参考資料

Application Note

MSPM0 — SPI を使用して AM62x と AM62L に ADC を接続

Divyansh Mittal, Anshu Madwesh, Yashraj Motwani, Anil Swargam, Anshu Choudhary, Krunal Bhargav, Alisa Thomas

概要

このアプリケーション ノートは、高速 ADC データ転送をサポートするためのシリアル ペリフェラル インターフェイス (SPI) を使用して、MSMP0 に搭載されている ADC を、コントローラである AM62x と AM62L として動作する 2 個のプロセッサに統合する方法について説明します。AM62x は、最大 4 個の Arm Cortex A53 プロセッサと 1 個の Arm Cortex M4F コアを搭載した異種プロセッサです。AM62L は AM62x の軽量版であり、Arm Cortex-M4 マイコン コアを搭載せず、2 個の Arm Cortex-A53 コアを特徴としています。AM62x にはオンボード ADC が搭載されていませんが、AM62L は約 10 ENOB ADC を搭載しています。このドキュメントでは、MSMP0 マイコンの ADC を AM62x と AM62L に統合するプロセスをご紹介します。この追加により、AM62x の ADC を有効にでき、AM62L 向けに高分解能オプションを提供できます。MSPM0 マイコンには 1 個のマルチチャネル ADC が搭載されています。この ADC を使うと、複数のアナログ信号を監視し、任意のデジタル信号を送信して、SPI 経由で SoC に送信できます。このドキュメントでは、全体的なデータ フロー、ハードウェアとソフトウェアの設定、アプリケーション コードを実行する手順、予期される結果についてさらに詳細に説明します。

目次

商標

1 概要

1.1 SPI トランザクション データフロー

MSPM0L130x マイコンに搭載されている ADC を構成し、AM62x SK/AM62L マイクロプロセッサとの SPI インターフェイスを確立しています。ここでは、プロセッサ AM62x/AM62L をコントローラとして、MSPM0L130x をペリフェラルとして構成しています。任意のチャネルから ADC データを取得するため、コントローラは TX バッファ内の対応するコマンドを用いて、SPI トランザクションを開始することができます。コントローラからコマンドを受信すると、ペリフェラルは、要求されたチャネルに基づいて、TX バッファにロードされた ADC データの送信を開始します。コントローラは予期した数のバイトをペリフェラルから受信し、そのトランザクションを終了します。ペリフェラルは、ADC データ値の読み取りと更新を継続的に行います。これらの更新の頻度は、ADC をトリガするために使用されるタイマによって異なります。[1]

