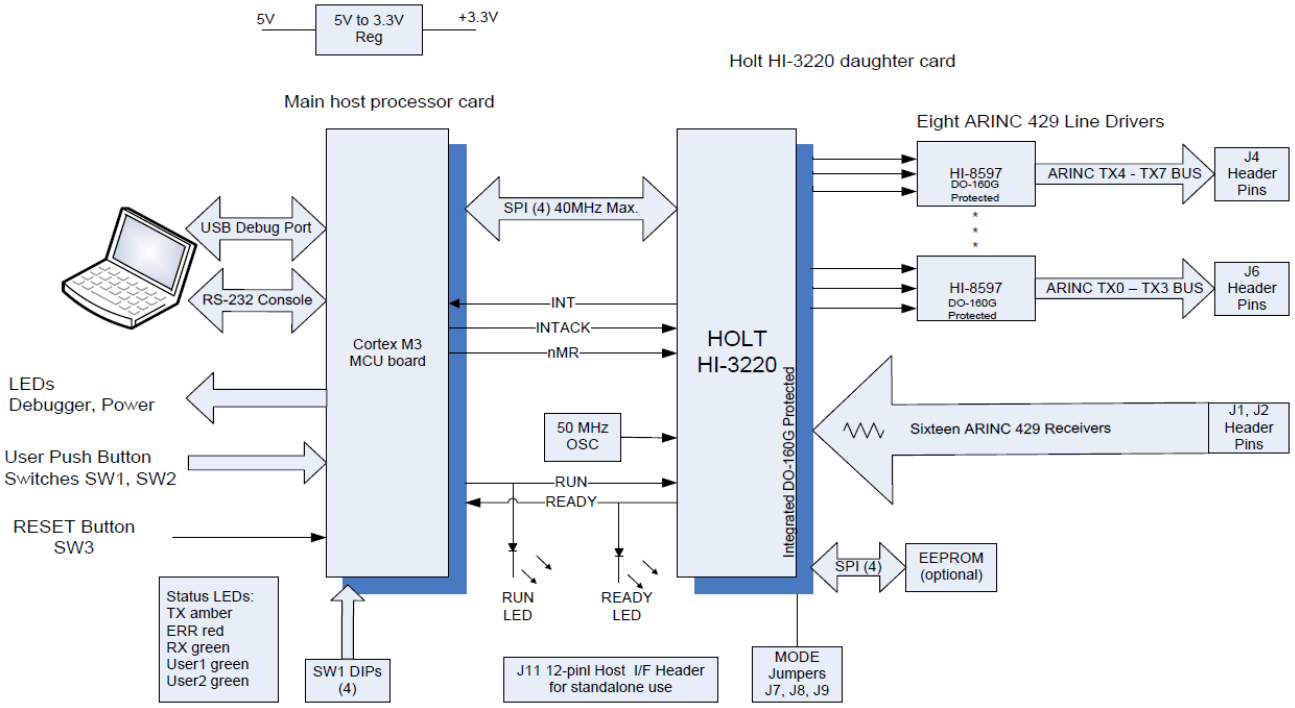


ADK-3220 CTX 評価ボード、ARM Cortex M3 MCU ボードにマウント

Evaluation Kit Contents【評価キット内容】

- 本ユーザー・ガイド
- Holt HI-3220 プロジェクト・ソフトウェア、および、ドキュメント USB メモリ・スティック
- USB メモリ・スティック上の IAR Systems Embedded Workbench® for ARM（32KB KickStart）
- プラグイン 5VDC 電源アダプタ
- USB ミニ・デバッグ・インターフェイス・ケーブル
- 接続されたコンピュータを使用した DB-9M から DB-9F コンソール I/O 用の RS-232 シリアル・ケーブル
- USB シリアル・ドングル・アダプタ
- 2 ボード・アセンブリの構成：
  上段 HI-3220 評価ボード。DIP スイッチはボードの動作を構成します。
  （Microchip/Atmel）ARM Cortex M3 16/32 Bit マイクロプロセッサ、デバッグ・インターフェイス、および、3.3VDC 安定化電源を備えた下段 MCU ボード

Hardware Block Diagram【ハードウェア・ブロック図】



HI-322x Highlighted Feature Differences Compared to HI-3200/3210 Family
【HI-322x は、HI-3200/3210 ファミリと比較した機能の違いハイライト】

- SPI クロック・レートを 40MHz に上昇
- DO-160G レベル 3 雷保護を備えた最大 16 の統合 ARINC 429 受信チャンネル（HI-3200/10 ファミリより 8CH 多い）。統合されたオンチップ・レシーバー（3220、3221、および、3222）を備えた部品で利用可能な耐雷保護
- 最大 8 つの ARINC 429 送信チャンネル（HI-3200/10 ファミリより 4CH 多い）
- 受信の 64 ワード深さの FIFO が 32 から増加
- 32 ワード深さの FIFO を送信
- 送信、および、受信 FIFO カウント・レジスタにより、効率的なホスト・リードが可能
- 送信、および、受信 FIFO しきい値カウント割り込みレジスタ
- 標準の ARINC 429 12.5 KHz、および、100 KHz レートに加えて、ARINC 送信、および、受信 50 KHz レートのサポート
- 特定のライン・ドライバで Hi-Z モードを有効にするための ARINC 送信 TRISTATE 制御ビット
- 送信スケジューラー機能の改善：
  - 簡略化された送信スケジューラーのオペ・コード
  - パケット・タイマー期間とオフセット制御
  - 遅延オペ・コード
  - 送信スケジューラー1 ms プリスケーラ・オプション
  - 送信ウォッチドッグ・タイマー機能
- リセット後にラッチされた Mode0-2 ピンの状態を示すレジスタ 0x807A のモード・ステータス・ビット[7：5]をリードする機能

Push-buttons【プッシュ・ボタン】

プッシュ・ボタン・スイッチ (on main CM3 board)	説明
SW1	チャンネル TX5、TX6、TX7 にテスト・メッセージを送信します TX5 = Low Speed （12.5 KHz） TX6 = Mid Speed （50 KHz） TX7 = High Speed （100 KHz）
SW2	シンプルな SPI シーケンス・テスト・シーケンス – SPI インターフェイスのトラブル・シューティングに便利です。本ドキュメントの最後にある「HI-3220 SPI basics for trouble shooting」を参照してください。

DIP SWITCHES【DIP スイッチ】

スイッチ	ディフォルト	説明
SW 1	OFF	SPI 周波数選択 OFF = 24 MHz ON = 16 MHz 電源投入時にチェックされます。
SW 2	ON	Open = TX ディスクリプタ・テーブルの値をコンソールに記録します
SW 3	ON	ユーザー利用可能
SW 4	ON	ユーザー利用可能

Jumper Settings for EEPROM auto-initialization (optional feature)
【EEPROM 自動初期化のジャンパー設定（オプション機能）】

工場出荷時のディフォルトは Mode 0 です。HI-3220 が各モードで実行するセルフチェックのリストについては、データシートの「リセット、および、起動モード」を参照してください。

MODE	M2 J9	M1 J8	M0 J7	EEPROM 初期化
0	0	0	0	NO
1	0	0	1	YES
2	0	1	0	NO
3	0	1	1	YES
4	1	0	0	NO
5	1	0	1	NO
6	1	1	0	RAM BIST ENA

HI-3220 supply current measurement test points TP13 and TP14
【HI-3220 供給電流測定テスト・ポイント TP13、および、TP14】

HI-3220 の供給電流を測定するには、R35 のゼロオーム抵抗を取り外し、TP13 と TP14 の間に電流計を挿入します。READY が High の場合、LED1 に加算電流があるため、R33 を削除するか、測定電流から LED1 電流（～3.6mA）を減算します。

External Host interface to the HI-3220 daughter board.
【HI-3220 ドーター・ボードへの外部ホスト・インターフェイス】

HI-3220 ドーター・ボードは MCU ボードから分離して、迅速なプロトタイピングの目的でユーザーFPGA、または、MCU ホストに接続できます。すべての重要なインターフェイス信号は、J11-12 ピン・ヘッダー・コネクタで提供されます。デモ・モード・スイッチと LED は含まれていませんが、必要に応じて J3、および、J5 ヘッダー・コネクタからアクセスできます。

J11 ピン	信号	J11 ピン	信号
1	nHCS (SPI)	7	RUN
2	HSCK (SPI)	8	INTn
3	HMOSI (SPI)	9	INTACK
4	HMISO (SPI)	10	SW1 -1 DIP
5	/MRST	11	GND
6	READY	12	3.3 VDD

LED indicators 【LED 表示】

LEDS	機能	典型的な使用法
LED1	READY	READY が High のとき ON（READY ピンに接続）

LED2	RUN	RUN が High のとき ON（RUN ピンに接続）
LED3	RX	ホスト制御：通常 FIFO メッセージをリードするときに ON になります
LED4	USER2	ホスト制御：ユーザー使用可能
LED5	USER1	ホスト制御：メイン・アイドル・ループのみで点滅
LED6	ERROR	ホスト制御：EEPROM 書き込みエラーのようなエラーフラグを立てます
LED7	TX	ホスト制御：通常は送信時に ON

Getting Started 【はじめに】

このユーザー・ガイドの最初のセクションでは、デモ・ボードをセットアップし、Microchip/Atmel ARM Cortex M3MCU に事前にプログラムされている組み込みのデモ・プログラムとユーティリティを実行する方法について説明します。次のセクションでは、IAR EWARMIDE と Holt デモンストレーション・プログラムをインストールする方法について説明します。USB メモリ・スティックに付属のデモ・プログラムは、MCU フラッシュ・メモリにプログラム済みのバージョンと同じです。

Hardware Design Overview 【ハードウェア設計概要】

上段ドーター・ボードと下段 MCU ボードの個別の回路図と部品表については、このドキュメントの最後を参照してください。

取り外し可能なドーター・ボードは、ユーザーが用意した代替 MCU、または、FPGA ボードに接続するために、下段の MCU ボードから分離できます。ボード間ヘッダーは、一般的なプロトタイピング・ボードとの互換性のために 0.1 インチ（2.54 mm）グリッドに配置されています。すべてのホスト・インターフェイス信号は、ボード間ヘッダーを通過します。前のセクションのピンリストについては、外部ホスト・インターフェイス・ヘッダーのピンの説明を参照してください。

下段 ARMCortex M3 ボードは、フラッシュ・プログラム可能な Microchip/Atmel AT91SAM3U-EKMCU に基づいています。4 線シリアル・ペリフェラル・インターフェイス（SPI）は、HI-3220DUT に接続します。UART ベースのシリアル・ポートは RS-232 コンソール I/O（オプション）を提供します。コミットされていない USB2.0 ポートは、将来の拡張に利用できます。ソフトウェアの相互作用には、2 つのプッシュ・ボタンを使用できます。RESET プッシュ・ボタンは ARM マイクロプロセッサをリセットし、ARM マイクロプロセッサは DUT マスター・リセット信号を制御します。

ARM Cortex M3 ボードには、www.segger.com からライセンス供与された「J-LinkOn Board」デバッグ・インターフェイスが含まれており、高価な JTAG デバッグ・ケーブルを購入することなく、すぐに使用できます。キットには、ボードのデバッグ・インターフェイスをコンピュータに接続するためのミニ USB ケーブルが含まれています。（リボンケーブル・コネクタを備えた ARM デバッグ・インターフェイスをすでに所有しているユーザーの場合、ARM 標準の 2x10 デバッグ・コネクタがデバッグ接続を提供します。この場合、下段ボードの下部にあるジャンパーJP2 をはんだ付けして閉じ、「J-Link On board」を無効にする必要があります）。

Initial Set Up 【初期設定】

デモ・プログラムを実行する前に、シリアル COM をサポートする適切な PC が必要です。これにより、コンソール・メニューでコマンドを入力してデモ・プログラムとユーティリティを呼び出すことができます。このデモ・プログラムはすでに CortexM3 MCU にフラッシュされており、そのまま動作します。デモ・プログラムを実行するために IDE フラッシュ・プログラミングは必要ありません。

1. PC には、シリアル（COM）ポートと TeraTerm のような「ターミナル・エミュレーション」プログラムが必要です。ほとんどのコンピュータには RS232 COM ポートがないため、ADK に付属のシリアル－USB アダプタが必要になります。これをコンピュータの USB ポートに接続し、9 ピン・コネクタを ADK ボードに接続します。
2. Windows 2000、または、Windows XP を使用している場合は、ターミナル・エミュレーションにハイパーターミナルを使用できます。[スタート]、[すべてのプログラム]の順にクリックして、ハイパーターミナルを開きます。 [Windows アクセサリ]、[通信プログラム]グループの順にクリックします。ハイパーターミナルをダブルクリックして実行します。次の段落をスキップします。

   Vista や Windows 7 または Windows 10 を使用している場合

   ハイパーターミナルは、これらのバージョンの Windows には含まれていません。Holt から提供されている teraterm-4.71.exe インストーラ・プログラムを実行して、無料のオープンソース・ターミナル・エミュレーション・プログラムである TeraTerm 4.71 をインストールします。著作権表示が保持されている場合に限り、再配布が許可されることを示す使用許諾契約に同意します。通知は、[Help]、[About TeraTerm]の順にクリックすると、TeraTerm ウィンドウから表示できます。インストールを続行しています…
   - ディフォルトのインストール先を受け入れて、[次へ]をクリックします。
   - コンポーネントの選択画面で、[追加プラグイン= TTXResizeMenu]を除くすべてのオプションの選択を解除し、[次へ]をクリックします。
   - インストールされている言語を選択し、[次へ]をクリックします。
   - ディフォルトのスタートメニューフォルダを受け入れ、[次へ]をクリックします。
   - 必要なショートカットを選択し、[次へ]をクリックします。
   - インストールの準備完了画面で、[インストール]をクリックします。

   TeraTerm プログラムを実行します。[New Connection]画面で、（x）シリアルを選択し、選択した COM ポートを選択します。[Setup]、[Serial Port]の順にクリックして、シリアル・ポート設定ウィンドウを開きます。次の設定を選択します：ボーレート：115,200、データ：8 ビット、パリティ：なし、停止：1 ビット、フロー制御：なし。

3. 付属の 5VDC 電源を接続し、ケーブルを下段回路基板の電源入力ジャックに接続します。TeraTerm が実行され、正しくコンフィグレーションされている場合、以下のコマンド・メニューがコンソール・ウィンドウに表示されます。このメニューは、ボードの電源が投入されたとき、または、RESET プッシュ・ボタンが押されたときに表示されます。評価ボードとの TeraTerm の通信が正しいことを確認した後、[Setup]、[Save Setup]の順にクリックして、ターミナルの設定を保存できます。

4. 電源投入時に、すべての LED が短時間点滅します。READY LED はオンのままで、RUNLED はオフであ

る必要があります。プログラムがメイン・アイドル・ループにあるとき、緑色の USER1LED が 1.2Hz で点滅します。

Getting Started with Demos【デモの開始】

ボードの電源を入れると、コマンドのメニューがコンソールに表示されます。

```
-------------------------------------------------
Holt Integrated Circuits HI-3220 Demo Program Ver. 1.0
Compiled: Jan 26 2018 08:27:43
-------------------------------------------------
DIP switches: 01
SPI 24MHz
Waiting for READY high after reset...
MSR = 20 Expected value is 20
READY OK
MODE Pins read: 0
MODES 1 or 3 auto-initializes from EEPROM - must have been programmed first or will fault

*************************************************
Holt Integrated Circuits HI-3220 Demo Program Ver. 1.0
Compiled: Jan 26 2018 08:27:42
*************************************************

------------- ADK-3220 DEMOS --------------
Press '0' - RX RAM RECEPTION LOOPBACK
Press '1' - RX RAM RECEPTION NO LOOPBACK
Press '2' - RX FIFO and TX FIFO Direct
Press '3' - TX FIFO Direct Immediate
Press '4' - TX0 Simple Transmit Scheduler-continuous
Press '5' - RX0 -> TX0 Repeater
Press '6' - RX0 + RX1 -> TX0 Concentrator
Press '7' - SUB MENU: 1-One Time Transmission, 2-PacketTimer, 3-RX&TXDemo
4- Transmit using all 8 schedulers
Press '8' - Enables RX0-RX15 receivers, use with command '9'
Press '9' - Reads and Displays all RX channels FIFOs
Press 'p' - Demo configuration for EEPROM demo
Press 'P' - Programs auto-initization EEPROM
Press 'k' - Transmit a message on TX2 and TX3

------------ UTILITIES --------------
Press 'a' - Enter Watch Window Address
Press 'c' - Clears most device memory 0-0x8088
Press 'd' - Display selected RX RAM Memory blocks 0-0x3FFF, q=quit
Press 'e' - Display Receiver FIFO look-up label and interrupt look-up tables Memory
Press 'h' - Display this help menu
Press 'i' - Set interrupts SUB-MENU
Press 'I' - Set Interrupt regs for testing
Press 'l' - Toggle the LOOPBACK register from 00 <-> 0xFF
Press 'L' - Demonstrate setting and clearing Label acceptance table memory
Press 'm' - Write byte to specified Address
Press 'M' - Write byte(s) to memory Range
Press 'O and o' - WDT Demo, init and refresh
Press 'r' - Toggle RUN pin state
Press 'R' - 3220 Software Reset
Press 't' - Display ALL Transmit Tables 0x4000-0x67FF, q=quit
Press 'w' - Display Device Memory from MAP
Press 'x' - Transmits number of mesgs on given channel
Press 'Space Bar' - Display Selected Registers

=========================================================
Press 'H' for this help menu, or press any valid menu key. >>
```

Demonstration Programs and Utilities【デモ・プログラムとユーティリティ】

このドキュメントで説明されているデモとユーティリティを最大限に活用できるように、最初に HI-3220 データシートを確認してデバイスの基本を理解することを強くお勧めします。

コンソールに入力されたほぼすべてのメニュー・コマンドは、HI-3220 でデータをリード／ライトするために SPI オペ・コードを発行するように ARM MCU に指示します。ホストは、メモリ空間 0x8000～0x8087 にある他の関連システム・レジスタとともに SPI インターフェイスを使用してさまざまな制御レジスタに書き込むことにより、HI-3220 をコンフィグレーションします。マスター・リセット後に READY ピンが High にアサートされると、抵抗、または、メモリ位置へのアクセスが有効になります。メッセージ・データ、送信スケジューラー、RX FIFO イネーブル、RX 割り込みイネーブル・テーブルは、メモリ位置 0x0000～0x8000 に存在します。データシートの 11 ページにある HI-3220 メモリ・マップを参照してください。ホストがデバイスをコンフィグレーションし、必要なデータ・テーブルをロードすると、RUN ピンが High に設定されている場合、デバイスはデータの送受信が可能になります。HI-3220 の RUN 入力ピンは MCU GPIO ピンによって制御され、「r」コマンドによって切換えられます。RUN ピンが High の場合、緑色の RUN ステータス LED が点灯します。RUN ピンについては後で詳しく説明します。

デモに加えて、レジスタ、または、メモリ範囲の表示と変更、および、RX 割り込みと RXFIFO イネーブル MAP テーブルのシングル・ビットの設定／クリアを可能にするさまざまな SPI ユーティリティがあります。これらは非常に便利なデバッグ・ユーティリティであり、従来の IDE デバッガは SPI インターフェイスを介してメモリにアクセスできないために提供されています。注：ディフォルトでは、すべての SPI ユーティリティは、値の前に従来の「0x」を表示せずに、アドレス、または、データの表示に 16 進形式を使用します。