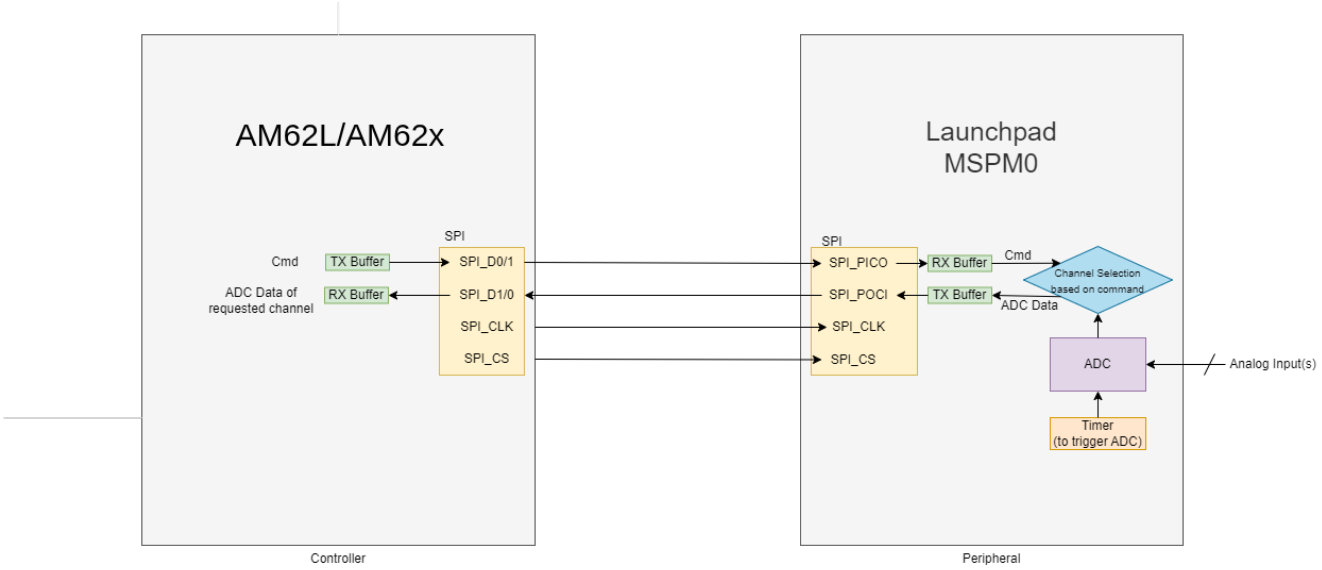


図 1-1. コントローラ (SK-AM62x または AM62L 評価基板) とペリフェラル (LPMSPM0L130x) 間の全体的なデータフロー

マルチチャネル モードを使用する場合のパイプライン方式全二重 SPI:

全二重 SPI モードでは、同じクロック サイクル セットを介してデータが同時に送受信されます。したがって、マルチチャネル ADC を使用する場合、コントローラからコマンドが送信されると、その最後のコマンドに対応する ADC データを同時に受信します。

このアプリケーションを実行する手順は次のとおりです。

1. コントローラ SK-AM62x/AM62L 評価基板とペリフェラル LP-MSPM0L130x 間の接続を含むハードウェア設定。
2. ワンタイム実行前ステップを含むソフトウェア設定。
3. 両方のボードでアプリケーションを実行して、SPI トランザクションを有効化。
4. 結果の分析。
5. システム性能の分析と消費電力の推定。

[1] 注:「マスター」と「スレーブ」という用語の使用は、「MOSI/MISO」という呼称と同様に、廃止の方向で検討されています。これらの用語は、それぞれ「コントローラ」および「ペリフェラル」および「PICO/POCI」に置き換えられます。

1.2 AM62x と AM62L プロセッサ

AM62x プロセッサ

図 1-2 に示されている AM62x Sitara マイクロプロセッサは、さまざまな組込みアプリケーション向けの設計を採用した異種プロセッサです。SPI は、A53 コアの MAIN ドメインによって有効化できます。図 1-2 に、AM62x の概略ブロック図を示します。