Demonstrations and Examples【デモと例】

デモ・プログラム 0～6 の最初のセットは、3、4、および、5 ページのデータシートに示されている図 1～5 の例によってモデル化されています。デモの説明では、ドキュメントは通常、3 列目に示されているこれらの方法を参照します。デモでは、データを送受信するためのこれらの方法の 1 つ、または、2 つを示します。以下にリストされている主なデモは、HI-3220 の重要な機能を示しています。コンソール・メニューに番号を入力すると、リストされている対応するデモが実行されます。

デモ番号	デモ説明	注記
Example 0	オンチップ RAM 受信 – ループバック	ダイレクト FIFO 方式によるトランスミッタも使用します。
Example 1	オンチップ RAM 受信	ダイレクト FIFO 方式によるトランスミッタも使用します。
Example 2	オンチップ FIFO によるデータ受信	RX FIFO 受信
Example 3	TX FIFO MCU から直接データを送信	FIFO ローディングによる直接送信
Example 4	TX シンプルデータ伝送のオンチップ・スケジューラ ディスクリプタ・テーブル必要	送信スケジューラー
Example 5	RX0 –> TX0 リピーター	RX0 で受信したメッセージは、TX0 で

		再送されます。
Example 6	RX0 + RX1 -> TX0 コンセントレーター	RX0 と RX1 で受信したメッセージは、両方とも TX0 で再送信されます
Example 7	サブメニューコマンド	以下の説明を参照
Example 8	これは、他のいくつかのデモで使用されている 16CH のレシーバーすべてを有効にするデモユーティリティ	16CH すべてのレシーバーを受信可能にします。
Example 9	これは、他のデモのいくつかで使用されているコンソールに任意の RX FIFO メッセージをリードし、表示する Demo ユーティリティ	

下段のボードの 2 つのプッシュ・ボタン・スイッチは 2 つのユーティリティ機能を提供します。

SW1 プッシュ・ボタン - MCR と TX5、TX6、および、TX7 制御レジスタをコンフィグレーションし、これら 3 つのトランスミッタ・チャンネルで単一のメッセージを送信します。TX5 は Low Speed で送信し、TX6 は 50K の速度で送信し、TX は High Speed で送信します。緑色の TXLED が短時間点滅します。

SW2 プッシュ・ボタン - HI-3220 と通信するための適切な SPI シグナリングを示す 4 つの重要な SPI シーケンスを生成します。SPI シグナリングの詳細については、このドキュメントの最後を参照してください。

リード・レジスタ MSR(0x02)は値 0x20 を返し、MAP レジスタに 0x8000 をライトします。

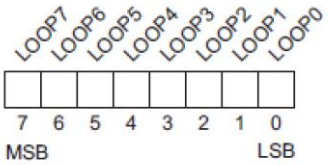
MAP アドレス位置(0x8000)に 0xC0 をライトし、MAP 位置のリード・バイトで 0xC0 を返します。

Demo Descriptions【デモの説明】

Demo 0:
「0」キーを押し RAM 受信 - ループバック

Demo 0 は、LOOPBACK レジスタを 0xFF でプログラムするため、すべての RX チャンネルが LOOPBACK モードで設定されます。メッセージは、ラベル値別に編成された受信データ・メモリにロードされます。各チャンネル 0～15 には、ラベル値で編成されたメッセージごとに 256 の 4 Byte の場所があります。TX0-TX7 送信機は、送信 FIFO にメッセージ・データをロードすることにより、直接送信方式を使用してテスト・データを送信するために使用されます。すべてのトランスミッタは High Speed でコンフィグレーションされており、RX0 と RX1 で Parity ON が有効になっています。16 個のレシーバーすべてが同様にコンフィグレーションされています。最初に、送信データはラベル 00 とデータ 00 45 67 に設定されます。ARINC 429 データは、直接送信方式（例 3）を使用して 8 つのチャンネルに送信されます。各トランスミッタ FIFO がロードされた後、ラベルと Byte 1 が 1 つインクリメントされます。LOOPBACK モードでは、以下に示すように、各トランスミッタは 2 つの連続するレシーバーにマップされます。

SELF-TEST LOOPBACK
(Address 0x8049)



Bit	Name	R/W	Default	Description
7	LOOP7	R/W	0	This bit is set to "1" to loop-back transmit channel 7 to receivers 14 and 15
6	LOOP6	R/W	0	This bit is set to "1" to loop-back transmit channel 6 to receivers 12 and 13
5	LOOP5	R/W	0	This bit is set to "1" to loop-back transmit channel 5 to receivers 10 and 11
4	LOOP4	R/W	0	This bit is set to "1" to loop-back transmit channel 4 to receivers 8 and 9
3	LOOP3	R/W	0	This bit is set to "1" to loop-back transmit channel 3 to receivers 6 and 7
2	LOOP2	R/W	0	This bit is set to "1" to loop-back transmit channel 2 to receivers 4 and 5
1	LOOP1	R/W	0	This bit is set to "1" to loop-back transmit channel 1 to receivers 2 and 3
0	LOOP0	R/W	0	This bit is set to "1" to loop-back transmit channel 0 to receivers 0 and 1

「0」キーを押すたびに、この出力がコンソールに表示されます。デバッグ情報については、各チャンネル 0～15 のメッセージの前に 3 つの割り込みフラグとシステム・レジスタが表示されます。

```
Quit> 0
 ARINC RAM Reception LOOPBACK
Loop Back Mode

 PIR REG 0x8004 = 0x04
 RPIRl REG 0x8006 = 0xFF
 RPIRH REG 0x8008 = 0xFF
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
 MAP:8000 C0 C0 30 00 00 00 00 00 00 00 00 00 00 FF FF FF
 MAP:8010 00 00 01 01 02 02 03 03 04 04 05 05 06 06 07 07
 MAP:8020 80 80 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0     (RX Control Registers)
 MAP:8030 80 80 80 80 80 80 80 80 00 00 00 00 00 00 00 00     (TX Control Registers)
 MAP:8040 00 00 00 00 00 00 00 00 00 FF F8 FF FF 00 00 00     (LOOPBACK Reg 0x8049=0xFF)
 MAP:8050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:8060 00 00 00 00 00 00 00 00 01 01 01 01 01 01 01 01
 MAP:8070 01 01 01 01 01 01 01 01 00 00 00 00 00 00 FF 00

 ** RX messages by polling **
 RX-CH0: Label:00 00 45 67
```

```
RX-CH1: Label:00 00 45 67
RX-CH2: Label:01 01 45 E7
RX-CH3: Label:01 01 45 E7
RX-CH4: Label:02 02 45 E7
RX-CH5: Label:02 02 45 E7
RX-CH6: Label:03 03 45 E7
RX-CH7: Label:03 03 45 E7
RX-CH8: Label:04 04 45 E7
RX-CH9: Label:04 04 45 E7
RX-CH10: Label:05 05 45 E7
RX-CH11: Label:05 05 45 E7
RX-CH12: Label:06 06 45 E7
RX-CH13: Label:06 06 45 E7
RX-CH14: Label:07 07 45 E7
RX-CH15: Label:07 07 45 E7
Quit>
```

オプションで、RAM アクセスによってレシーバー・データをリードする代わりに、FIFO からデータをリードすることができます。これは Demo 2 で示されています。

ディフォルトでは、C プログラムはマクロ INTERRUPT_MESG_ENA を使用してポーリング用にコンパイルされます。 メッセージをリードするための簡単な割り込みメソッドは、3220_initialization.h ファイルでこのマクロを 1 に変更することで提供されます。これにより、メッセージがリードされ、割り込みハンドラーに表示されます。

Demo 1:

「1」キーを押し RAM 受信 - 通常モード

Demo 1 は、LOOPBACK 機能がオフになっていることを除いて Demo 0 と同じデモです（LOOPBACK レジスタ 0x8049 = 0x00）。オシロスコープを使用して、J4、または、J6 ヘッダー・ピンの ARINC 送信を表示できます。たとえば、J6 ピン 1 は TX0AOUT で、ピン 2 は TX0BOUT です。トランスミッタとレシーバーの間にジャンパー線が接続されていない場合、レジスタ・データのみがコンソールに表示されます。オプションで、外部 ARINC 429 High Speed ソースをヘッダー・コネクタのレシーバー入力 CH0～15 のいずれかに接続します。RX チャンネル 0～7 は J1 で使用でき、チャンネル 8～17 は J2 で使用できます。
受信チャンネルが受信する単一の送信チャンネルを設定するには、クリップ・リードのペアを使用して以下を接続します。TX0 出力を RX0 入力に接続し、「1」キーが押されたときにデータがコンソールに表示されることを確認します。オプションで、より多くの CH、または、外部 ARINC 429 ソースを接続し、もう一度「1」キーを押してデータを表示します。

J6-1 TX0AOUT	Jumper to	J1-1 RIN0P
J6-2 TX0BOUT	Jumper to	J1-2 RIN0N

```
> 1
 ARINC RAM Reception NO LOOPBACK
Normal Mode

  PIR REG 0x8004 = 0x04
  RPIRI REG 0x8006 = 0x02
  RPIRH REG 0x8008 = 0x75
            0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
  MAP:8000 C0 C0 30 00 00 00 00 00 00 00 00 00 00 02 75 FF
  MAP:8010 00 03 00 00 00 00 00 00 04 00 05 00 06 06 07 00
  MAP:8020 80 80 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0
  MAP:8030 80 80 80 80 80 80 80 80 00 00 00 00 00 00 00 00
  MAP:8040 00 00 00 00 00 00 00 00 00 00 F8 FF FF 00 00 00
  MAP:8050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
  MAP:8060 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00
  MAP:8070 01 00 01 00 01 01 01 00 00 00 00 00 00 00 FF 00

  ** RX messages by polling **
RX-CH0: Label:00 00 45 67
Quit>
```

Demo 2:
「2」キーを押し、RX FIFO 受信と TX 直接送信

これは、16 個のレシーバーすべてでのレシーバーFIFO 受信を示しています。
各送信チャンネル TX0～TX7 は、32 個のメッセージを送信してメッセージ・データを RX0～RX15 レシーバーに提供します。LOOPBACK は Demo 0、および、1 でデモンストレーションされたため、このデモは外部クリップ・リード接続に依存しています。各チャンネルには、クリップ・リードのペアが必要です。たとえば、TX0 を RX0 に、TX7 を RX15 に接続するには、次の接続を行います。

J6-1 TX0AOUT	Jumper to	J1-1 RIN0P
J6-2 TX0BOUT	Jumper to	J1-2 RIN0N
J4-7 TX7AOUT	Jumper to	J2-15 RIN15P
J4-8 TX7BOUT	Jumper to	J2-16 RIN15N

データ・パターン関数は、送信 FIFO にロードされる 32 個のメッセージに対して次のようなパターンを作成します。最初の 2 Byte（label と Byte1）にはラベルがロードされます。Byte 2 には、値 0x00 から始まる増分データがロードされます。Byte 3 は Byte 2 を補完するものです。
各レシーバー・チャンネルは、64 メッセージの深さの FIFO を備えています。ソフトウェアの RXFIFO リード・セクションは、最初に各 RX FIFO チャンネルのメッセージ数をリードし、次に関数を呼び出してその数のメッセージをリードします。これを 16 のレシーバー・チャンネルすべてに対して繰り返します。正確な詳細については、デモ・コードを参照してください。

接続が確立され、「2」キーが押されると、コンソールの表示は次のようになります。

```
RX 0:0   00 00 00 FF
RX 0:1   00 00 01 FE
       *
       *
       *
RX 0:30  00 00 1E E1
RX 0:31  00 00 1F E0

RX 15:0  07 07 00 FF
RX 15:1  07 07 01 FE
       *
       *
       *
RX 15:30  07 07 1E E1
RX 15:31  07 07 1F E0
```

Demo 3:
「3」キーを押し、CPU による直接送信を使用して TX2、および、TX3 でメッセージを送信

この方法は、前の Demo 0～2 ですでに示されています。ダイレクト方式による送信は、送信 FIFO にデータがロードされ、RUN ピンが High に設定されている場合です。これにより、FIFO が空になるまで、その

チャンネルの FIFO 内のデータが送信されます。「SW3」プッシュ・ボタンを押すと、この方法を使用して 3 つのメッセージが送信され、TX5、TX6、および、TX7 にそれぞれ 1 つのメッセージが送信されます。Demo k もこの方法を使用して、TX2 と TX3 でメッセージを送信します。

Transmit Scheduler demos 【送信スケジューラーのデモ】

次の一連のデモでは、HI-3220 の送信スケジューラー機能を使用してさまざまな機能を実行します。各送信チャンネル TX0～TX7 には、アドレス 0x4000 から始まる送信コントローラーとサポート・ディスクリプタ・テーブル・スペースがメモリ内にあります。チャンネル 0、および、1 の場合、128、または、256 のメッセージ・ディスクリプタをディスクリプタ・メモリ空間にロードできます。チャンネル 2～7 は 128 メッセージに固定されています。各ディスクリプタは、オペ・コード、タイマー値、および、メッセージ・データ用の 8 Byte でコンフィグレーションされます。送信スケジューラーの動作の詳細については、データシートを参照してください。ディスクリプタが初期化され、制御レジスタ・ビットが適切に設定されると、RUN ピンが High に設定されたときにスケジューラーが実行されます。スケジューラーを停止するには、RUN ピンをローに設定します。
これらのデモでは、コマンド「r」を使用して RUN ピンを High、または、Low に切換えます。

各送信スケジューラーを使用できるいくつかの例：

1. 最大 128、または、256 の即時メッセージを送信します。
2. メッセージが任意のレシーバー・チャンネルから ARINC ラベル値で受信された場合、メッセージを条件付き、または、無条件で自動的に送信します。
3. 上記の項目 2 と同じメッセージを送信しますが、オプションで ARINC SDI ビット（ARINC Bit 9、および、10）を再挿入します。
4. 自動初期化 EEPROM がプログラムされた後の自律動作。
5. テーブルの終わりに達するか、スケジューラーがシーケンスの終わりの値 0x00 に遭遇するまで、プログラムされたメッセージ・ディスクリプタの数をワンショット、または、継続的に送信します。
6. 送信スケジューラー・メッセージの自動ルーティング機能により、デバイスをデータ・コンセントレータ、または、リピーターとしてコンフィグレーションできます。

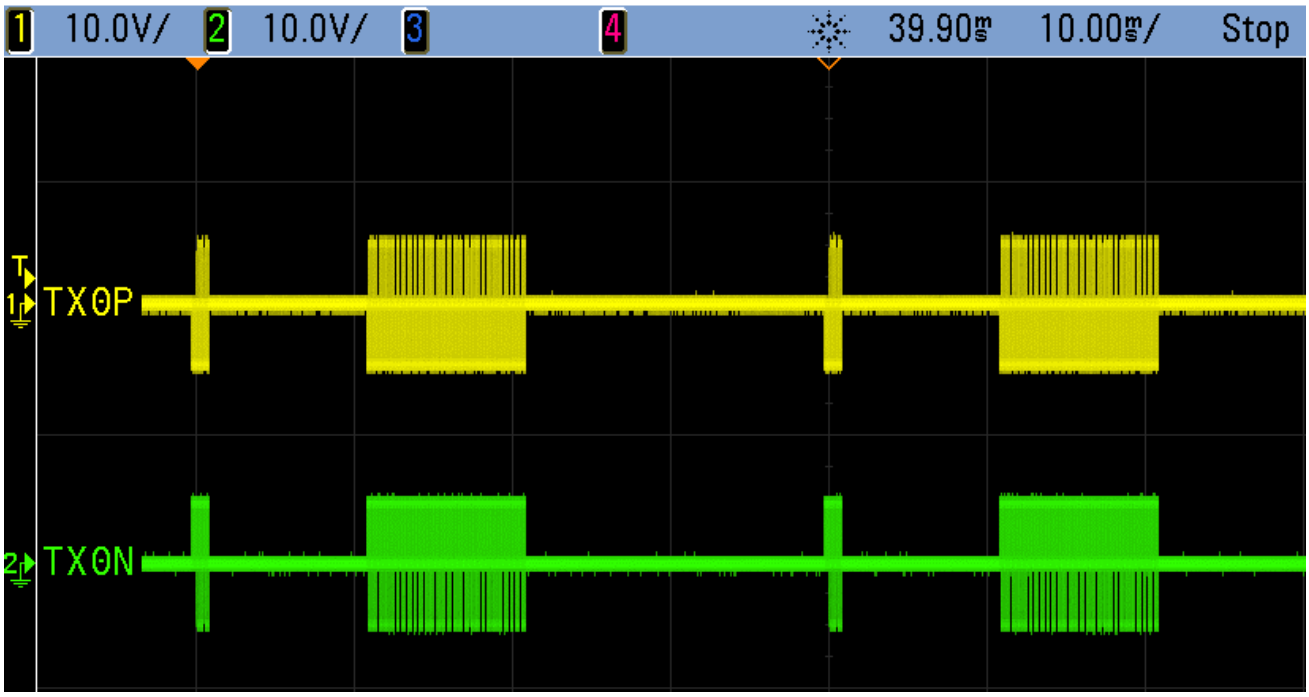
Demo 4:
「4」キーを押し、TX0 単純なスケジュール ― 連続

このデモは、送信 TX0 ディスクリプタ・テーブルをイミディエイト・オペ・コード付きの 31 メッセージで初期化します。3 番目のメッセージが送信された後、残りのメッセージが送信される前に、10 ms の遅延オペ・コードが送信を遅延させます。31 個のメッセージはすべて、インクリメントする byte 3、ラベル 00、byte 1：0x00、byte2 インクリメント・データ、および、byte 3 = 0x04 を除いて同じです。
各メッセージがディスクリプタ・テーブルにロードされた後、ラベルは次のディスクリプタのためにインクリメントされます。プログラムが 32 個のディスクリプタをすべてロードした後、シーケンス終了オペ・コードが 0x00 の次のアドレスにライトされます。繰り返しレートは 40 ms に設定されています。

「4」を押すと、デモはすぐにメイン・コンソール・メニューに戻ります。
「r」を押して RUN ピンを High に設定し、スケジューラーを開始します。最初の 3 つのメッセージが送信され、その後 10 ms の遅延が続き、残りのメッセージが送信されます。これは 40 ms ごとに発生します。
「r」をもう一度押すと、RUN がローになり、送信が停止します。「r」をもう一度押すと、RUN ピンが再

び High に設定され、送信が再開されます。

オシロスコープを使用して、J6 ピン 1、および、2 の ARINC TX0 チャネル送信を表示します。



データをリードバックし、FIFO 受信方式を使用してコンソールに表示するには、最初に次のように TX0 トランスミッタ出力を RX0 レシーバー入力に接続します。

J6-1 TXOAOUT	Jumper to	J2-1 RINOP
J6-2 TXOBOUT	Jumper to	J2-2 RINON

「8」を押してレシーバー0～15 を初期化し、「r」を押して送信を開始します。 もう一度「r」を押すと、送信が停止します。

「9」を押すと、コンソールに 64 個のメッセージが表示されます。 これがレシーバーFIFO の深さであるため、64 個のメッセージがあります。おそらく、1 回目と 2 回目の「r」キーを押す間にいくつかの 64 メッセージ・グループが送信され、FIFO がオーバーフローしました。受信 FIFO がいっぱいになると、後続のメッセージが FIFO にロードされ、保存されている最も古いメッセージが上書きされます。

RUN を High のままにして、ランダムに「9」を押して FIFO メッセージを表示することもできます。この場合、最初に表示されるメッセージがディスクリプタ・テーブルの最初のメッセージになる可能性は低くなります。

コンソール・メニューのコマンド「8」および「9」をいつでも使用して、RX レシーバーをコンフィグレーションし、RX FIFO データを表示できます。

リピーター・デモ

「5」キーを押し、RX0 -> TX0 リピーター

このデモでは、TX0 スケジューラ・テーブルに単一のディスクリプタを設定してリピーター機能を実行し、制御をコンソールのメイン・ループに戻します。「r」を押すと、RUN ピンが High に設定され、スケジューラーが開始されます。レシーバー0 で受信したメッセージは、ホストの介入なしに TX0 でメッセージを自動的に再送信します。オプションで、ホストは必要に応じて RAM 受信、または、FIFO 受信を使用してメッセージを受信できます。

ディスクリプタは、RAM メモリ（0x0000）の RX0 チャンネル、ラベル 0 メッセージを参照するようにコンフィグレーションされています。

この例では、Byte 1 に IDC オペ・コードを使用します。再送信されたメッセージのラベル値は Byte 5 に格納されます。Byte 2、3、および、4 はインデックスが付けられ、ラベル 0x00 メッセージ・データが格納されている RX0 チャンネルから取得されますが、実際には最初の Byte はこの場所のステータス・バイトです。3220Demo.c の Demo 5 コードを確認して、C コードのコンフィグレーション手順を確認します。

インデックス付き条件付きオペ・コードは、受信データ・ステータス・バイトで NEWTX0～NEWTX3 ビットを使用します。このステータス・バイトは、受信したメッセージごとにロードされる受信データ・メモリ（データシートの 19 ページを参照）の最初のバイトです。メッセージを受信すると、このステータス・バイト・レジスタのすべてのビットが設定されます。これは、新しいメッセージが受信されたときにスケジューラーが検出する方法です。ディスクリプタがこのブロックからのデータのいずれかを参照する場合、対応する NEWTXx ビットはトランスミッタ・チャンネル（TX0-TX7）に応じてクリアされます。

以下に示す 8 Byte でコンフィグレーションされる単一のトランスミッタ・ディスクリプタは、TX0 ディスクリプタ・テーブルの最初の場所である開始場所 0x4000 にロードされます。これらの値は配列に設定されているため、実験用に C コードで簡単に変更できます。

ディスクリプタ Bytes 1-8

Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
Op Code	RX Label	PTP/Delay	PTO	ARINC Label	ARINC Byte1	ARINC Byte3	ARINC Byte3
IDC (0101)	0x02	0x00	0x00	0x02	N/A	N/A	N/A

リピーターの場合、PRESCALE ビット 6 が TX0 制御レジスタに設定されます。これにより、繰り返しレートが 1 ms に設定され、RX0 レシーバーにメッセージが到着してからスケジューラーが TXO にメッセージを再送信するまでの待ち時間が最小限に抑えられます。

メモリ内のディスクリプタ・バイトを表示するには、「a」コマンド・ユーティリティを使用して、アドレス 4000 を入力します。ディスクリプタ・メモリは、次のデータとともに表示されます（部分的にのみ表示されています）。

```
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
MAP:4000 50 02 00 00 02 00 00 00 00 00 00 00 00 00 00 00
                                  ^--- EOS
```

例：

RX0 受信メッセージ： 00 02 34 56
TX0 リピート・メッセージ： 00 02 34 56

送信されたラベルを変更するには、ラベル（Byte 5）をソフトウェアで任意の値に変更するだけです。デモ・プログラムでは、ディスクリプタ・バイトは 8 Byte arrays[]に格納されます。実験は、配列内のディスクリプタ・バイト値を変更してプロジェクトをリビルドするか、コンソールに入力された任意のアドレスにバイトを書き込むことができる SPI ユーティリティ・コマンド「m」を使用してバイトを変更することによって実行できます。SPI ユーティリティ・コマンド「a」を使用して結果を表示し、開始アドレスを設定し、アドレスのブロックと対応するデータ値を表示します。
このデモでは、2 つのクリップ・リードを接続して、TX2 出力と RX0 入力の間に外部接続を確立します（前のデモと同様）。「k」キーを押すと、メッセージが TX2 で送信されるため、これが、リピーターが TX0 で再送信するソース・メッセージ、または、入力メッセージになります。

手順：

「5」を押して、リピーターのデモを実行します。
「r」を押してスケジューラーを実行します（RUN high）
「k」を押して、TX2、および、TX3 でテスト・メッセージを送信します（このデモでは TX3 は無視されます）。
「9」を押してメッセージを表示します。

繰り返し送信を確認するには、TX0 出力ピンから RX3 チャンネル入力にクリップ・リードの別のペアを接続します。これにより、コンソールの TX0 でリードバックおよび再送信されたメッセージを表示できます。

結果：

> 5
リピーター・デモの場合は「6」を押します
「8」を押してレシーバーを有効にします
「r」を押して、スケジューラーを実行する RUN を High に設定します
「k」を押して TX2 TX3 テスト・メッセージを送信します – TX2 でメッセージ 0x02 0x02 0x34 0x56 を送信
「9」を押して FIFO データをリードし、コンソールに表示します

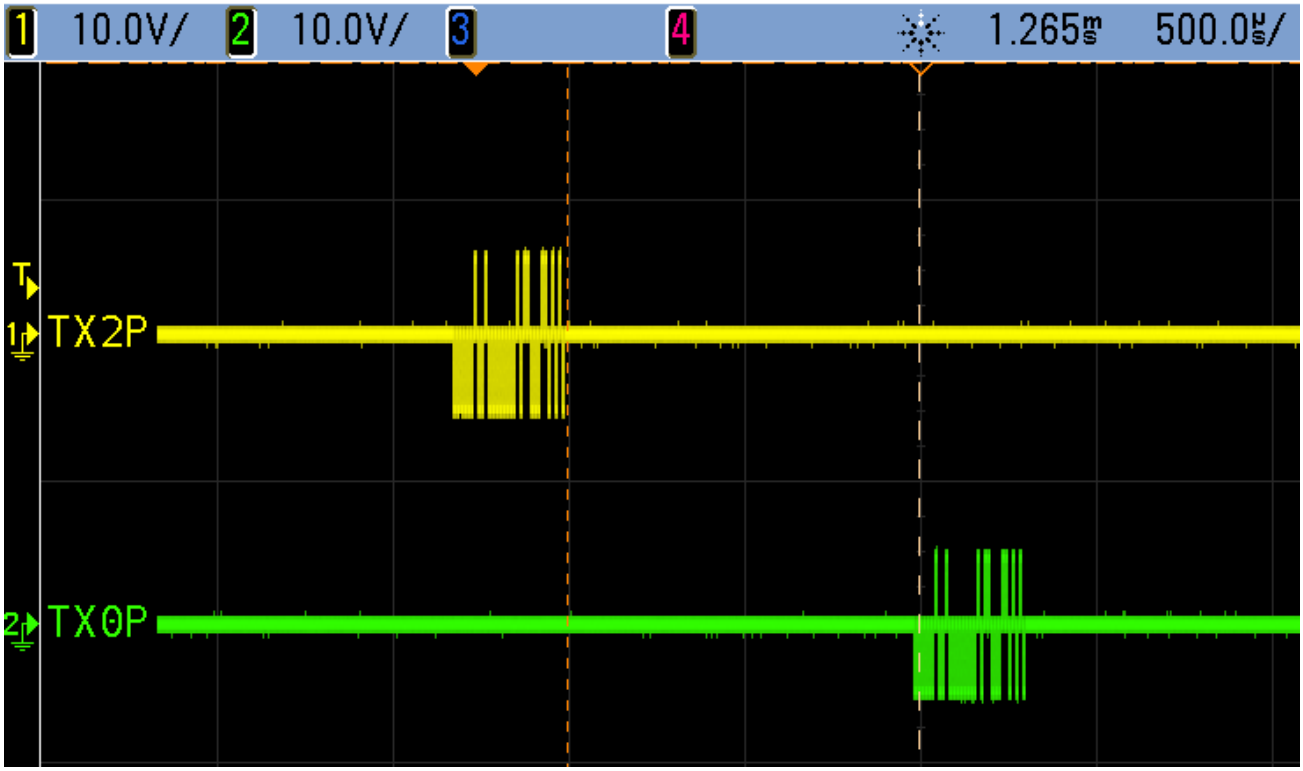
表示結果：

```
mesgs: 1
RX 0:0 02 02 34 56
mesgs: 1
RX 3:0 02 02 34 56 Elapsed time 0.007210 >
```

「k」が押されるたびに、メッセージが TX2 で送信され、これが RX0 によって受信され、スケジューラーが TX0 でメッセージを再送信します。

以下のスコープ・キャプチャは、上部に TX2 送信を示し、下部に TX0 で再送信されたメッセージを示しています。スケジューラーの 1 ms タイマーはメッセージに対して非同期であるため、この場合の最悪の場

合の遅延は 1 ms で以下に示されています。「k」を数回押すと、2 つの垂直カーソル・ラインの間のどこかに遅延が表示されます。



「6」キーを押し、RX0 + RX1 -> TX0 コンセントレーター・デモ

コンセントレーターのデモは、2 つのレシーバー・チャンネル（RX0 と RX1）でメッセージを受信し、両方を TX0 トランスミッタで再送信します。

このデモでは、TX0 スケジューラ・テーブルに 2 つのディスクリプタを設定します。最初のディスクリプタは RX0 チャンネル・メッセージを参照し、2 番目のディスクリプタは RX1 メッセージを参照します（両方ともラベル 0）。「6」を押すと、コントロールはメイン・コンソール・メニューに戻ります。「r」を押して RUN ピンを High に設定し、スケジューラーを起動します。レシーバー0、または、レシーバー1 で受信した新しいメッセージは、ホストの介入なしに TX0 に自動的に再送信されます。オプションで、ホストは必要に応じて、RAM アクセス、または、FIFO 受信のいずれかを使用してメッセージを受信できます。これは、2 つのディスクリプタが場所 0x4000 の TX0 ディスクリプタ・テーブルにロードされることを除いて、Repeater デモと非常によく似ています。

メモリ位置 0x4000 のディスクリプタ・テーブルを表示します（「a」コマンド使用）

```
             0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
MAP:4000 50 02 00 00 11 00 00 00 51 03 00 00 22 00 00 00
MAP:4010 00 (EOS)
```

ここでも「k」コマンドを使用して、TX2、および、TX3 出力でテスト・メッセージを送信します。
TX2 と TX3 の出力はそれぞれ RX0 と RX1 に接続する必要があります。そして最後に、コンソールに表示された RX3 によって受信された TX0 出力を表示するためのジャンパーTX0 から RX3

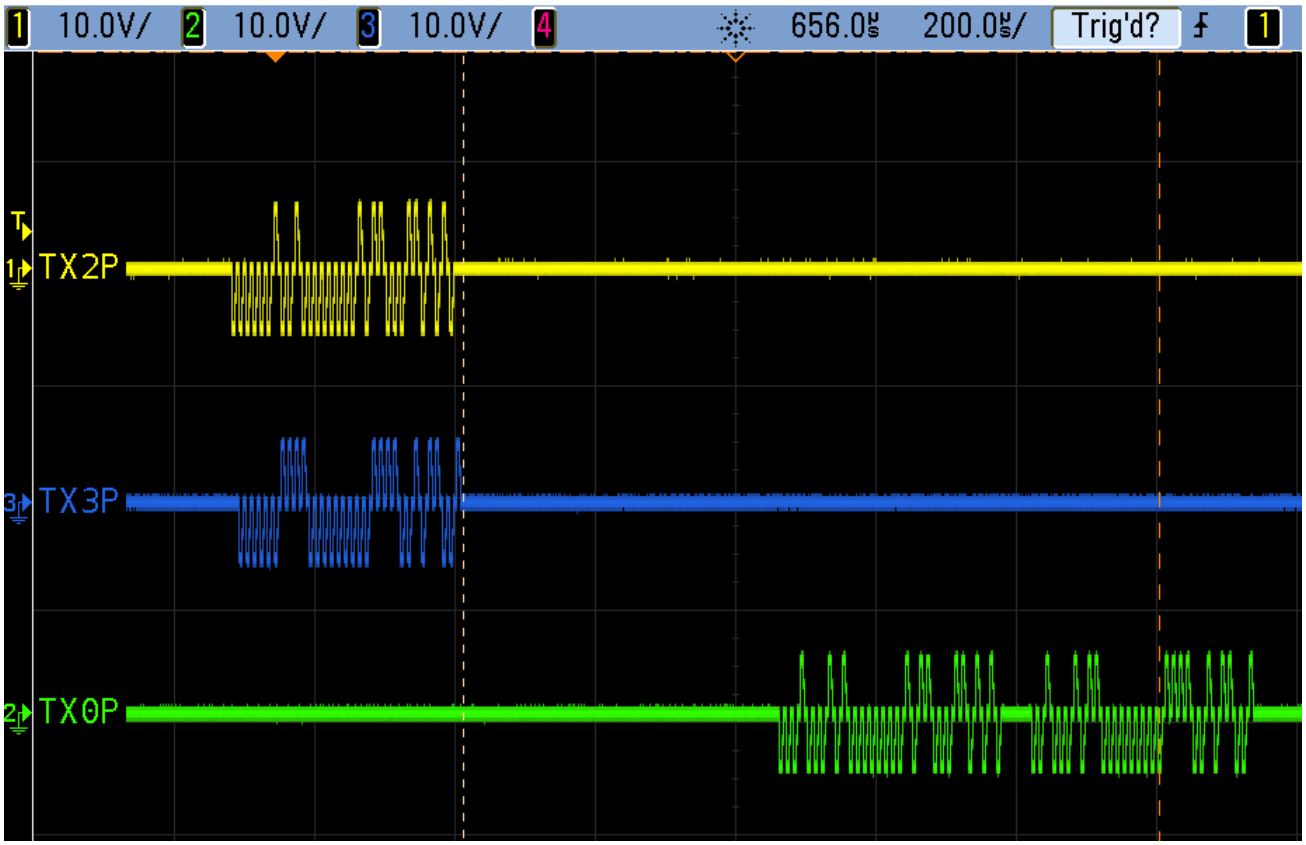
TX2 トランスミット 00 02 34 56 -> RX0 で受信される
TX3 トランスミット 00 03 78 9A -> RX1 で受信される

結果：

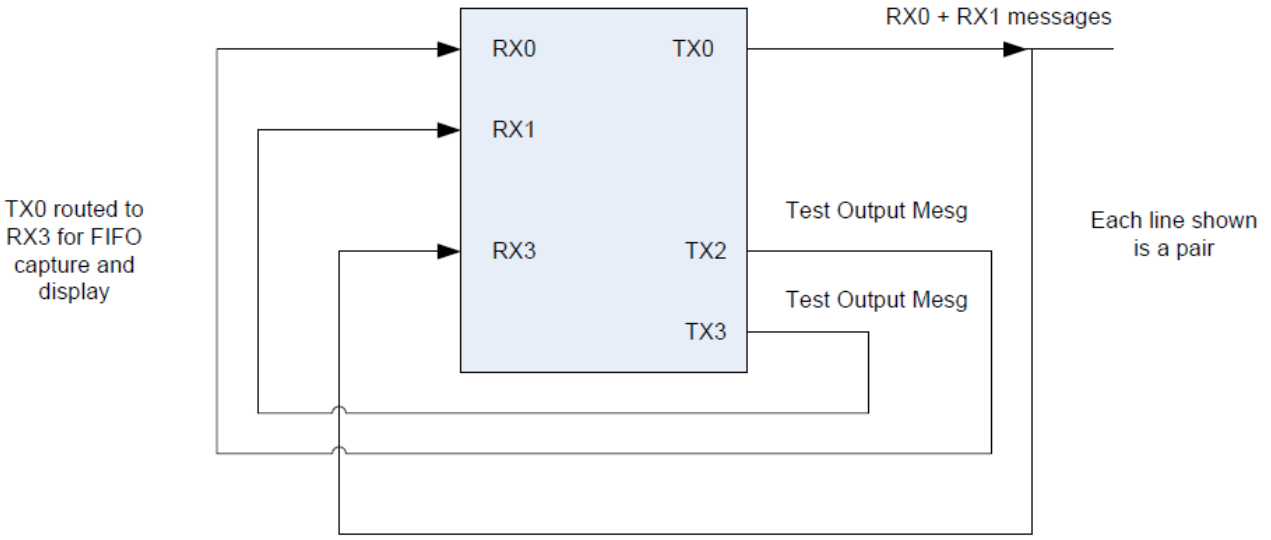
```
> 6
Concentrator Demo
Press 8 to enable the receivers
Press r to set RUN high to run scheduler
Press k to transmit TX2 TX3 test messages
Press 9 to read the FIFO data and display it on the console

        mesgs: 1
        RX 0:0 02 02 34 56
        mesgs: 1
        RX 1:0 03 03 78 9A
        mesgs: 2
        RX 3:0 11 02 34 D6
        RX 3:1 22 03 78 9A
```

ここで注意すべき点がいくつかあります。このデモのディスクリプタ・テーブルでは、ラベル値が 0x11 と 0x22 に変更されたため、これらはコンセントレーターが送信する新しい送信ラベル値です。RX3 レシーバーは TX0 出力を監視しており、RX0 と RX1 の両方からメッセージを受信して表示されます。

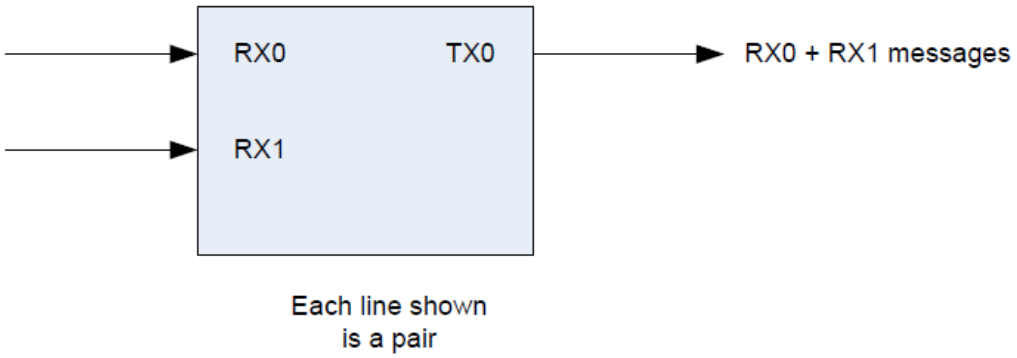
コンセントレーター送信のスコープ・キャプチャが表示されます：



ブロック図は、このデモのペア接続を示しています。



簡略化された実装では、2 つの RX0、および、RX1 チャンネル入力と、このコンセントレーター用の 1 つの TX0 チャンネル出力のみが必要であり、すべてのループバック接続は必要ありません。



Demo 5 は単一のディスクリプタを使用しますが、Demo 6 は 2 つのディスクリプタのみを使用し、使用可能なディスクリプタ・テーブル・メモリをほとんど使用しません。この未使用のディスクリプタ・スペースは、他のメッセージング・タスクに使用できます。

TX0 ワンタイム送信デモ。「7」キーを押し、サブメニューから「1」を押す

このデモは、Demo 3 とまったく同じデモ・コード関数を使用しますが、10 ms の遅延の後、TX0 制御レジスタが再度ライトされて RUN ビットをクリアします。これは、プログラムされたすべてのディスクリプタ・メッセージの 1 回限りの送信を示しています。すべてのメッセージがシーケンスされて送信された後、スケジューラーは停止します。TX0 出力のスコープで送信されたメッセージを表示するか、任意のレシーバーに外部接続し、「8」、および、「9」コマンドを使用して受信を有効にし、前のデモと同様にコンソールにメッセージを表示します。