詳細については、『AM62x Sitara プロセッサ データシート』を参照してください。

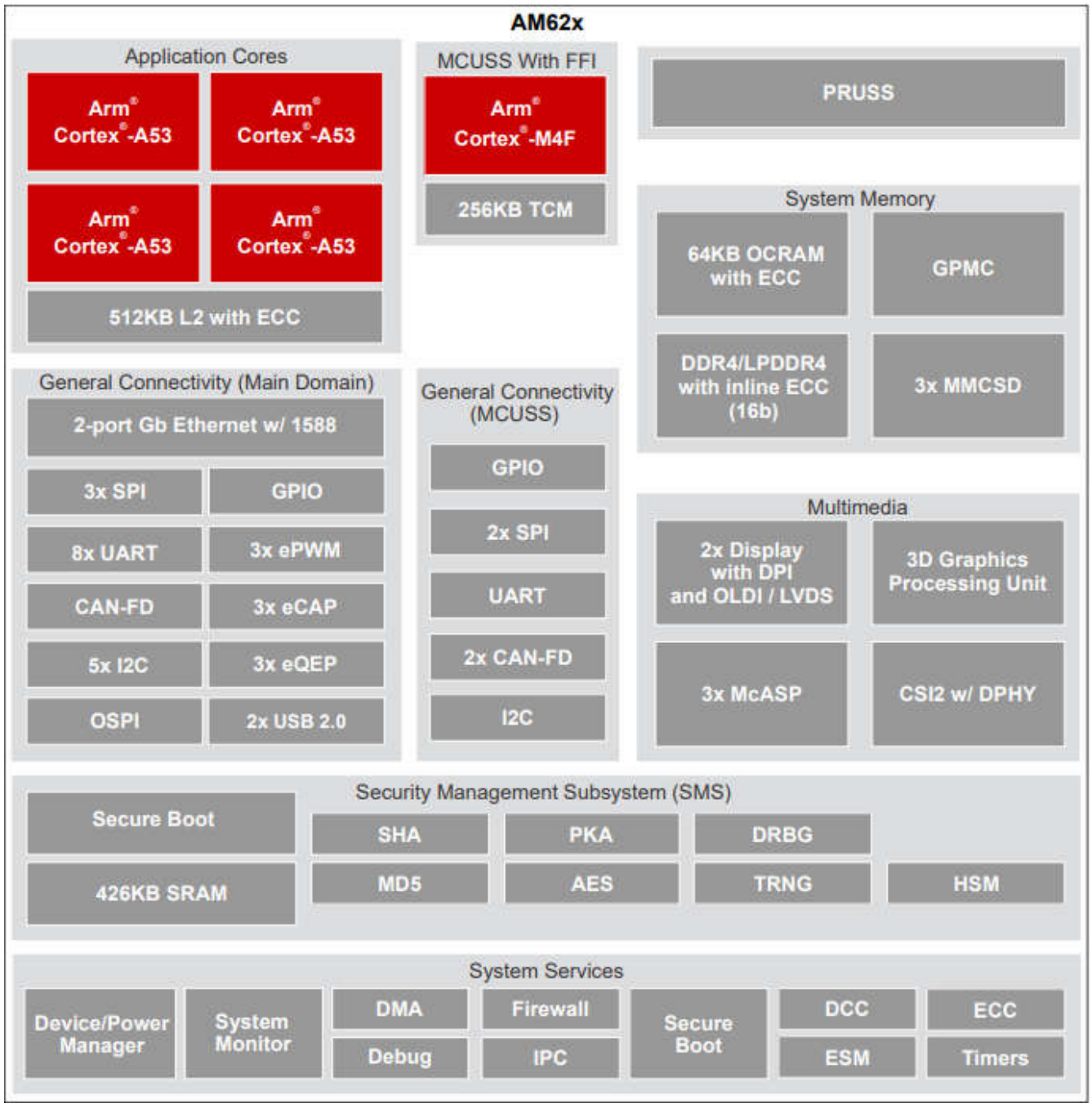


図 1-2. AM62x の概略ブロック図

AM62L プロセッサ

図 1-3 に示されている AM62L Sitara マイクロプロセッサは、AM6x ファミリの中でも性能が最適化された低コストのプロセッサです。SPI は、A53 コアの MAIN ドメインによって有効化できます。図 1-3 に、AM62L の概略ブロック図を示します。