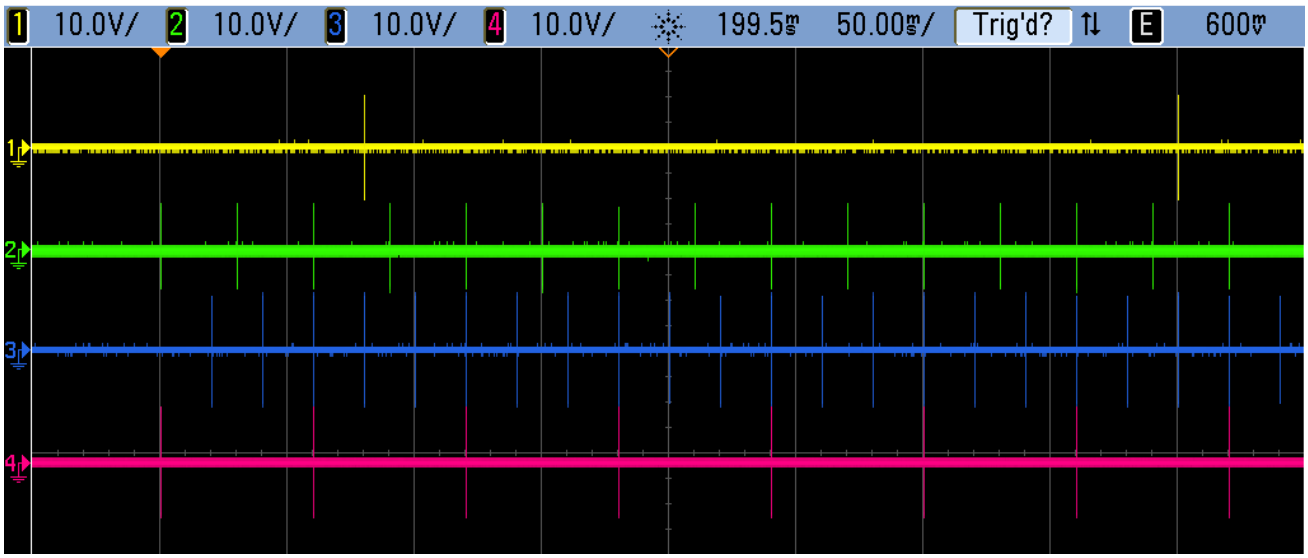
TX0-TX3 スケジューラ・パケットタイマー・デモ。「7」キーを押し、サブメニューから「1」を押す

パケットタイマー・デモは、TX0、TX1、TX2、および、TX3 のディスクリプタ・テーブルをプログラムして、HI-3220 データシートの「パケット・タイマー」セクションにリストされている例 1～4 を実装します。各例では、ディスクリプタ・バイトの PTP（byte 3）と PTO（byte 4）に異なる値を使用して、送信スケジューラーの柔軟なタイミング・オプションを示しています。このスコープ・プロットを複製するには、オシロスコープの外部トリガー入力を使用して TP2 の RUN 信号を正にトリガーし、4 チャンネル・オシロスコープで TX0、TX1、TX2、および TX3 を表示します。

「7」キーを押す、それから、「2」キーを押す

「r」キーを押して、RUN ピンを High に設定するスケジューラーを開始します

TX0 = 黄色、TX1 = 緑、TX2 = 青、TX3 = 赤



Demo P：このデモでは、小文字の「p」と大文字の「P」の両方が使用されています

事前にプログラムされた EEPROM とともに HI-3220 を使用すると、MCU、または、FPGA の制御なしで自動初期化と自律動作が可能になります。コンフィグレーション・データが EEPROM に保存されると、モード[2:0]の場合、HI-3220 は、データシートのメモリ・マップに影付きで示されているすべてのレジスタと選択されたメモリ領域を EEPROM から HI-3220 に自動的にロードします。入力は、データシートの 27 ページのフローチャートに従ってモード 1、または、モード 3 用にコンフィグレーションされています。

ユーザーは、「P」コマンドを使用して EEPROM をプログラミングする前に、デバイスに制御レジスタ、ディスクリプタ・テーブルなどをコンフィグレーションして、目的の機能を実現する必要があります。「P」コマンドは、HI-3220 データとレジスタ・スペースを EEPROM にコピーするだけです。

このデモでは、関数 autoInitEEPROMExample()が最初に呼び出されます。この関数は、16 個のレシーバーすべてと 8 個のトランスミッタを LOOPBACK モードでの High Speed 動作用にコンフィグレーションします。このコンフィグレーションは、このデモの EEPROM にプログラムされています。

デモ手順：

1. J7、J8、および、J9 のジャンパー・シャントを Down 位置に配置して、3 つのモードピン、Mode0、Mode1、および、Mode2 をローに設定します。注:これは、EEPROM が以前にプログラムされたことがない場合にのみ必要です。
2. 「p」を押して autoInitEEPROMExample()関数を実行します。このデモ・コンフィグレーションをテストするには、以下の手順に従います。この演習は、EEPROM をプログラミングする前に、コンフィグレーションが希望どおりに動作することを確認するために実行されます。
   a. 「r」キーを押して、RUN を High にセット
   b. 「k」キーを押して、TX2、および、TX3 でメッセージを送信
   c. 「9」キーを押して、受信したメッセージをリードします。LOOPBACK により、各トランスミッタは 2 つのレシーバーに内部的にルーティングされて複製されます。

```
mesgs: 1
RX 4:0   02 02 34 56
mesgs: 1
RX 5:0   02 02 34 56
mesgs: 1
RX 6:0   03 03 78 9A
mesgs: 1
RX 7:0   03 03 78 9A
```

3. 「P」キーを押して、このコンフィグレーションを EEPROM にプログラムします。赤いエラーLED が短時間点滅します。
4. Mode1、または、Mode3 の Mode0-2 ジャンパーを変更し、ボードの電源をオフ／オンにするか、RESET を押します。HI-3220 は EEPROM から自動初期化する必要があります。
5. 上記のシーケンス・ステップ a～c を繰り返して、デモが以前と同じように機能することを確認しますが、自動初期化シーケンスによってコンフィグレーションが EEPROM から HI-3220 にコピーされたため、今回は autoInitEEPROMExample()関数呼び出しは必要ありません。

EEPROM からの自動初期化はオプション機能です。この機能を使用しない場合、EEPROM は必要ありません。

Demo x：カスタム・メッセージを送信します

「x」を押して、TX0～TX7 トランスミッタ・チャンネルでカスタム・メッセージを送信します。このユーティリティは、チャンネル番号（0～7）、メッセージ数（0～32）、ラベル値（バイト）の入力を求め、送信制御レジスタ値がメッセージを送信します。

Console Menu Utilities【コンソール・メニュー・ユーティリティ】

SPI ペリフェラルは、開発中にいくつかの問題を提示します。従来のバス・インターフェイス・デバッグ環境では、メモリ・ウィンドウを使用して、メモリ・マップド・レジスタ、または、メモリの内容を表示します。この機能は SPI を使用して使用できないため、Holt デモには、レジスタとメモリ・スペースのライト／リードのためのさまざまな SPI ユーティリティ機能が用意されています。これらのユーティリティは開発中のデバッグに不可欠ですが、ハイパーターミナルや TeraTerm などのターミナル・エミュレーションを実行している PC に接続された UART／シリアル・ポートが必要です。提供されている関数の一部は、単一のアドレス位置を書き込み、または、表示します。他の人はアドレスの範囲にアクセスします。ディフォルトでは、すべてのアドレスとデータの値は 16 進数です。

コマンド「w」:

このコマンドは、位置 0x8000 で始まる 256 Byte の領域を表示します。0x8000 はシステム・レジスタの開始アドレスであるため、これは便利です。電源投入、または、リセット後、ほぼすべてのメモリにゼロが含まれます。アドレス 0x8002 の上部ニブルの「2」は、MSR の READY ビットが High に設定されていることを示していることに注意してください。

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
 MAP:8000 00 00 20 00 00 00 00 00 00 00 00 00 00 00 00 FF
 MAP:8010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:8020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:8030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:8040 00 00 00 00 00 00 00 00 00 00 F8 00 00 00 00 00
 MAP:8050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:8060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:8070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00
 MAP:8080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
============================================================
Keys: 'W' Watch '<' Up '>' Down 'R' Refresh 'A' Address
0x8000-0x808F  End MAP:8090
```

「<」キーを押すと開始アドレスから 256 が差し引かれ、「>」を押すと開始アドレスに 256 が追加されます。

コマンド「a」:

これにより、4 桁の 16 進アドレスの入力が求められ、その 256 Byte の領域が表示されます。

```
a
Enter 16-bit hex address: 4000
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
 MAP:4000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:4010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:4020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:4030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:4040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:4050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:4060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:4070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:4080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:4090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:40A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:40B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:40C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:40D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 MAP:40E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
   MAP:40F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  ============================================================
  Keys: 'W' Watch '<' Up '>' Down 'R' Refresh 'A' Address
0x4000-0x40FF End MAP:4100
```

コマンド「c」:

ボードのリセットや電源の入れ直しを使用する代わりに、コンソールからすべてのデバイス・メモリをクリアすると便利な場合があります。これにより、新しい初期化値が保証され、デバイス・メモリがゼロにクリアされるため、後で更新された値を簡単に識別できます。「c」を押して、すべてのメモリをクリアします。

コマンド「d」:

このコマンドは、チャンネル値の入力を求め、0～F が入力されることを想定しています。256 個のラベルすべてに対応する RXRAM メッセージ・バッファが表示されます。ユーティリティは、プロセスを繰り返すために別の値を継続的に要求します。「q」を押してユーティリティを終了します。

コマンド「L」:

このコマンドは、Received Data EnableLook-Up テーブルの ARINC ラベルを設定、または、クリアする 2 つの便利な機能を示しています。20 ページの HI-3220 データシートを参照してください。関数 setLabel(chx、label)および clearLabel(chx、label)は、ユーザーが独自の C プロジェクトで呼び出すことができます。これらの関数の入力パラメーターは、レシーバー・チャンネル番号(0～15)とラベル(0～255dec)です。テーブルで適切なビットを手動で設定することは、それ以外の場合は面倒なプロセスです。データ・イネーブル・ルックアップ・テーブル図がデータシートに記載されており、ラベル・イネーブル・ビットがテーブル・メモリにどのように配置されているかを示しています。

「L」キーを押して、デモで呼び出されるラベルを設定します。

次のシーケンスが実行されます。

1. チャンネル CH0 ラベル設定:0、16、127、128、254
2. チャンネル CH7 ラベル設定:1、17、128、255
3. チャンネル CH15 ラベル設定:255
4. テーブル・メモリがコンソールに表示されます
5. チャンネル CH15 ラベルがクリアされます
6. テーブル・メモリがコンソールに再度表示され、CH15 ラベル 255 がクリアされたことを示します。

結果

```
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
MAP:6800 01 00 01 00 00 00 00 00 00 00 00 00 00 00 00 80   RX0 Labels 0, 16, 128 set
MAP:6810 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 40   RX0 Labels 128, 254 set
MAP:6820 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6830 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6840 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6850 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6860 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6870 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6880 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6890 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68E0 02 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00   RX7 Labels 1, 17 set
```

```
MAP:68F0 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80   RX7 Labels 128, 255 set
MAP:6900 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6910 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6920 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6930 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6940 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6950 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6960 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6970 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6980 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6990 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80  ←RX15 label 255 set

          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
MAP:6800 01 00 01 00 00 00 00 00 00 00 00 00 00 00 00 80
MAP:6810 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 40
MAP:6820 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6830 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6840 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6850 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6860 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6870 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6880 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6890 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68E0 02 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68F0 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80
MAP:6900 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6910 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6920 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6930 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6940 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6950 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6960 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6970 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6980 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6990 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ←– CH15 label 255 cleared
```

オプションで「a」または「e」キーを使用して、デバイス・メモリのこの領域を表示します。

コマンド「m」:（レジスタ／メモリ値を変更）

このコマンドは、任意の 16 Bit アドレスにバイトをライトするために使用されます。

```
> m


Enter 16-bit hex address: 69ff
```

Enter hex byte : 80 readback: 80 (This action just re enabled CH15 label 255 again from the previous demo.

ユーザーが探索でき、役立つと思われるユーティリティ・コマンドが他にもいくつかあります。

Application considerations 【アプリケーションに関する考慮事項】

ARINC 429 Reception by RAM vs. Reception by FIFO (and host keeping pace with data) 【ARINC 429 RAM による受信と FIFO による受信（およびホストがデータと歩調を合わせる）】

ARINC 429 メッセージをリードする方法は 2 つあります。RAM アクセスによる受信では、チャンネル番号（0～15）とラベル値に従って 16K デバイス・メモリにマップされた単一の 4 Byte のバッファ位置が使用されます。メモリ・コンフィグレーションについては、18 ページのデータシートを参照してください。FIFO アクセスによる受信では、レシーバー・チャンネル（0～15）ごとに 64 メッセージ FIFO が使用されます。

推奨される方法は、ユーザーの要件、ホスト・コントローラーの速度、ソフトウェアのオーバーヘッド、および ARINC 429 トラフィックによって異なります。RAM アクセスによる受信を使用する場合、ホストは次のメッセージが到着する前に同じチャンネルと同じラベルでメッセージをリードする必要があります。そうしないと、次のメッセージ・データが前のデータを上書きします。次のメッセージが別のチャンネル、または、別のラベルにある場合、そのチャンネル、または、ラベルは別の 4 Byte のバッファ位置にマップされるため、上書きは発生しません。INT 割り込みを使用して、これらのメッセージのリードに対する高速応答を実現できます。High Speed では、ARINC429 メッセージは約 360 µs ごとに到着する可能性があります。

FIFO アクセスを使用すると、ホストが着信トラフィックに追いつくのに十分な速度でメッセージをリードできなかった場合に上書きが発生する前に、最大 64 のメッセージを保存することにより、厳しいタイミング制約を緩和できます。RX チャンネル割り込みと FIFO しきい値レジスタは、アプリケーション要件を満たすのに十分な速度でメッセージを処理するソリューションを設計するための柔軟性を提供します。上書き（または、データ損失）が発生する前に最大 64 のメッセージを保存する FIFO の機能のため、割り込みの代わりにポーリング戦略がよく使用されます。

ARINC 429 メッセージを受信するようにデバイスをコンフィグレーションする最も簡単な方法は、FIFO 受信方式を使用することです。ARINC 429 データをコンフィグレーション、および、送信する最も簡単な方法は、MCU 直接送信方式を使用することです。

Improved Host Receiver FIFO reading efficiency with the HI-322x family 【HI-322x ファミリによるホスト・レシーバ FIFO リード効率の向上】

HI-3220 ファミリの新機能は、送信 FIFO、または、受信 FIFO のいずれかの現在のメッセージカウントをリードする機能です。これにより、ホストはチャネルの FIFO カウントをリードし、そのカウントを Read_FIFO16 関数に渡して、すべてのメッセージを 1 回のショットでリードすることができます。さらに、新しい C ドライバーは一度に 16 Bit をリードします。この機能はいくつかの場所のデモで使用されていますが、コードの 2 つの重要な行は、

```
mesgCnt = ReadRegister(FCVOR, 0); Fetch message count of Receiver RX0 FIFO
Read_FIFO16(ch0, bufferRX, mesgCnt); Reads all FIFO messages into the buffer
```

EEPROM auto-initialization feature 【EEPROM 自動初期化機能】

オプションの EEPROM 自動初期化機能により、デバイスは、電源投入、または、リセット後に、レジスタ、送信スケジューラ・テーブル、割り込みマップ、FIFO イネーブル・マップの値を事前にプログラムされた EEPROM からデバイスのメモリ・スペースに自動的にコピーできます。この機能は、ホスト MCU、または、FPGA が初期化を実行する前にデバイスを初期化して実行する必要がある場合、または、アプリケーションがホスト MCU、または、FPGA なしで自律的に動作する自律データ・コンセントレータ、または、リピーターである場合に役立ちます。ほとんどの場合、ホストが使用可能な場合、「EEPROM からの初期化」は通常必要ありません。

EEPROM を使用しない場合、Mode0-2 ピンは、必要な電源投入モードに応じて、1K～3.3K の抵抗を介して VDD、または、GND に接続する必要があります。これらのピンにはプルアップ抵抗、または、プルダウン抵抗が組み込まれておらず、データシートの図 1 に示すように、3bit モードは 0、1、2、または、5 のいずれかである必要があります。

HI-3220 SPI basics for troubleshooting 【トラブル・シューティングのための HI-3220 SPI の基本】

これは、HI-3220 とのホスト SPI 通信に関する簡単なチュートリアルです。以下で説明するように、SW2 プッシュ・ボタン・スイッチを押すと、4 つの単純な SPI シーケンスが生成されます。ロジック・アナライザのプロットは、これらのそれぞれの波形を示しています。この情報は、マスターSPI インターフェイスのトラブル・シューティングに役立つ場合があります。

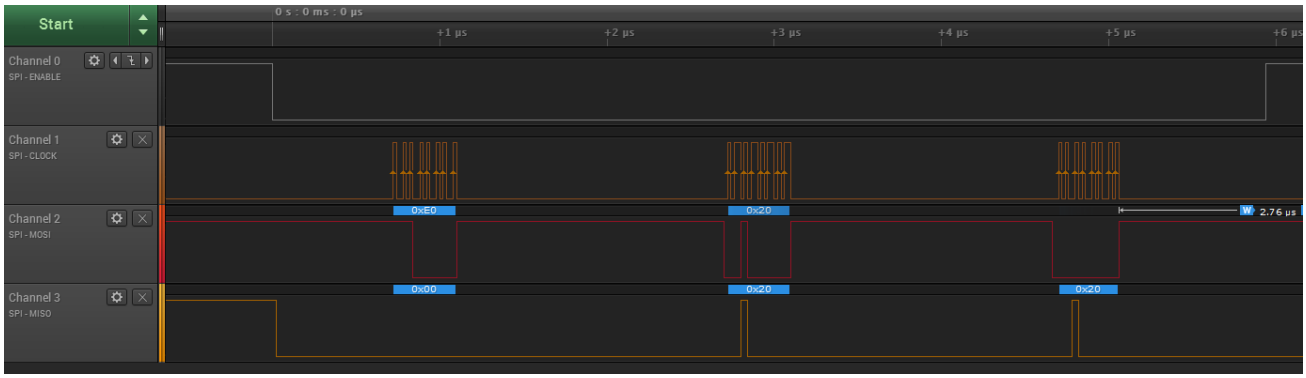
これらの波形を生成する 4 つの別々の SPI シーケンスを以下に示します。