詳細については、『AM62L Sitara プロセッサ データシート』を参照してください。

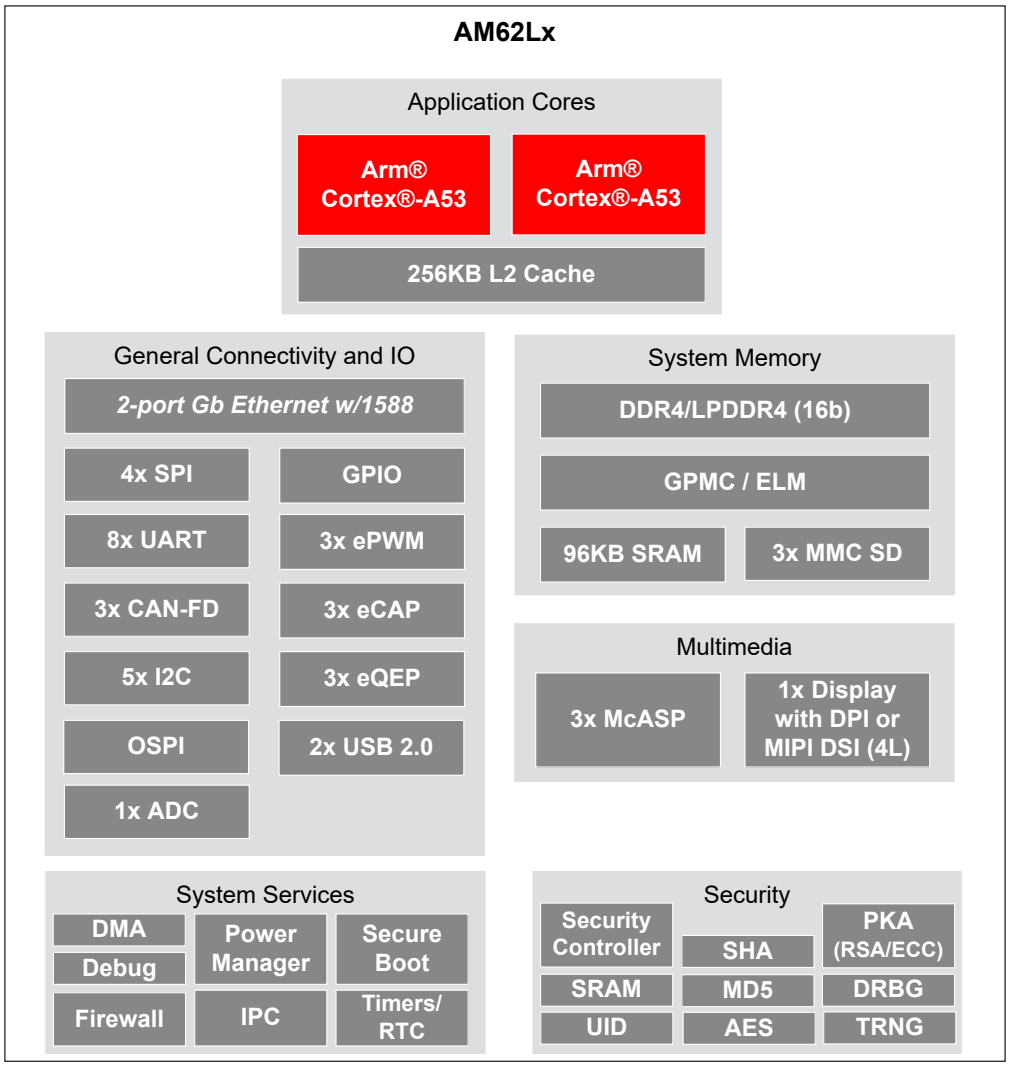


図 1-3. AM62L の概略ブロック図

1.3 MSPM0L130x マイクロコントローラ

図 1-4 に示されている MSPM0L130x マイコンは、使いやすい評価基板 (EVM) です。

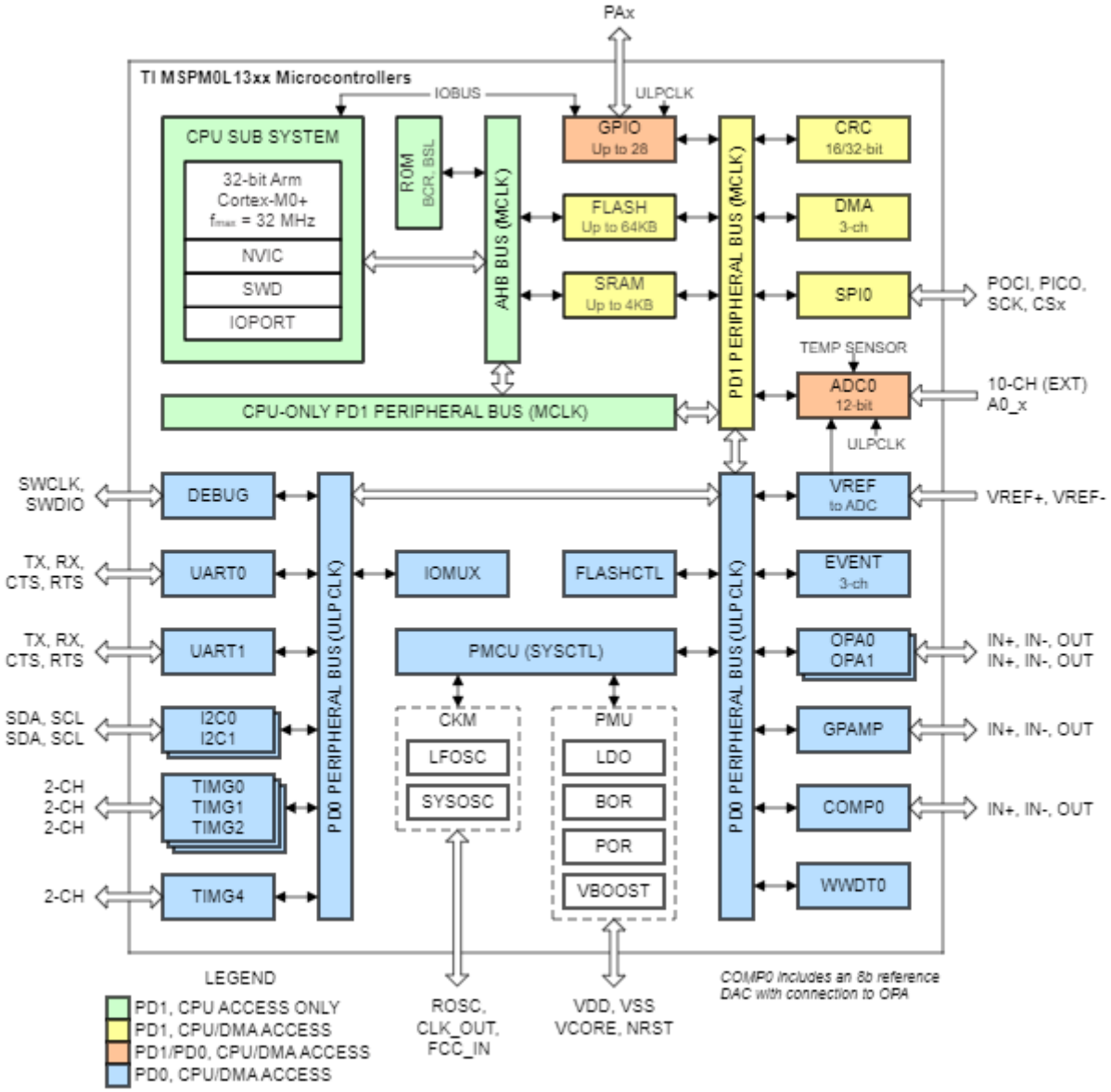


図 1-4. MSPM0L130x の概略ブロック図

M0L デバイスのメイン コンピューティング サブシステムおよびインターフェイス サブシステムは次のとおりです。

- Arm Cortex-M0+ コア:このプラットフォームは、最大 32MHz の周波数で動作可能です。このコスト最適化済みのマイコンは、高性能アナログ ペリフェラルを統合しています。
- このオンボード ADC は、12、10、8 ビットの高速な A/D 変換をサポートしています。12 ビットの SAR コア、サンプルおよび変換モード制御、最大 4 個の独立した変換および制御バッファを実装しており、12 ビットの分解能で 1.68Msps の変換レートを実現しています。
- 16Mbits/s の速度で動作できる SPI モジュールが搭載されています。

詳細については、『MSPM0L130x マイコン』データシートを参照してください。

2 ハードウェア設定

アプリケーション コードを実行するには、以下のケーブル接続を行う必要があります。これらの接続用に選択されるピンは、特定の SPI チャネル用であることに注意してください。SPI チャネルまたはピン多重化に変更を加えた場合、対応するピンはデータシートを介して確認してから、使用する必要があります。

2.1 AM62x

AM62x は 2 個のドメイン (MAIN - 4xA53 と MCU-1xM4F) で構成されています。ハードウェア設定は、両方のコアに適用されます。

2.1.1 A53 コアのハードウェア設定

A53 コアを使用する場合、SK-AM62x の SPI 用のペリフェラル ピンはユーザー拡張ヘッダ内にあります。図 2-1 に、ハードウェア設定を示します。

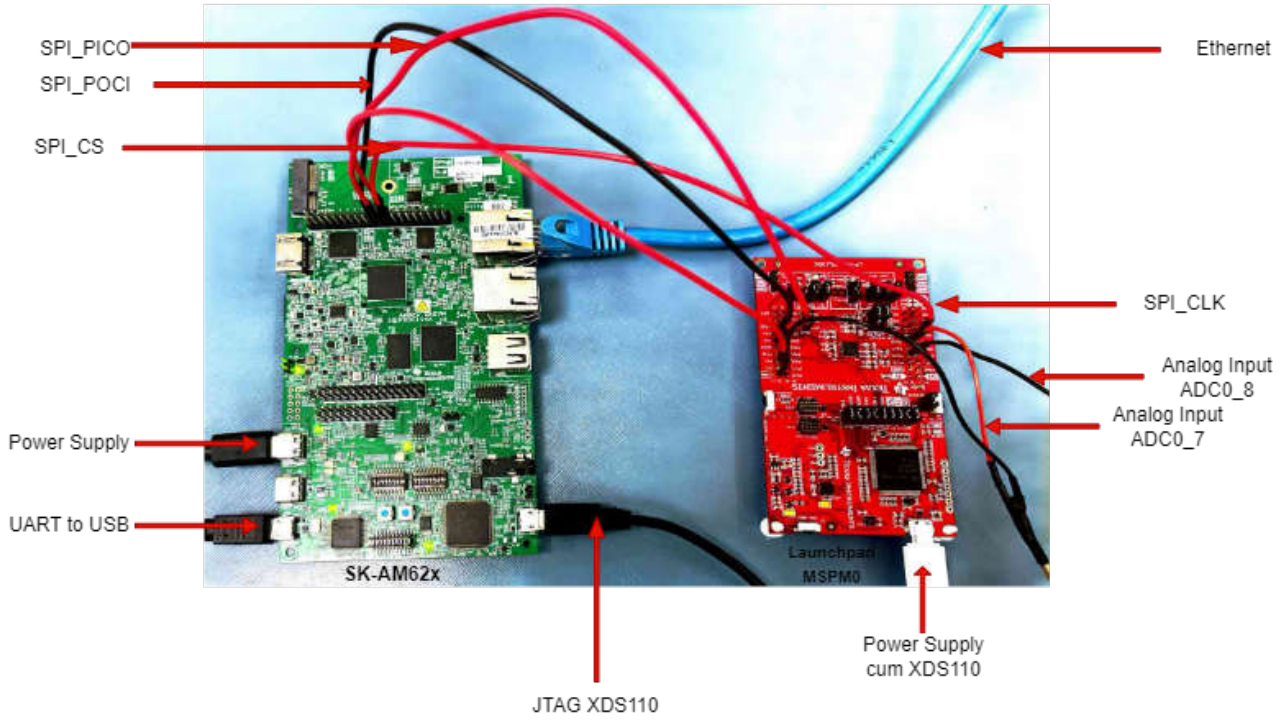


図 2-1. SPI 通信用 SK-AM62x (A53 コア) と LP-MSPM0L1306 の間のケーブル接続。

- SK-AM62x の場合:
  - Type-C 電源を接続します。
  - JTAG XDS110 用の UART-to-USB と USB をコンピュータに接続します。
- LP-MSPM0 の場合:
  - 電源カム XDS110 をコンピュータに接続します。
  - Launchpad MSPM0 の J3_PA18 (ADC0_7) にアナログ信号入力を接続します。
  - 必要に応じて、Launchpad MSPM0 のアナログ信号入力を J3_PA16 (ADC0_8) に接続します。
- SK-AM62x から LP-MSPM0 への接続の場合
  - SK-AM62x ユーザー拡張コネクタのピン 19 (B13: SPI0_D0) を、Launchpad MSPM0 の J2_PA4 (SPI_POCI) に接続します。
  - SK-AM62x ユーザー拡張コネクタのピン 21 (B14: SPI0_D1) を、Launchpad MSPM0 の J2_PA5 (SPI_PICO) に接続します。

- SK-AM62x ユーザー拡張コネクタのピン 23 (A14: SPI0_CLK) を、Launchpad MSPM0 の J1_PA6 (SPI_CLK) に接続します。
- SK-AM62x ユーザー拡張コネクタのピン 24 (A13: SPI0_CS0) を、Launchpad MSPM0 の J2_PA3 (SPI_CS (PWM)) に接続します。