```
retVal = ReadRegister(0x02, 0);     // read MSR register 0x8002
Write_MAP (0x8000,0);               // write 0x8000 address (MCR) to the MAP pointer
Write_MAP_Byte (0xC0,0);            // write 0xC0 to MCR - MAP points to MCR register
retVal2 = Read_MAP_Byte(0);         // read byte at existing MAP pointer
```

SW2 を押すと、コンソールに retVal と retVal2 の値が表示されます。

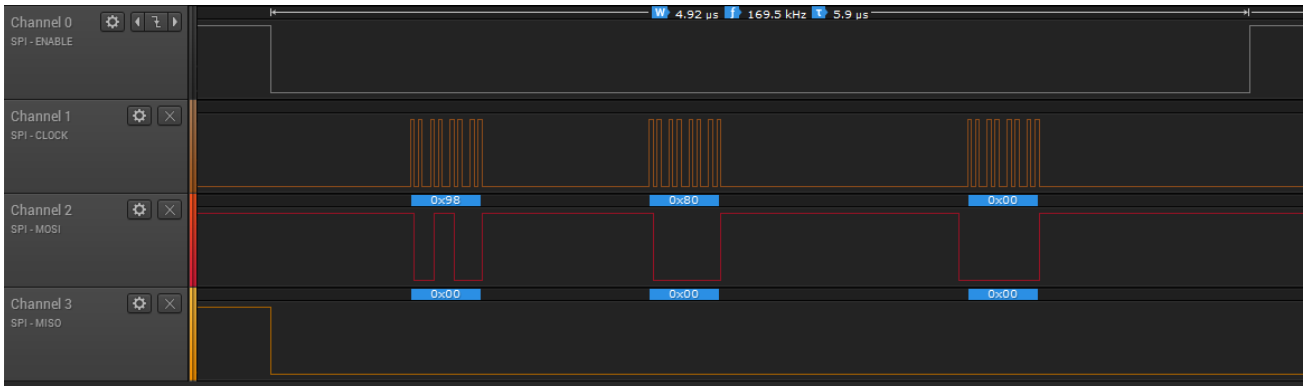
```
Simple SPI example
MSR = 20
MCR = C0
```

インターフェイスのテストに使用できる最も単純な SPI 動作は、Bit5 READY が High に設定された 0x20 を含むマスター・ステータス・レジスタ（MSR）の値をリードするための Read Register OpCode です。リセット後に HI-3220 が READY になると、READY 出力ピンが High に設定され、SPI によってリードされる場合は MSR のビット 5 も High になります。レジスタ・リード 2 Byte のオペ・コード 0xE0 + 0x20 は、オペ・コード・バイトとそれに続く 16 Bit アドレスの下位部分の別のバイトでコンフィグレーションされます。AAA-AAAA フィールドは、最初のオペ・コード・バイトと 2 番目のオペ・コード・バイトに分割されます。このオペ・コードは、AAA-AAAA フィールドによって提供される下位アドレス・バイトに上位アドレス・バイトの 0x80 を自動的に追加するため、アドレスは 0x8002 になります。マスター・ステータス・レジスタの場所は 0x8002 であるため、このオペ・コードで返されるレジスタです。SPI シーケンスを以下に示します。ユーザーは、マスターSPI 信号がこの波形に類似していることを確認する必要があります。SO データは SPI クロックの立ち下がりエッジでクロック・アウトされ、マスターは立ち上がりエッジでデータをクロック・アウトする必要があることに注意してください。これは SPI モード 0 です。

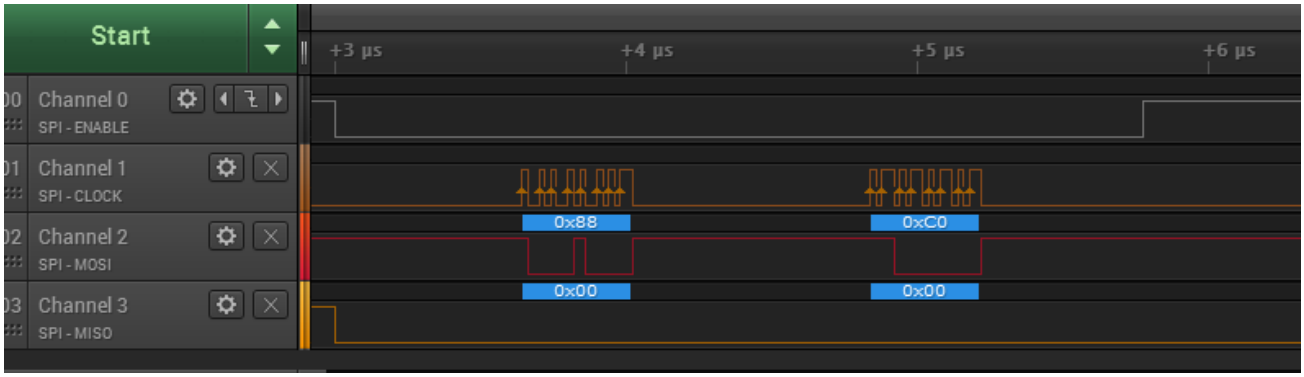


MSR のリード (0x8002)

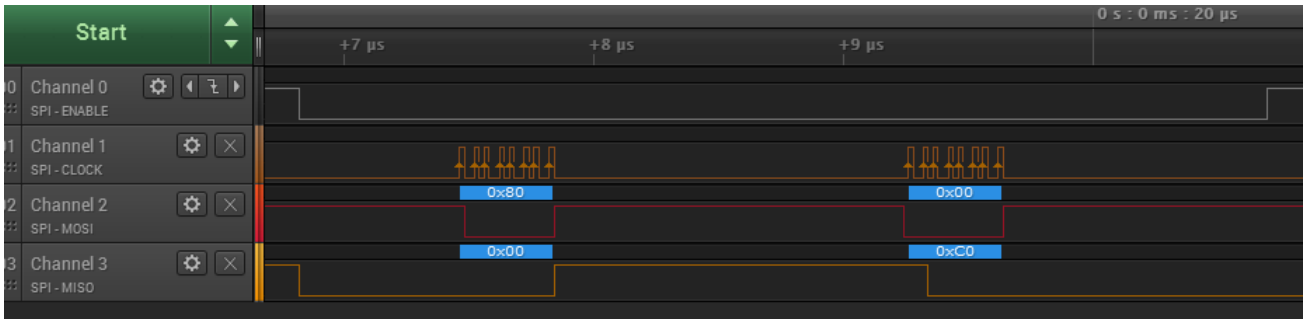
リード・レジスタのオペ・コードが正しく機能している場合、ユーザーは次の手順に進んで、SPI ライトが機能しているかどうかを判断する次の 3 つの SPI シーケンスを確認できます。MAP（メモリ・アドレス・ポインタ）レジスタは、レジスタ（または、メモリ）の場所をリード／ライトするために使用されます。最初にアドレスが 16 Bit MAP ポインタに書き込まれ、次に書き込みアドレスに MAP ポインタ値を使用するオペ・コードを使用してバイト値がこの場所に書き込まれます。次に、同じ MAP ポインタ・アドレスを使用して、値が読み戻されます。これにより、最初にライトされた元のバイト値が返されます。これらの例については、以下の SPI シーケンスを参照してください。



アドレス 0x8000 を MAP ポインタにライトします



MAP アドレスに値 0xC0 をライトします



MAP アドレスの値をリードします（0xC0 である必要があります）

最後のヒント：

1. 制御レジスタ、または、他のメモリ空間にライトするには、READY 出力ピンを High にする必要があります。READY は、/MRST 入力での適格な負のリセット・パルスの後に High にアサートされます。
2. トランスミッタ・ディスクリプタ・テーブルをロードするときは、テーブルが完全にロードされていない限り、最後にロードされたディスクリプタの後に常に EOS（0x00）バイトをロードして、ディスクリプタを終了します。
3. レシーバー、または、トランスミッタが動作するには、RUN ピンが High である必要があります。
4. EEPROM を使用して自動初期化する場合は、事前にプログラムされている必要があります。プログラムされていない EEPROM を使用してモード 1、または、モード 3 で HI-322x の電源を入れると、障害が発生し、赤い LED が点灯します。
5. FIFO メッセージ受信用に RX レシーバー・チャンネルを初期化するときは、受信した FIFO 有効ルックアップ・テーブルも初期化して、目的のラベルが有効になるようにしてください。Logic-1 はラベルを有効にし、0 はラベルを無効にします。デバイスは、すべてのラベル・エントリが 0x00 にクリアされた状態で起動するため、ディフォルトではすべてのラベルが無効になっています。ラベルが無効になっている場合、そのチャンネルでのメッセージ受信は発生しません。デモでは通常、すべてのラベル・テーブル・メモリが 0xFF に設定されます。

Getting Started with the Holt API demo software project and installing IAR Systems Embedded Workbench for ARM Compiler 【HoltAPI デモ・ソフトウェア・プロジェクトの開始と ARM コンパイラ用の IAR Systems Embedded Workbench のインストール】

1. Holt デモ・プロジェクトを追加する前に、IAR Systems Embedded Workbench for ARM（EWARM）コンパイラをインストールする必要があるため、すべての Atmel ボード・ライブラリ・ファイルとデモ・プロジェクト・フォルダが適切な場所に作成されます。Holt CD-ROM の Project フォルダにある「HoltHI-3220 デモ・プロジェクト・インストール・ガイド」に従ってください。次の手順に進む前に、そのガイドに従って、IAR をインストールし、Holt プロジェクトフォルダを適切なフォルダの場所に配置する必要があります。この時点以降の手順は、上記のインストールタスクを完了していることを前提としています。
   Holt デモ・プロジェクトには、IAREWARM バージョン 7.1 以降が必要です。IAR 8.x バージョンを使用している場合は、別のテクニカルノートに追加のガイダンスが記載されています。

2. Windows の[スタート]メニューから IAR Embedded Workbench を起動します。空白の画面が表示されます。IAR ファイル・プルダウン・メニューから HoltHI-3220 デモ・プロジェクトを開き、[ファイル] / [開く] / [ワークスペース]をクリックしてプロジェクトフォルダの場所に移動し、[HI-3220 Demo.eww]を選択して、[開く]ボタンをクリックします。

3. デバッグには、IAR Embedded Workbench®を実行しているコンピュータと HI-3220 アプリケーション開発キット間のインターフェイスが必要です。付属の USB ケーブルの小さい方の端を DEBUG とマークされた評価ボードの USB コネクタに接続します。USB ケーブルのもう一方の端を空きコンピュータの USB ポートに接続します。ARM 用 IARC-SPY デバッガには、組み込みの「J-link On Board」を含む、多数のターゲット・システム・インターフェイス用のドライバーが含まれています。

   評価ボードの USB ケーブルを初めてコンピュータに接続すると、J-Link デバイスに対して Windows の「新しいハードウェアが見つかりました」というメッセージが表示されます。数秒後、Windows は適切なドライバーをロードし、「ハードウェアを使用する準備ができました」とアドバイスする必要があります。Windows が J-Link ドライバーを見つけられない場合は、ドライバー・ディレクトリで IAR Embedded Workbench®インストール CD を探すように指示します。

   手順 5 でデバッグ・セッションを開始するときに問題が発生した場合は、[プロジェクト]、[オプション]の順にクリックします。開いたウィンドウで、Category = Debugger の下の J-Link / J-Trace を強調表示します。[接続]というラベルの付いたタブをクリックし、[通信] = [USB]および[インターフェイス] = [SWD]を確認します。

4. IAREmbeddedWorkbench®を開きます。[ファイル]、[ワークスペースを開く]の順にクリックし、手順 4 で作成したプロジェクト・サブディレクトリに移動します。拡張子が.eww のプロジェクトファイルを選択し、[開く]をクリックします。（次に EmbeddedWorkbench®を開いたときに、[ファイル]をクリックすると、このプロジェクトが[最近のワークスペース]リストに表示されます。）

5. IAR のインストール、または、IAR デバッガの使用で問題が発生した場合、Holt CDROM に含まれているこれらの問題の解決に役立つ 2 つの Holt テクニカルノートが提供されます。

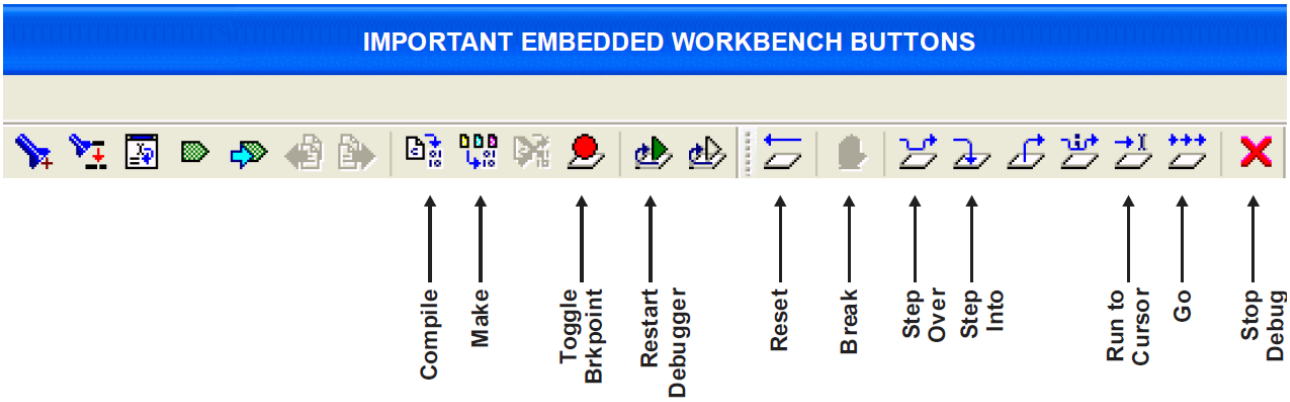
   デモ・プロジェクトでは、符号なし整数変数のみを使用します。必要に応じて、変数の最上位ビットが切換ると

きに発生する迷惑なコンパイラ・メッセージをオフにします。メッセージは次のようになります。
備考[Pe068]：整数変換により符号が変更されましたこの診断メッセージを無効にするには、[プロジェクト]をクリックしてから[オプション]をクリックします
Category = C/C++ Compiler
Tab = Diagnostics
Suppress these diagnostics: add "Pe068" to list

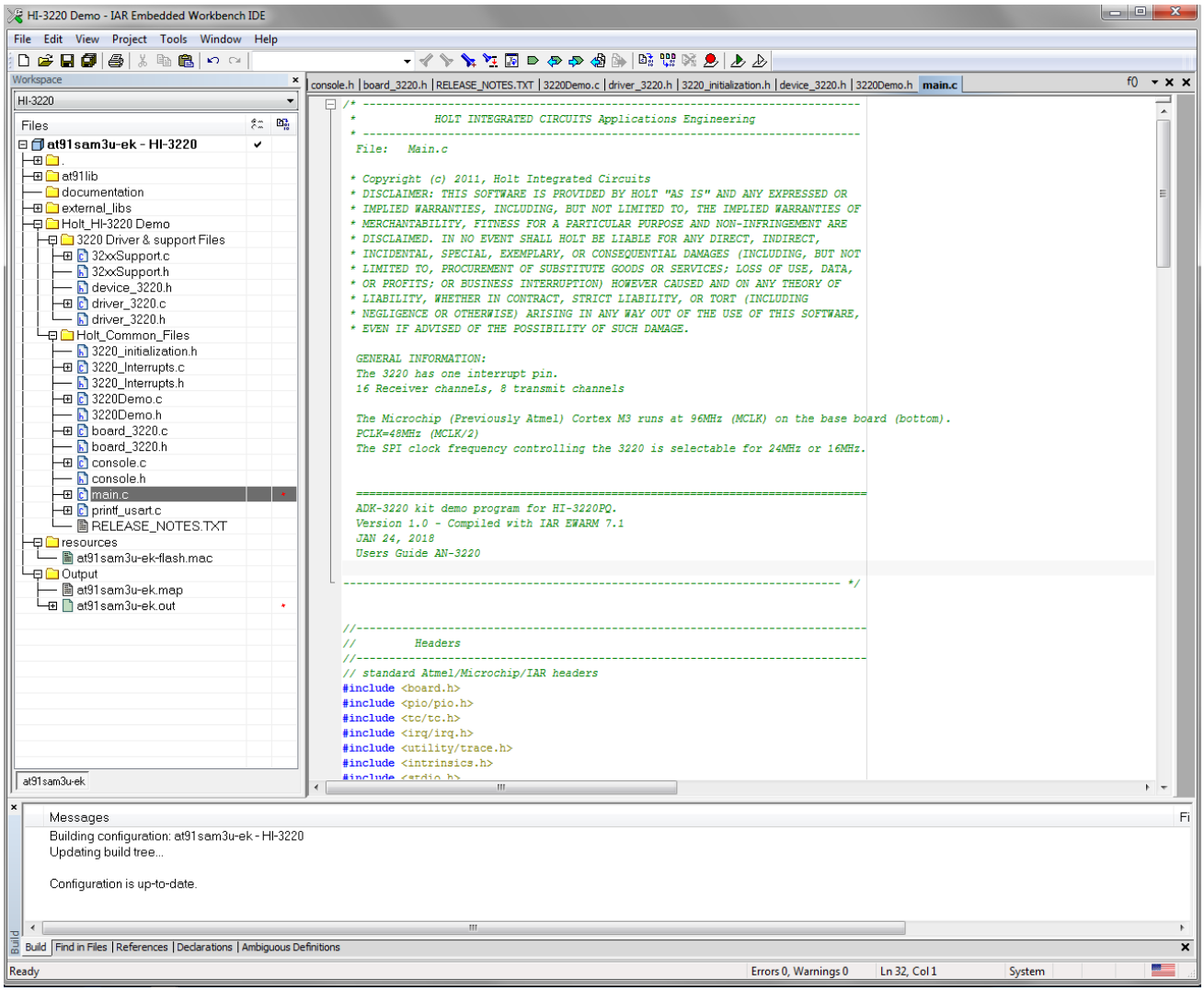
6. MCU の RAM の量が限られているため、RAM ベースのプロジェクトはサポートされていません。設計上、Cortex™-M3 はフラッシュよりも RAM での実行速度が遅いため、RAM ベースのプロジェクトはほとんど必要ありません。

   [作成]ボタンをクリックしてプロジェクトをコンパイルします。次の図を参照してください。
   IAREmbeddedWorkbench®の[メッセージの作成]ウィンドウにエラーや警告が表示されない場合は、続行できます。エラーが発生した場合は、エラーを修正してプログラムを再コンパイルしてください。

7. [デバッガの再起動]ボタンをクリックして、デバッグ・セッションを開始します。これにより、コンパイルされたプログラムが MCU にダウンロードされ、ボードでプログラムを実行できるようになります。[実行]をクリックして実行を開始します。
   実行を停止するには、[ブレーク]（通常は実行中に赤い支持された手として表示されます）をクリックします。



HoltHI-3220 プロジェクトがロードされた IARIDE 画面。IDE 画面のサイズ変更が必要になる場合があります。


HI-3220 Demo - IAR Embedded Workbench IDE
File Edit View Project Tools Window Help
Workspace
HI-3220
Files
at91sam3u-ek - HI-3220
at91lib
documentation
external_libs
Holt_HI-3220 Demo
3220 Driver & support Files
32xxSupport.c
32xxSupport.h
device_3220.h
driver_3220.c
driver_3220.h
Holt_Common_Files
3220_initialization.h
3220_Interrupts.c
3220_Interrupts.h
3220Demo.c
3220Demo.h
board_3220.c
board_3220.h
console.c
console.h
main.c
printf_usart.c
RELEASE_NOTES.TXT
resources
at91sam3u-ek-flash.mac
Output
at91sam3u-ek.map
at91sam3u-ek.out
at91sam3u-ek
console.h | board_3220.h | RELEASE_NOTES.TXT | 3220Demo.c | driver_3220.h | 3220_initialization.h | device_3220.h | 3220Demo.h | main.c
HOLT INTEGRATED CIRCUITS Applications Engineering
File: Main.c
Copyright (c) 2011, Holt Integrated Circuits
DISCLAIMER: THIS SOFTWARE IS PROVIDED BY HOLT "AS IS" AND ANY EXPRESSED OR
IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE
DISCLAIMED. IN NO EVENT SHALL HOLT BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
GENERAL INFORMATION:
The 3220 has one interrupt pin.
16 Receiver channels, 8 transmit channels
The Microchip (Previously Atmel) Cortex M3 runs at 96MHz (MCLK) on the base board (bottom).
PCLK=48MHz (MCLK/2)
The SPI clock frequency controlling the 3220 is selectable for 24MHz or 16MHz.
ADK-3220 kit demo program for HI-3220PQ.
Version 1.0 - Compiled with IAR EWARM 7.1
JAN 24, 2018
Users Guide AN-3220
Headers
standard Atmel/Microchip/IAR headers
#include <board.h>
#include <pio/pio.h>
#include <tc/tc.h>
#include <irq/irq.h>
#include <utility/trace.h>
#include <intrinsics.h>
Messages
Building configuration: at91sam3u-ek - HI-3220
Updating build tree...
Configuration is up-to-date.
Build
Find in Files | References | Declarations | Ambiguous Definitions
Ready
Errors 0, Warnings 0
Ln 32, Col 1
System

Project File List with Selected Descriptions 【プロジェクトファイルリスト】

対応する C ファイルのないヘッダー・ファイル

device_3220.h

レジスタの定義ステートメント、制御レジスタビット、オペ・コード、選択したテーブル開始アドレスなど、HI-3220 のすべてのマクロが含まれています。

3220_initialization.h

デモで割り込みを無効、または、有効にするために使用される INTERRUP_MESG_ENA マクロを含むいくつかの構成設定の定義。

HI-322x 低レベル SPI ドライバーおよびその他の 3220 固有のサポート機能

device_3220.h （上と同じ）

レジスタの定義ステートメント、制御レジスタビット、オペ・コード、選択したテーブル開始アドレスなど、HI-3220 のすべてのマクロが含まれています。

driver_3220.c/driver_3220.h

Cortex M3SPI 低レベル SPI ドライバーとメッセージング機能をホストします。

32xxSupport.c/32xxSupport.h

コンフィグレーション、テーブルの初期化、および ARINC メッセージの送受信のためのホスト・ヘルパー関数。

対応するヘッダー・ファイルを含むその他のメイン C ファイル

main.c

main()のプライマリ・プログラム・エントリとメイン・ループ。

Board_3220.c

SPI 初期化、LED、およびさまざまな I / O 用のマクロが含まれています。 Timer1 ティック 1ms 初期化は、遅延関数とホスト・タイマー割り込みハンドラーをサポートします。 ホスト MCU に関連するボード。

3220_Interrupts.c

3220INT ピンでのメッセージ割り込みの初期化、および、IRQ ハンドラー。 割り込みラベル・イネーブル・テーブル・メモリを設定、または、クリアする機能。

3220Demo.c (these demos are called mostly from the console)

すべてのデモはここに含まれています。 完全なリストについては、ヘッダー・ファイルを参照

console.c

全てのデモで使用されるコンソール機能

Show_menu(); メイン・メニューの表示

chk_key_input(); キー・エントリの検出とデモ実行ハンドラー

Application Development Kit Notes 【アプリケーション開発キットノート】

HI-3220 は、シリアル・ペリフェラル・インターフェイス（SPI）を備えたマイクロコントローラ、または、FPGA との互換性のために設計されました。デバイスの RAM とレジスタの位置は、8 ビット SPI コマンドを使用して書き込まれるかリードされます。ほとんどのリード、または、ライト操作では、メモリ・アドレス・ポインタ（MAP）を使用して、次にアクセスする場所のアドレスを指定します。マルチ・ワード転送を高速化するために、メモリ・アドレス・ポインタは、各ワードのリード、または、ライトが実行された後、自動的に次のアドレスにインクリメントします。

HI-3220 のデータ転送速度は、MCU SPI インターフェイスによって提供される SPI クロック周波数に依存します。ARM Cortex M3 MCU マスター・クロック周波数は、内部 PLL を使用して 96MHz です。MCU ペリフェラルは、48MHz クロック・ソースの場合、このクロックを 2 で除算します。SPI ブロックは、これを 16MHz SPI クロックの場合は 3 で除算し、24MHzSPI クロックの場合は 2 で除算します。HI-3220 の最大 SPI クロック周波数は 40MHz です。

デモ・ボードには、フル機能のハイエンド USB デバッガ・ポートが含まれています。デモ・プロジェクトを再構築し、ARM Cortex M3 プロセッサを再フラッシュするために、追加のデバッグツールは必要ありません。

HI-3220 use with an external MCU or FPGA 【HI-3220 を外部 MCU または FPGA で使用】

50MHz オシレータがこのカードに含まれており、重要なインターフェイス信号が J11 ヘッダー・コネクタで利用できるため、外部 MCU、または、FPGA で HI-3220 ドーター・カードを簡単に使用できます。これらの詳細については、このドキュメントの前のセクションにある回路図とヘッダーのピン配列の説明を参照してください。

Summary 【まとめ】

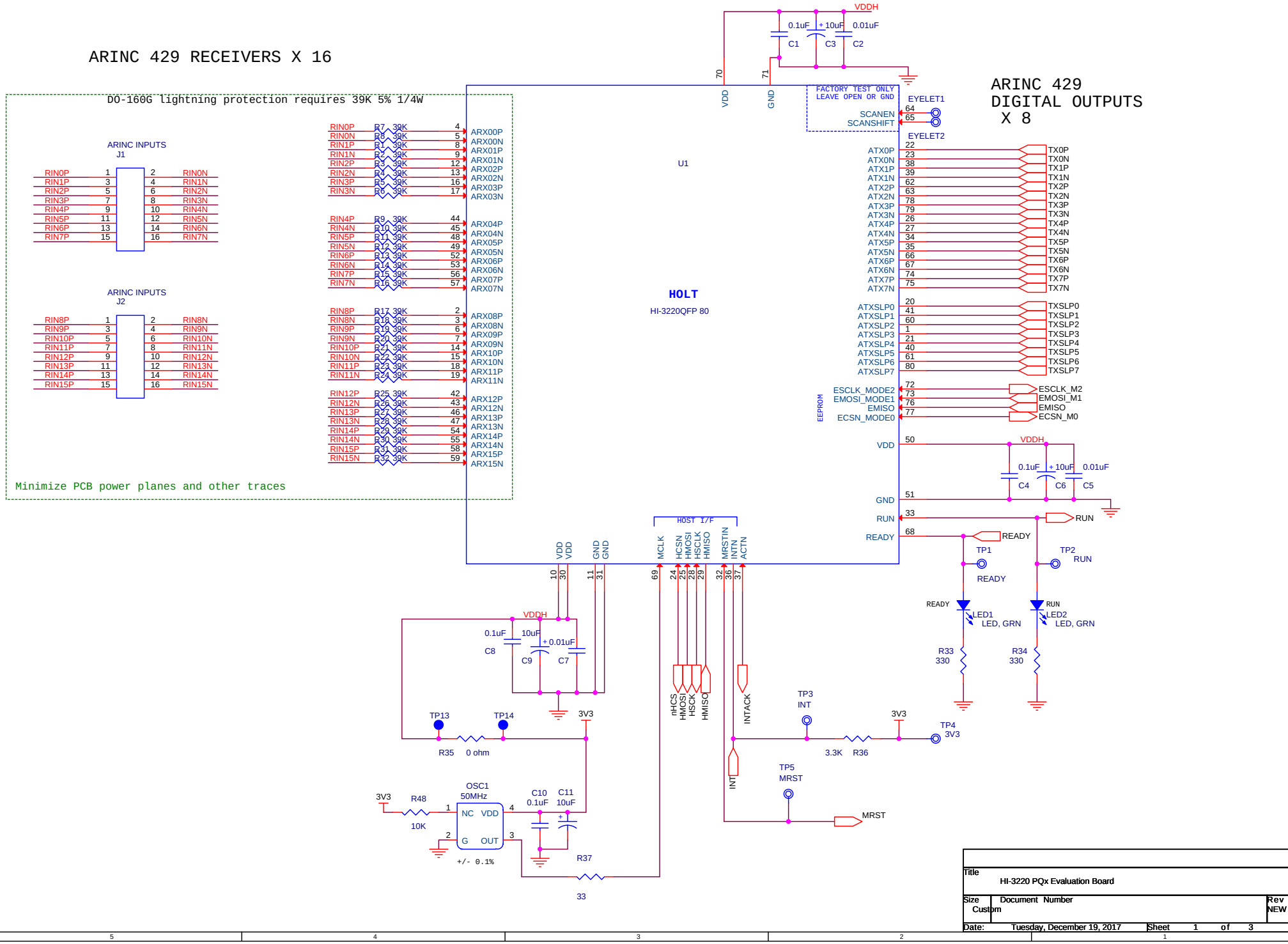
HI-3220 は、強力な ARINC 429 マルチ・チャンネル・トランスミッタ、および、レシーバー・ソリューションです。その高速 4 線式 SPI インターフェイスにより、制御が容易になります。最大 40 MHz の SPI レートと HI-3220 の新機能により、ホストの通信効率が向上します。HI-3220 は、EPPROM から自動初期化でき、一部のアプリケーションでは FPGA、または、MCU なしで動作します。デモ・プログラムは、デバイスのほぼすべての側面を示し、ローレベルの C ドライバーにより、他のプラットフォームへの移植が容易になります。メニューSPI アクセス・ユーティリティの豊富なセットにより、コンソールでデバイス・メモリを表示／変更できます。

PCB Rev. A
P/N: HV033

Bill of Materials
HI-3220PQ
Evaluation Board

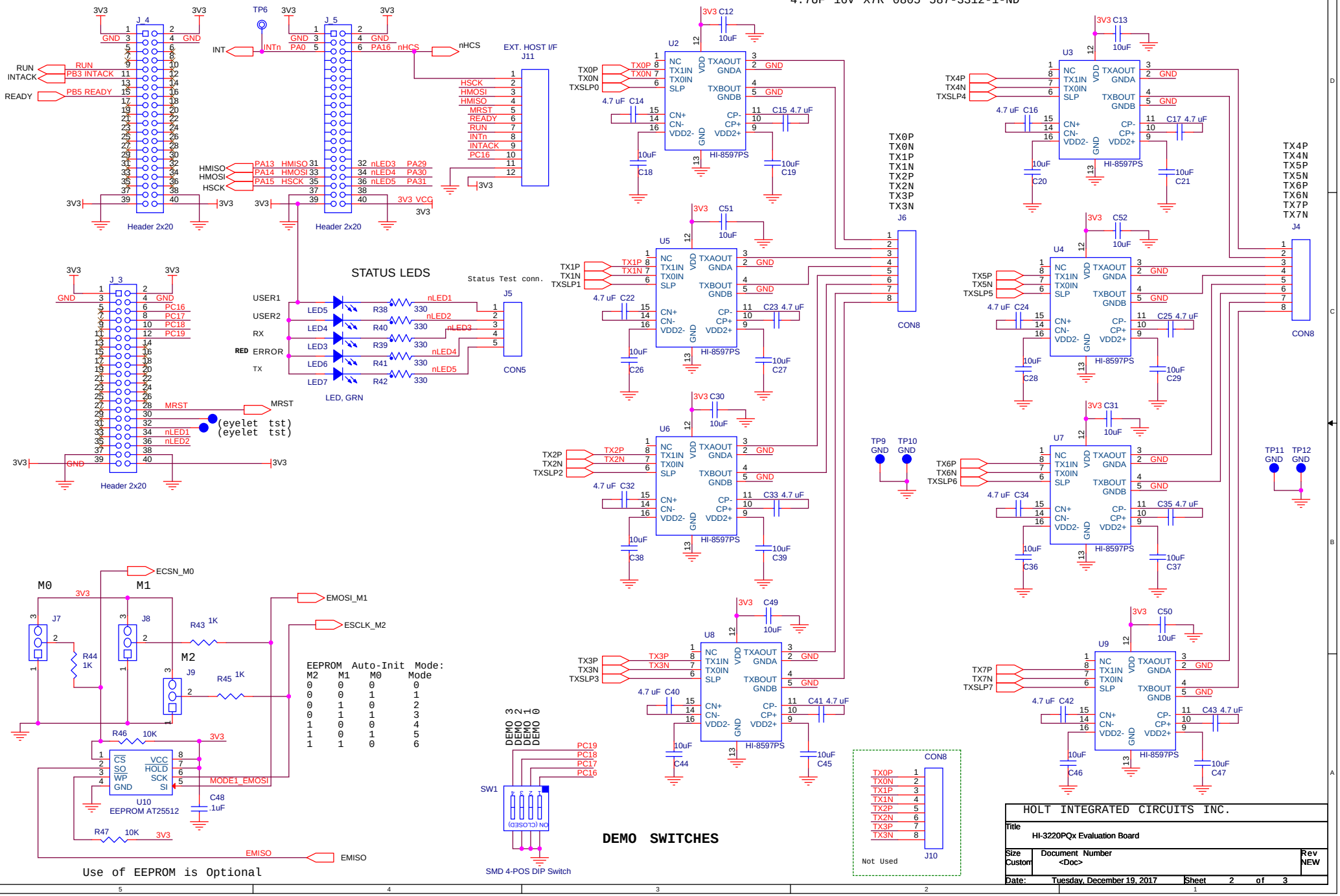
1/25/2018

Item	Qty	Description	Reference	Digikey P/N	Mfg P/N
1	1	PCB, Bare, Evaluation Board	N/A		JetTech # 42174
2	5	Capacitor Cer 0.1uF 25V 10% X7R 0805	C1,C4,C8,C10,C48	478-3755-1-ND	AVX 08053C104KAT2A
3	3	Capacitor Cer 0.01uF 100V 10% X7R 0805	C2,C5,C7	478-1358-1-ND	AVX 08051C103KAT2A
4	16	Capacitor Cer 4.7uF 16V 10% X7R 0805	C14-C17,C22-C25,C32-C35,C40-C43	587-3312-1-ND	Taiyo EMK212AB7475KGHT
5	4	Capacitor Tant 10uF 10V 10% 1206	C3,C6,C9,C11	399-3684-1-ND	Kemet T491A106K010AT
6	24	Capacitor Cer 10uF 16V 20% X7R 0805	C12-C13,C18-C21,C26-C31,C36-C39,C44-C47,C49-C52	587-3319-1-ND	Taiyo EMK212BB7106MG-T
7	1	Resistor 0, 1/8W 5% 0805 SMD	R35	P0.0ACT-ND	Panasonic ERJ-6GEY0R00V
8	1	Resistor 33, 1/8W 5% 0805 SMD	R37	P33ACT-ND	Panasonic ERJ-6GEYJ330V
9	7	Resistor 330, 1/8W 5% 0805 SMD	R33-R34,R38-R42	P330ACT-ND	Panasonic ERJ-6GEYJ331V
10	3	Resistor 1K, 1/8W 5% 0805 SMD	R43-R45	P1.0KACT-ND	Panasonic ERJ-6GEYJ102V
11	1	Resistor 3.3K, 1/8W 5% 0805 SMD	R36	P3.3KACT-ND	Panasonic ERJ-6GEYJ332V
12	3	Resistor 10K, 1/8W 5% 0805 SMD	R46-R48	P10KACT-ND	Panasonic ERJ-6GEYJ103V
13	32	Resistor 39K, 1/4W 0.5% 0805 SMD	R1-R32	P21199CT-ND	Panasonic ERJ-PB6D3902V
14	3	Test Point, Black Insulator, 0.062"	GND (TP10-TP12)	36-5011-ND	Keystone 5011
15	1	Test Point, Red Insulator, 0.062"	3V3 (TP4)	36-5010-ND	Keystone 5012
16	1	Test Point, White Insulator, 0.062"	RUN (TP2)	36-5012-ND	Keystone 5012
17	1	Test Point, Orange Insulator, 0.062"	RDY (TP1)	36-5013-ND	Keystone 5013
18	6	LED Green 0805 SMD	LED1,2,3,4,5,7	160-1423-1-ND	LiteOn LTST-C171GKT
19	1	LED Red 0805 SMD	LED6	160-1422-1-ND	LiteOn LTST-C171EKT
20	3	Header, Male 1x3, .1" Pitch	J7-J9	S1012E-03-ND	Sullins PEC03SAAN
21	1	Header, Male 1x5, .1" Pitch	J5	S1012E-05-ND	Sullins PEC05SAAN
22	3	Header, Male 1x8, .1" Pitch	J4,J6,J8	S1012E-08-ND	Sullins PEC08SAAN
23	1	Header, Male 1x12, .1" Pitch	J11	S1012E-12-ND	Sullins PEC12SAAN
24	2	Header, Male 2x3, .1" Pitch	J-5A	S2012EC-03-ND	Sullins PREC003DAAN-RC
25	2	Header, Male 2x5, .1" Pitch	J_5B	S2012EC-05-ND	Sullins PREC005DAAN-RC
26	2	Header, Male 2x20, .1" Pitch	J_3,J_4	S2012EC-20-ND	Sullins PREC020DAAN-RC