2.1.2 M4F コアのハードウェア設定

M4F コアを使用する場合、SK-AM62x の SPI 用のペリフェラル ピンはマイコン ヘッダ内にあります。図 2-2 に、ハードウェア設定を示します。

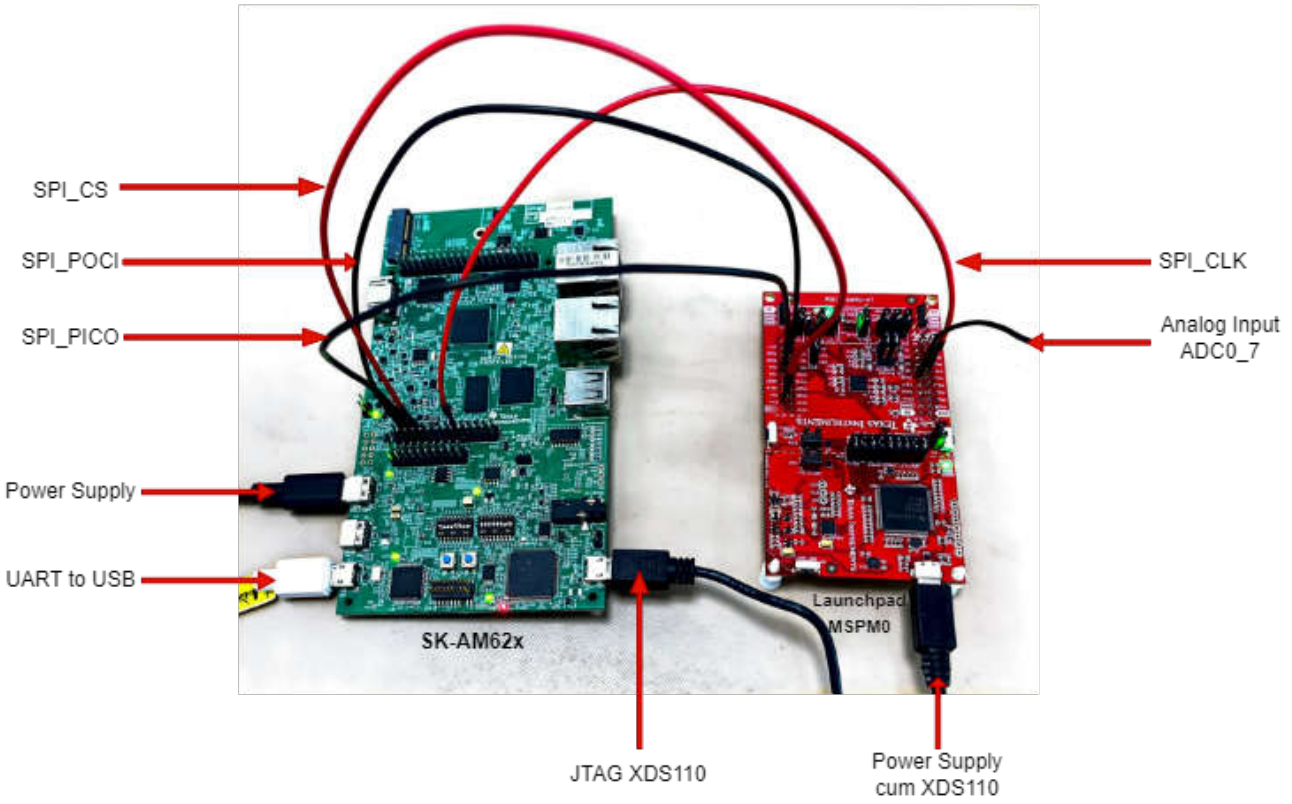


図 2-2. SPI 通信用 SK-AM62x (M4F コア) と LP-MSPM0L1306 の間のケーブル接続。

- SK-AM62x の場合:
  - Type-C 電源を接続します。
  - JTAG XDS110 用の UART-to-USB と USB をコンピュータに接続します。
- LP-MSPM0 の場合:
  - 電源カム XDS110 をコンピュータに接続します。
  - アナログ信号入力を LP-MSPM0 の J3_PA18 (ADC0_7) に接続します。
  - 必要に応じて、Launchpad MSPM0 のアナログ信号入力を J3_PA16 (ADC0_8) に接続します。
- SK-AM62x から LP-MSPM0 への接続の場合:
  - SK-AM62x マイコン ヘッダにあるピン 4 (C9: MCU_SPI0_D1) を、LP-MSPM0 の J2_PA4 (SPI_POCI) に接続します。
  - SK-AM62x マイコン ヘッダにあるピン 6 (D9: MCU_SPI0_D0) を、LP-MSPM0 の J2_PA5 (SPI_PICO) に接続します。
  - SK-AM62x マイコン ヘッダにあるピン 8 (B8: MCU_SPI0_CS1) を、LP-MSPM0 の J2_PA3 (SPI_CS(PWM)) に接続します。

– SK-AM62x マイコン ヘッダにあるピン 18 (A7: MCU_SPI0_CLK) を、LP-MSPM0 の J1_PA6 (SPI_CLK) に接続します。

2.2 AM62L

AM62L は単一の MAIN ドメインのみ (2xA53 コア搭載) で構成されており、その内部に ADC を介した SPI データ転送のためのハードウェア構成が実装されています。ハードウェア設定は次のとおりです。