27	1	Switch Tape Seal 4 Pos SMD	SW1	CT2194MST-ND	CTS 219-4MST
28	1	IC EEPROM 512KBIT 20Mhz 8SOIC	U10	CAT25512VI-GT3OSCT-ND	On Semi CAT25512VI-GT3
29	1	Oscillator XO 50.0Mhz HCMOS SMD	OSC1	535-9330-1-ND	Abracon ASV-50.000MHZ-EJ-T
30	8	HI-8597PS	U2-U9	Holt Inc.	HI-8597PS 16-ESOIC
31	1	HI-3220PQ	U10	Holt Inc.	HI-3220 80-QFP


ARINC 429 RECEIVERS X 16
DO-160G lightning protection requires 39K 5% 1/4W
ARINC INPUTS
J1
RIN0P 1 2 RIN0N
RIN1P 3 4 RIN1N
RIN2P 5 6 RIN2N
RIN3P 7 8 RIN3N
RIN4P 9 10 RIN4N
RIN5P 11 12 RIN5N
RIN6P 13 14 RIN6N
RIN7P 15 16 RIN7N
ARINC INPUTS
J2
RIN8P 1 2 RIN8N
RIN9P 3 4 RIN9N
RIN10P 5 6 RIN10N
RIN11P 7 8 RIN11N
RIN12P 9 10 RIN12N
RIN13P 11 12 RIN13N
RIN14P 13 14 RIN14N
RIN15P 15 16 RIN15N
RIN0P R7 39K 4 ARX00P
RIN0N R8 39K 5 ARX00N
RIN1P R1 39K 8 ARX01P
RIN1N R2 39K 9 ARX01N
RIN2P R3 39K 12 ARX02P
RIN2N R4 39K 13 ARX02N
RIN3P R5 39K 16 ARX03P
RIN3N R6 39K 17 ARX03N
RIN4P R9 39K 44 ARX04P
RIN4N R10 39K 45 ARX04N
RIN5P R11 39K 48 ARX05P
RIN5N R12 39K 49 ARX05N
RIN6P R13 39K 52 ARX06P
RIN6N R14 39K 53 ARX06N
RIN7P R15 39K 56 ARX07P
RIN7N R16 39K 57 ARX07N
RIN8P R17 39K 2 ARX08P
RIN8N R18 39K 3 ARX08N
RIN9P R19 39K 6 ARX09P
RIN9N R20 39K 7 ARX09N
RIN10P R21 39K 14 ARX10P
RIN10N R22 39K 15 ARX10N
RIN11P R23 39K 18 ARX11P
RIN11N R24 39K 19 ARX11N
RIN12P R25 39K 42 ARX12P
RIN12N R26 39K 43 ARX12N
RIN13P R27 39K 46 ARX13P
RIN13N R28 39K 47 ARX13N
RIN14P R29 39K 54 ARX14P
RIN14N R30 39K 55 ARX14N
RIN15P R31 39K 58 ARX15P
RIN15N R32 39K 59 ARX15N
Minimize PCB power planes and other traces
U1
HOLT
HI-3220QFP 80
VDDH
0.1uF
10uF
0.01uF
C1
C3
C2
70 VDD
71 GND
FACTORY TEST ONLY
LEAVE OPEN OR GND
EYELET1
SCANEN 64
SCANSHIFT 65
EYELET2
ARINC 429
DIGITAL OUTPUTS
X 8
ATX0P 22 TX0P
ATX0N 23 TX0N
ATX1P 38 TX1P
ATX1N 39 TX1N
ATX2P 62 TX2P
ATX2N 63 TX2N
ATX3P 78 TX3P
ATX3N 79 TX3N
ATX4P 26 TX4P
ATX4N 27 TX4N
ATX5P 34 TX5P
ATX5N 35 TX5N
ATX6P 66 TX6P
ATX6N 67 TX6N
ATX7P 74 TX7P
ATX7N 75 TX7N
ATXSLP0 20 TXSLP0
ATXSLP1 41 TXSLP1
ATXSLP2 60 TXSLP2
ATXSLP3 1 TXSLP3
ATXSLP4 21 TXSLP4
ATXSLP5 40 TXSLP5
ATXSLP6 61 TXSLP6
ATXSLP7 80 TXSLP7
EEPROM
ESCLK_MODE2 72 ESCLK_M2
EMOSI_MODE1 73 EMOSI_M1
EMISO 76 EMISO
ECSN_MODE0 77 ECSN_M0
VDD 50
GND 51
RUN 33 RUN
READY 68 READY
HOST I/F
MCLK 69
HCSN 24 nHCS
HMOSI 25 HMOSI
HSCLK 28 HSCK
HMISO 29 HMISO
MRSTIN 32 MRST
INTN 36 INT
ACTN 37 INTACK
VDD 10 VDD 30
GND 11 GND 31
C8 C9 C7
TP13 TP14
R35 0 ohm
OSC1
50MHz
NC VDD G OUT
+/- 0.1%
R48 10K
C10 0.1uF
C11 10uF
R37 33
TP1 READY
TP2 RUN
LED1 LED, GRN
LED2 LED, GRN
R33 330
R34 330
TP3 INT
TP4 3V3
R36 3.3K
TP5 MRST
C4 C6 C5
Title
HI-3220 PQx Evaluation Board
Size Custom
Document Number
Rev NEW
Date: Tuesday, December 19, 2017
Sheet 1 of 3

HOST INTERFACE

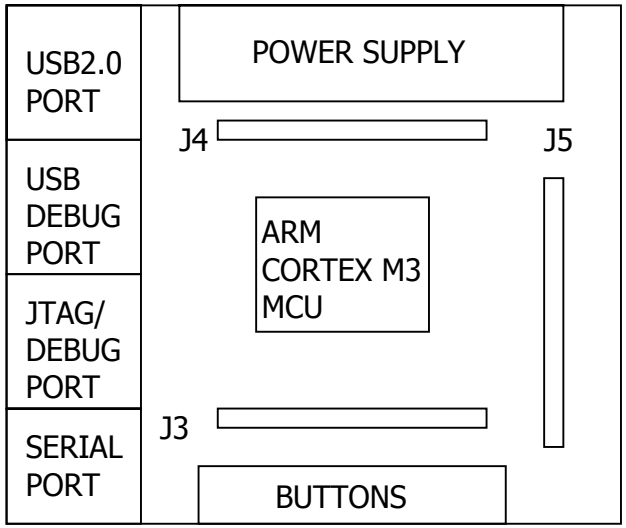
HI-8597 Low ESR ceramic ML capacitors:
10UF 16V X7R 0805 587-3319-1-ND
4.7UF 16V X7R 0805 587-3312-1-ND



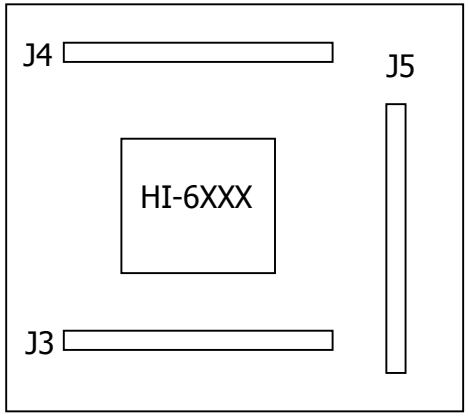
Title	HOLT INTEGRATED CIRCUITS INC. HI-3220PQx Evaluation Board
Size: Custom	Document Number: <Doc>
Rev	NEW
Date:	Tuesday, December 19, 2017
Sheet	2 of 3

Bill of Materials		ARM Cortex M3 MCU Board			Revised: 9 Sept 2011
Item	*Qty*	*Description*	*Reference*	*DigiKey*	*Mfr P/N*
1	1	PCB, Bare, Evaluation Board, revision B or C	N/A	--------	
2	1	Ferrite Bead, 220 Ohm @ 100MHz 300mA DC 0805	FB1	732-1602-1-ND	Wurth 742792034
3	2	Capacitor, Ceramic 10nF 10% 50V X7R 0805	C1,C42	399-1158-1-ND	Kemet C0805C103K5RACTU
4	2	Capacitor, Ceramic 10pF 10% NP0 C0G 0V 0805	C23,C34	478-3731-1-ND	AVX 080551A100KAT2A
5	4	Capacitor, Ceramic 20pF 5% NP0 C0G 0V 0805	C14,C21,C25, C27	478-3735-1-ND	AVX 080551A200JAT2A
6	28	Capacitor, Ceramic 100nF 20% 50V Z5U 0805	C2,C4,C6-C11, C13,C15-C19, C22,C24,C26, C28,C29,C33, C35-C40,C45-46	399-1176-1-ND	Kemet C0805C104M5UACTU
7	4	Capacitor, Tantalum 4.7uF 10% 10V Low ESR SMD 1206	C5,C20,C31, C32	478-2391-11-ND	AVX TPSA475K010R1400
8	4	Capacitor, Tantalum 10uF 10% 10V Low ESR SMD 1206	C3,C12,C30,C41	478-3317-1-ND	AVX TPSA106K010R1800
9	1	Capacitor 22uF 10% 6.3V Tantalum Low ESR SMD C	C43	495-1504-1-ND	Kemet B45197A1226K309
10		Capacitor 100uF 10% 6.3V Tantalum Low ESR SMD C	C44	495-1509-1-ND	Kemet B45197A1107K309
11	1	Header, Male Shrouded 2x10 0.1" Pitch	J1	MHB20K-ND	3M 2520-6002UB
12	1	Connector, Receptacle USB Mini B Rt-Angle PCB Mount	J2	H2959CT-ND	Hirose UX60-MB-5ST
13	1	Connector DB9F, Right-Angle PCB Short Body, Board Lock	J6	182-109FE-ND	NorComp 182-009-213R-561
14	1	Jack, DC Power, 2.5mm ID x 2.1mm pin	J7	CP-102AH-ND	CUI PJ-102AH
15	2	Receptacle, Female 2x20 0.1" Pitch, 8.5mm Height, 3.2mm Solder Tails	J3,J4	S6104-ND	Sullins PPTC202LFBN-RC
16	1	Receptacle, Female 2x4 0.1" Pitch, 8.5mm Height, 3.2mm Solder Tails	J5A (J5 lower end, close to Bus B)	S7072-ND	Sullins PPTC042LFBN-RC
17	1	Receptacle, Female 2x5 0.1" Pitch, 8.5mm Height, 3.2mm Solder Tails	J5B (J5 upper end, close to Bus A)	S6105-ND	Sullins PPTC052LFBN-RC
18	1	Solder Jumper	JP1	SOLDER CLOSED	
19	2	Inductor, 10uH, 100mA 0805	L1,L2	490-4029-1-ND	Murata LQM21FN100M70L
20	1	LED Green 0805	LED1	160-1179-2-ND	LiteOn LTST-C170GKT
21	0	Resistor, Prov 1/8W 0805	R1	DO NOT STUFF	
22	5	Resistor, 0 ohm 1/8W 0805	R12,R13,R22, R23,R29	311-0ARCT-ND	Panasonic ERJ-6GEY0R00V
23	2	Resistor, 1.0 5% 1/8W 0805	R7,R8	P1.0ACT-ND	Panasonic ERJ-6GEYJ1R0V
24	2	Resistor, 39 5% 1/8W 0805	R4,R5	P39ACT-ND	Panasonic ERJ-6GEYJ390V
25	1	Resistor, 150 5% 1/8W 0805	R17	P150ACT-ND	Panasonic ERJ-6GEYJ151V
26	1	Resistor, 4.7K 5% 1/8W 0805	R3	P4.7KACT-ND	Panasonic ERJ-6GEYJ472V
27	1	Resistor, 6.8K 5% 1/8W 0805	R6	P6.8KACT-ND	Panasonic ERJ-6GEYJ682V
28	1	Resistor, 47K 5% 1/8W 0805	R18	P47KACT-ND	Panasonic ERJ-6GEYJ473V
29	1	Resistor, 68K 5% 1/8W 0805	R19	P68KACT-ND	Panasonic ERJ-6GEYJ683V
30	10	Resistor,100K 5% 1/8W 0805	R2,R10,R11, R20,R21,R24, R25,R26,R27, R28	P100KACT-ND	Panasonic ERJ-6GEYJ104V
31	3	Pushbutton	SW1,SW2,SW3	P10886SCT-ND	Panasonic EVQ-QWS02W
32	2	Test Point, Black Insulator, 0.062" hole	TP2,TP3	5011K-KD	Keystone 5011

33	1	Test Point, Orange Insulator, 0.062" hole	TP1	5008K-ND	Keystone 5008
34	1	Test Point, Yellow Insulator, 0.062" hole	TP4	5009K-ND	Keystone 5009
35	2	Test Point, Hole / Pad Only	TP5,TP6		
36	1	IC, MCU 32-Bit 256KB Flash, 144-LQFP	U1	ATSAM3U4EA-AU-ND	Atmel ATSAM3U4EA-AU
37	1	IC, ESD Protection Array 3-Channel SOT-5	U2	296-21885-1-ND	Texas Inst TPD3E001DRLR
38	1	IC, RS232 Driver/Receiver 3.0 to 5.5VDC 16-SOIC (3.9mm wide)	U3	296-19752-1-ND	Texas Inst MAX3232EIDR
39	1	IC, Single Inverter 74LVC1G04 SC70-05	U4	296-11600-1-ND	Texas Inst SN74LVC1G04DCKR
40	1	IC Voltage Regulator 3.3V 1A LDO, SOT-223	U5	497-1228-1-ND	ST Micro LD1117AS33TR
41	1	PolyZen 5.6V PPTC protected Zener SMD	U6	ZEN056V130A24LSCT-ND	Tyco ZEN056V130A24LS
42	1	Filter, EMI 35dB 10A 1MHz-1GHz SMD	U7	490-5052-1-ND	Murata BNX022-01L
43	1	IC Voltage Ref 2.5V 1% Micropower SOT-23	VR1	576-1047-1-ND	Micrel LM4040DYM3-2.5
44	1	Crystal 12.00MHz, 50ppm 20pF, HC-49US leaded	Y1	631-1105-ND	Fox FOXSLF/120-20
45	1	Crystal, 32768 Hz 12.5pF cylinder leaded	Y2	535-9033-1-ND	Abracon AB26TRB-32.768KHZ-T
		J-Link On-Board Circuitry…			
46	10	Capacitor, Ceramic 100nF -20% / +80% 25V Y5V 0603	C48-C53, C55-C58	490-1575-1-ND	Murata GRM188F51E104ZA01D
47	1	Capacitor, Ceramic 33pF 5% 50V C0G 0603	C59	490-1415-1-ND	Murata GRM1885C1H330JA01D
48	2	Capacitor, Ceramic 15pF 5% 50V C0G 0603	C60,C61	490-1407-1-ND	Murata GRM1885C1H150JA01D
49	2	Capacitor, Ceramic 10pF 5% 50V C0G 0603	C62,C63	490-1403-1-ND	Murata GRM1885C1H100JA01D
50	1	Capacitor, Ceramic 1nF 20% 50V X7R 0603	C64	490-1495-1-ND	Murata GRM188R71H102MA01D
51	1	Capacitor, Ceramic 10nF 10% 50V X7R 0603	C65	490-1512-1-ND	Murata GRM188R71H103KA01D
52	1	Capacitor, Ceramic 4.7uF -20% / +80% 6.3V 0603	C47	587-1313-1-ND	Taiyo Yuden JMK212F475ZD-T
53	1	Ferrite Bead, 220 Ohm @ 100MHz 300mA DC 0805	FB2	SAME AS FB2 ABOVE	
54	1	Solder Jumper	JP2	LEAVE OPEN	
55	1	Connector, Receptacle USB Mini B Rt-Angle PCB Mount	J8	SAME AS J2 ABOVE	
56	1	LED Green 0805	LED2	SAME AS LED1 ABOVE	
57	1	Resistor, 0 ohm 1/10W 0603	R30	P0.0GCT-ND	Panasonic ERJ-3GEY0R00V
58	1	Resistor, 220 ohm 5% 1/10W 0603	R31	P220GCT-ND	Panasonic ERJ-3GEYJ221V
59	2	Resistor, 1.5K ohm 5% 1/10W 0603	R32,R41	P1.5KGCT-ND	Panasonic ERJ-3GEYJ152V
60	1	Resistor, 47K ohm 5% 1/10W 0603	R33	P47KGCT-ND	Panasonic ERJ-3GEYJ473V
61	1	Resistor, 100 ohm 5% 1/10W 0603	R34,R35,R37 R39	P100GCT-ND	Panasonic ERJ-3GEYJ101V
62	1	Resistor, 27 ohm 5% 1/10W 0603	R36,R38	P27GCT-ND	Panasonic ERJ-3GEYJ270V
63	1	Resistor, 300 ohm 5% 1/10W 0603	R40	P300GCT-ND	Panasonic ERJ-3GEYJ301V
64	1	IC AT91SAM7S64 64-PQFP programmed by Segger	U8	from Segger	
65	1	Crystal 18.432MHz, 30ppm 10pF, SMD 3.2x2.5 mm	Y3	535-10909-1-ND	Abracon ABM8G-18.432MHZ-4Y-T3



LOWER CIRCUIT BOARD

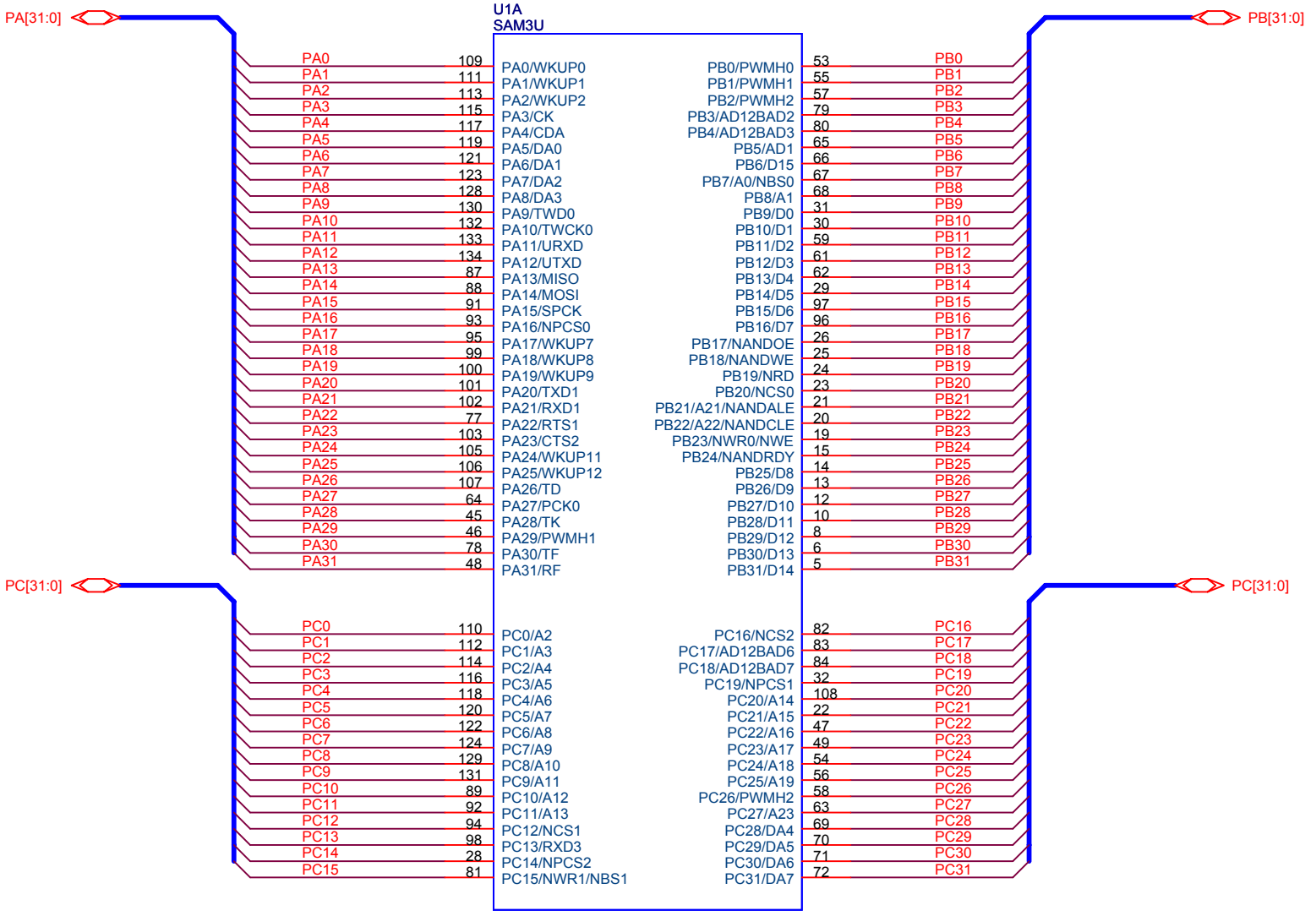


STACKING UPPER CIRCUIT BOARD

J3,J4 & J5 ARE DUAL-ROW STACKING RECEPTACLES (LOWER BOARD) AND HEADERS (UPPER BOARD).

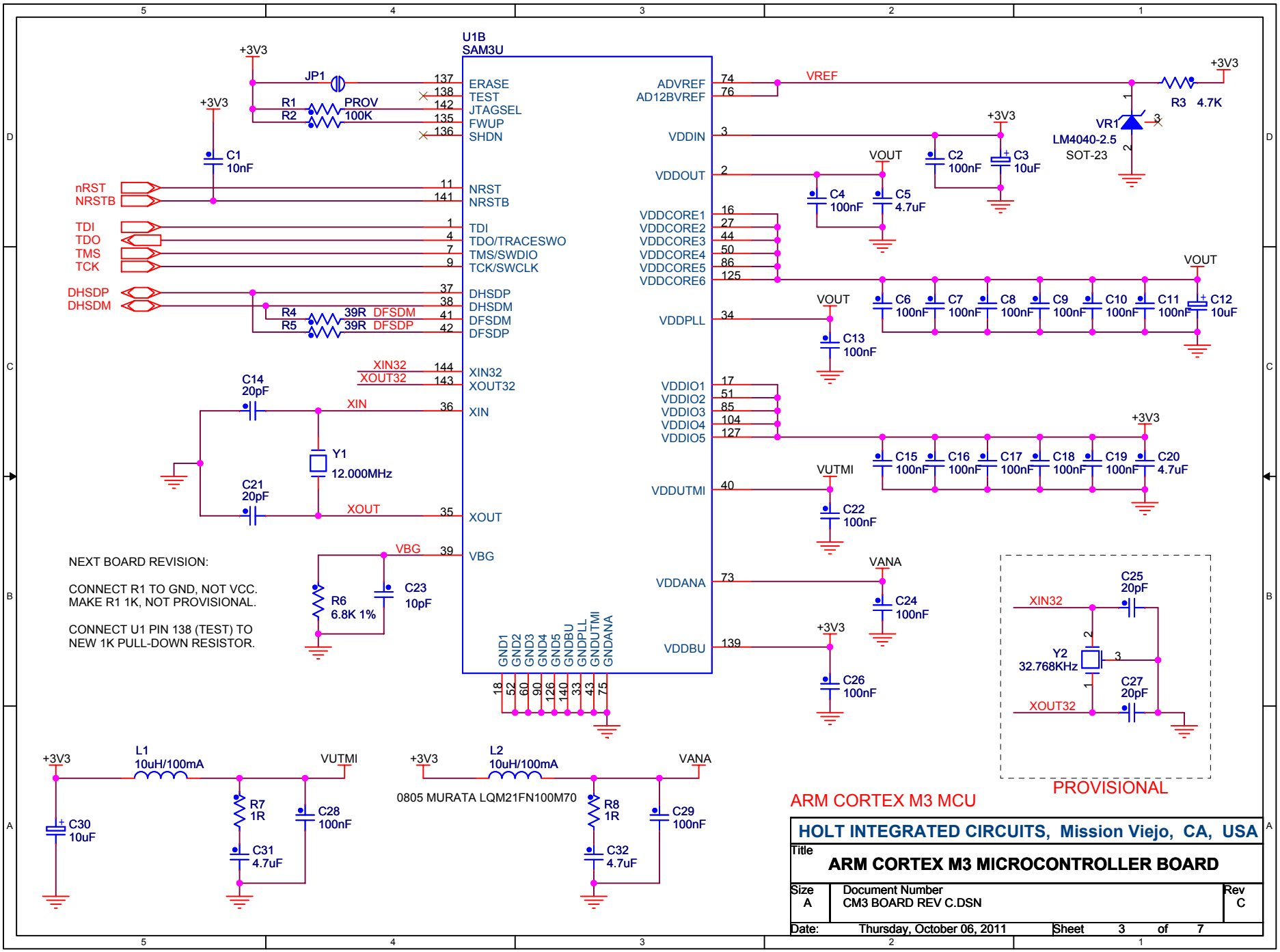
HOLT INTEGRATED CIRCUITS, Mission Viejo, CA, USA		
Title: ARM CORTEX M3 MICROCONTROLLER BOARD		
Size: A	Document Number: CM3 BOARD REV C.DSN	Rev: C
Date: Thursday, October 06, 2011	Sheet 1 of 7	

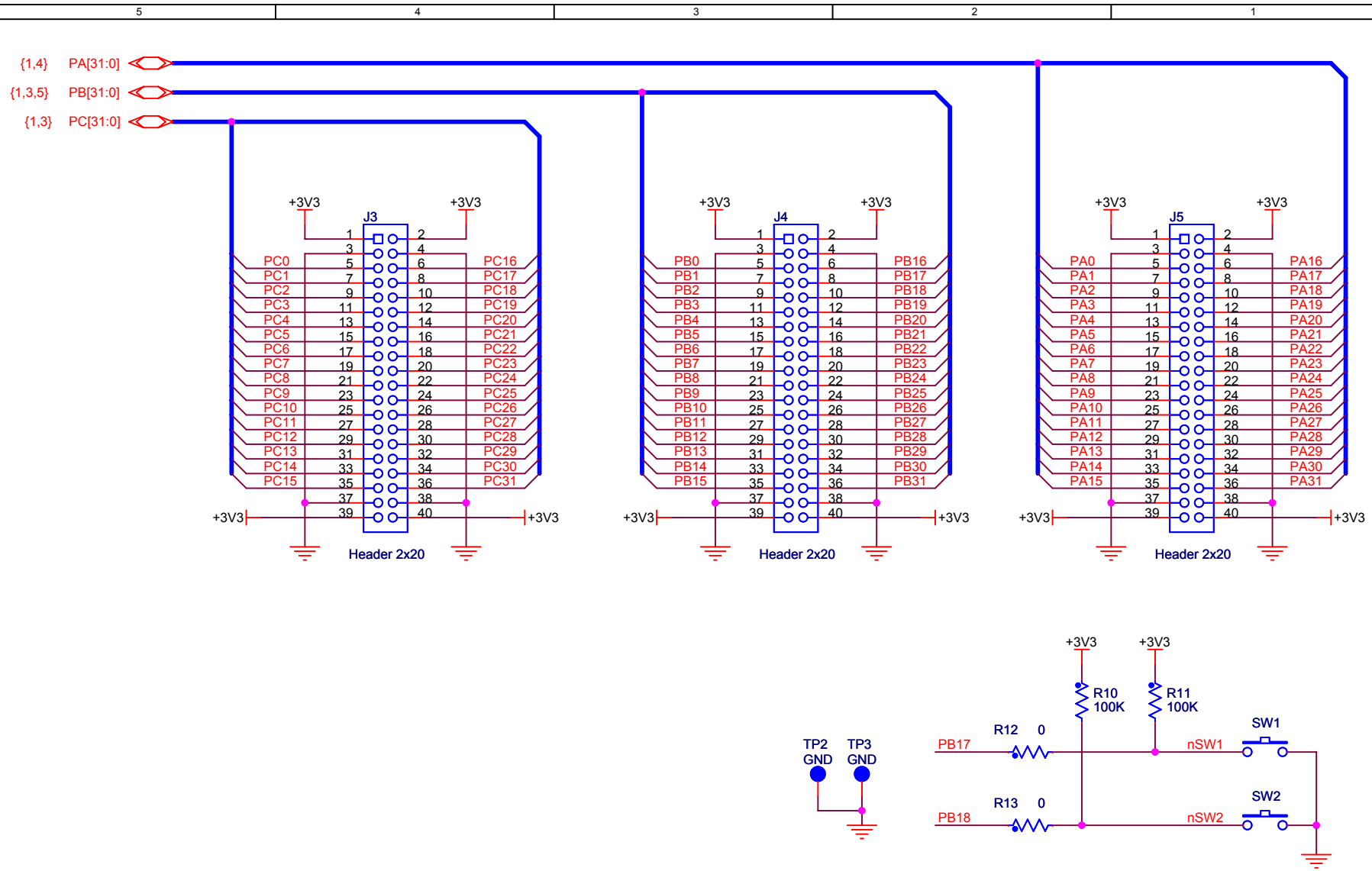
BUTTONS



ARM CORTEX M3 PIO

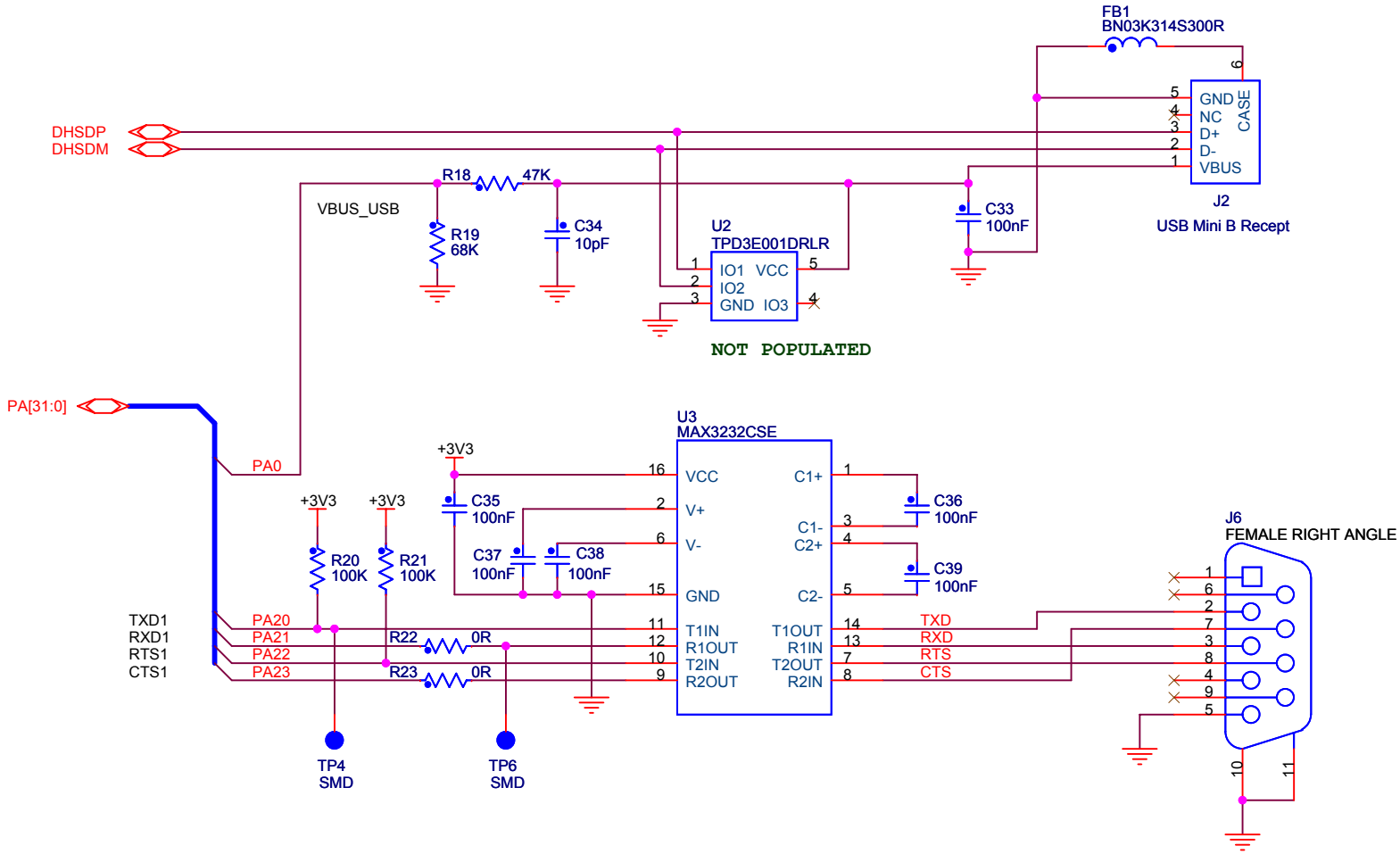
HOLT INTEGRATED CIRCUITS, Mission Viejo, CA, USA		
Title: ARM CORTEX M3 MICROCONTROLLER BOARD		
Size: A	Document Number: CM3 BOARD REV C.DSN	Rev: C
Date: Thursday, October 06, 2011	Sheet 2 of 7	


U1B
SAM3U
+3V3
JP1
R1
R2
PROV
100K
C1
10nF
nRST
NRSTB
TDI
TDO
TMS
TCK
DHSDP
DHSDM
R4
R5
39R DFSDM
39R DFSDP
C14
20pF
C21
20pF
XIN32
XOUT32
XIN
XOUT
Y1
12.000MHz
VBG
R6
6.8K 1%
C23
10pF
NEXT BOARD REVISION:
CONNECT R1 TO GND, NOT VCC.
MAKE R1 1K, NOT PROVISIONAL.
CONNECT U1 PIN 138 (TEST) TO
NEW 1K PULL-DOWN RESISTOR.
ERASE
TEST
JTAGSEL
FWUP
SHDN
NRST
NRSTB
TDI
TDO/TRACESWO
TMS/SWDIO
TCK/SWCLK
DHSDP
DHSDM
DFSDM
DFSDP
XIN32
XOUT32
XIN
XOUT
VBG
GND1
GND2
GND3
GND4
GND5
GNDBU
GNDPLL
GNDUTMI
GNDANA
ADVREF
AD12BVREF
VDDIN
VDDOUT
VDDCORE1
VDDCORE2
VDDCORE3
VDDCORE4
VDDCORE5
VDDCORE6
VDDPLL
VDDIO1
VDDIO2
VDDIO3
VDDIO4
VDDIO5
VDDUTMI
VDDANA
VDDBU
VREF
R3 4.7K
VR1
LM4040-2.5
SOT-23
C2 100nF
C3 10uF
C4 100nF
C5 4.7uF
C6 100nF
C7 100nF
C8 100nF
C9 100nF
C10 100nF
C11 100nF
C12 10uF
C13 100nF
C15 100nF
C16 100nF
C17 100nF
C18 100nF
C19 100nF
C20 4.7uF
C22 100nF
C24 100nF
C26 100nF
VOUT
VUTMI
VANA
L1
10uH/100mA
L2
10uH/100mA
0805 MURATA LQM21FN100M70
C30
10uF
R7
1R
C28
100nF
C31
4.7uF
R8
1R
C29
100nF
C32
4.7uF
C25
20pF
Y2
32.768KHz
C27
20pF
PROVISIONAL
ARM CORTEX M3 MCU
HOLT INTEGRATED CIRCUITS, Mission Viejo, CA, USA
Title
ARM CORTEX M3 MICROCONTROLLER BOARD
Size
A
Document Number
CM3 BOARD REV C.DSN
Rev
C
Date:
Thursday, October 06, 2011
Sheet
3
of
7



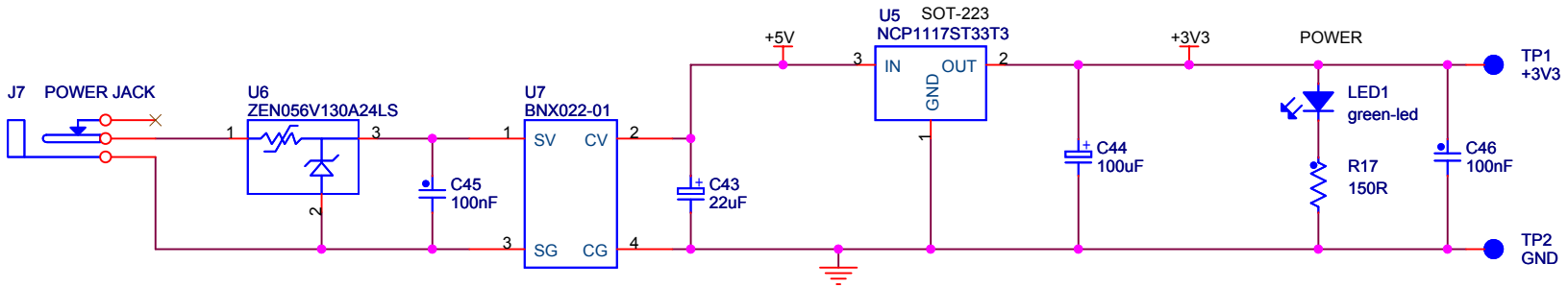
BOARD I/O HEADERS, BUTTONS

HOLT INTEGRATED CIRCUITS, Mission Viejo, CA, USA		
Title: ARM CORTEX M3 MICROCONTROLLER BOARD		
Size: A	Document Number: CM3 BOARD REV C.DSN	Rev: C
Date: Thursday, October 06, 2011	Sheet 4 of 7	



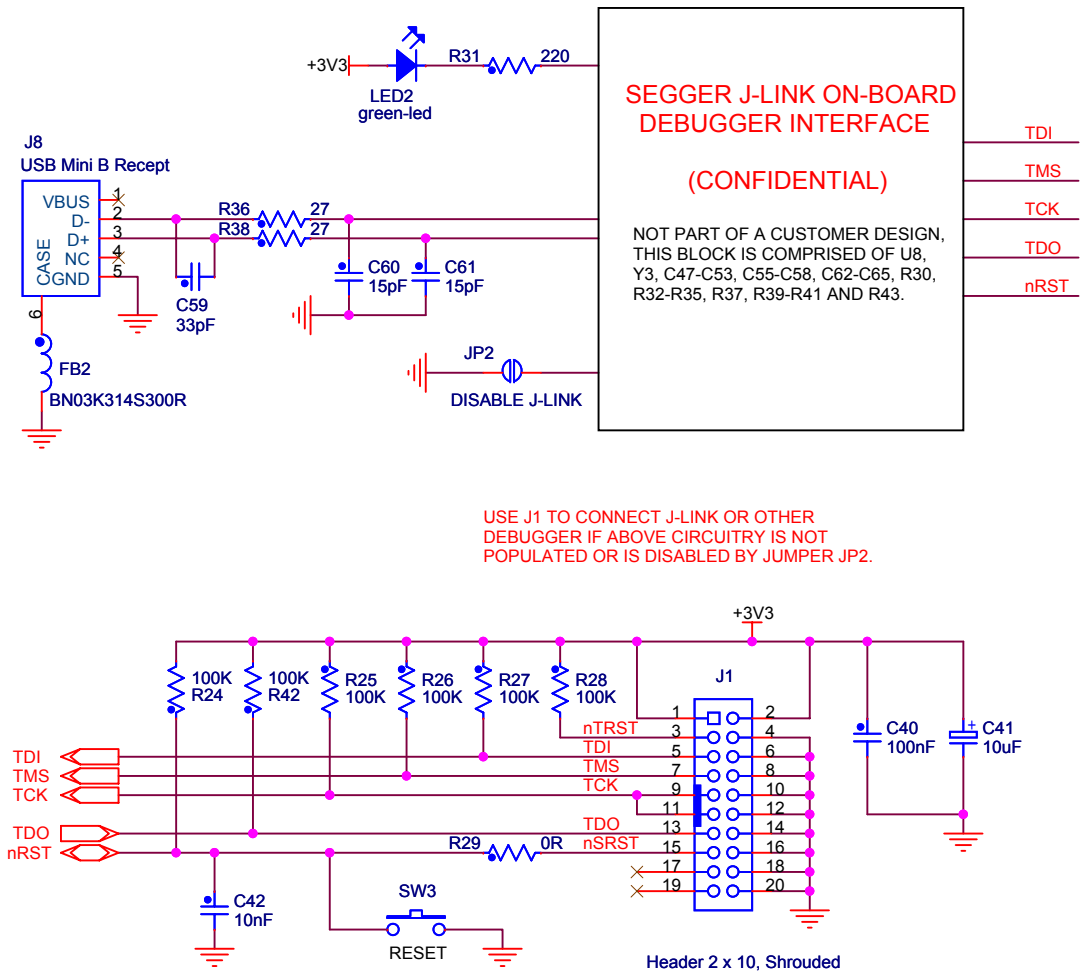
USB & RS-232 SERIAL

HOLT INTEGRATED CIRCUITS, Mission Viejo, CA, USA		
Title: ARM CORTEX M3 MICROCONTROLLER BOARD		
Size: A	Document Number: CM3 BOARD REV C.DSN	Rev: C
Date: Thursday, October 06, 2011	Sheet 5 of 7	



POWER SUPPLY

HOLT INTEGRATED CIRCUITS, Mission Viejo, CA, USA		
Title: ARM CORTEX M3 MICROCONTROLLER BOARD		
Size: A	Document Number: CM3 BOARD REV C.DSN	Rev: C
Date: Thursday, October 06, 2011	Sheet 6 of 7	



DEBUG INTERFACE

HOLT INTEGRATED CIRCUITS, Mission Viejo, CA, USA		
Title: ARM CORTEX M3 MICROCONTROLLER BOARD		
Size: A	Document Number: CM3 BOARD REV C.DSN	Rev: C
Date: Thursday, October 06, 2011	Sheet 7 of 7	