2.2.1 A53 コアのハードウェア設定

A53 コアを使用する場合、AM62L の SPI 用のペリフェラル ピンはユーザー拡張ヘッダにあります。図 2-3 に、ハードウェア設定を示します。

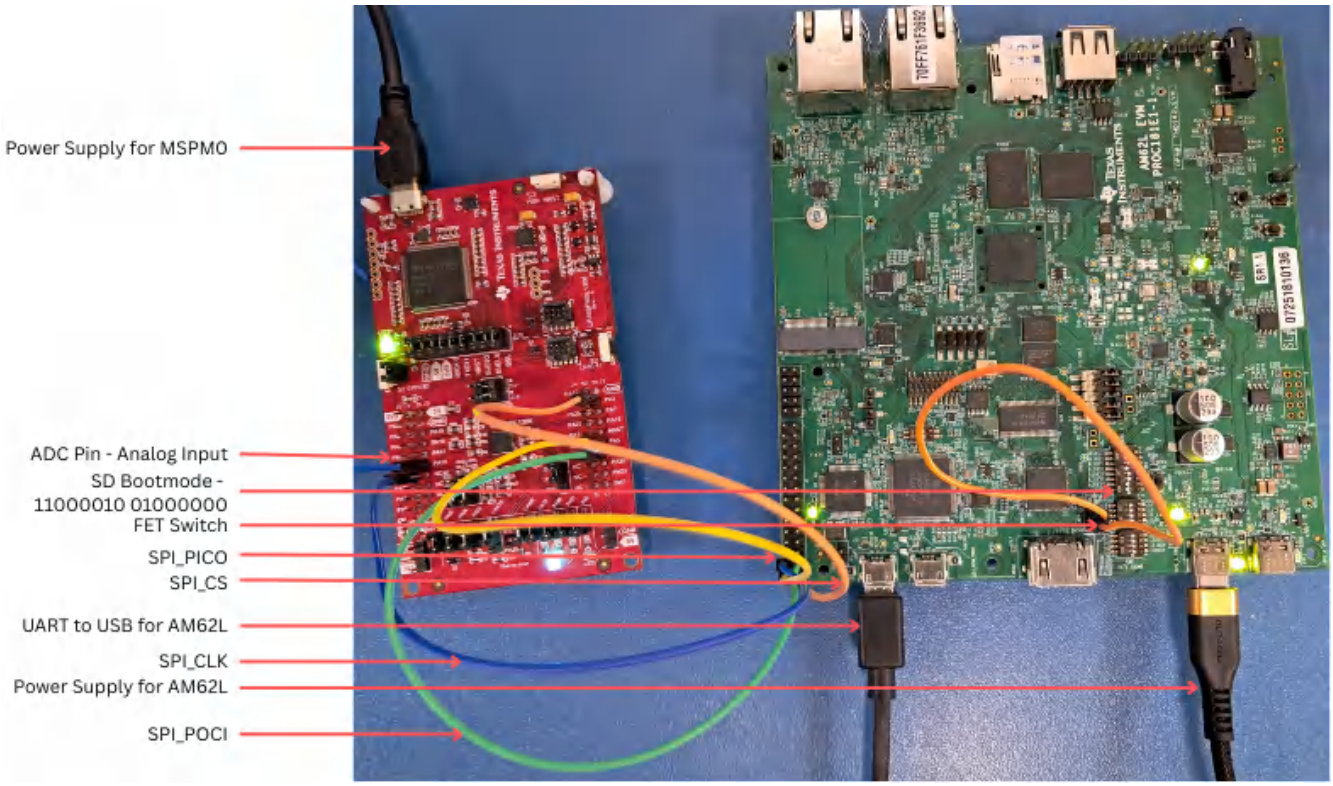


図 2-3. SPI 通信用の AM62L (A53 コア) と LP-MSPM0L1306 の間のケーブル接続。

- AM62L の場合:
  - Type-C 電源を接続します。
  - JTAG XDS110 用の UART-to-USB と USB をコンピュータに接続します。
- LP-MSPM0 の場合:
  - 電源カム XDS110 をコンピュータに接続します。
  - アナログ信号入力を LP-MSPM0 の J3_PA18 (ADC0_7) に接続します。
  - 必要に応じて、Launchpad MSPM0 のアナログ信号入力を J3_PA16 (ADC0_8) に接続します。
- AM62L 評価基板から MSPM0L への接続の場合
  - AM62L 評価基板ユーザー拡張コネクタのピン 28 (SPI1_D0_EXP) を、Launchpad MSPM0Lx の J2_PA4 (SPI_POCI) に接続します。
  - AM62L 評価基板ユーザー拡張コネクタのピン 29 (SPI1_D1_EXP) を、Launchpad MSPM0Lx の J2_PA5 (SPI_PICO) に接続します。

  - AM62L 評価基板ユーザー拡張コネクタのピン 27 (SPI1_CLK_EXP) を、Launchpad MSPM0Lx の J1_PA6 (SPI_CLK) に接続します。
  - AM62L 評価基板ユーザー拡張コネクタのピン 30 (SPI1_CS0_EXP) を Launchpad MSPM0Lx の J2_PA3 (SPI_CS (PWM)) に接続します。
- TMDS62LEVM FET スイッチ J29 は、SOC から HDMI または IO エキスパンダへの出力を制御します。なお、デフォルト設定は HDMI です。したがって、トグルする必要があります。この構成により、ブート時にアドレス 0x23 の I2C エクスパンダのピン 1 が自動的に High 出力状態に設定されます。

3 ソフトウェアの設定

AM62x または AM62L と MSPM0L1306 を設定する手順を以下に示します。

3.1 Beyond SDK GitHub リポジトリのクローニング

- Beyond SDK は、この実験に必要なファイルの一部を含む GitHub リポジトリです。
- ファイルは、*Beyond-SDK/collaterals/appnotes/slaaej0-MSPM0-ADC-Attach* にあります。
- コントローラを選択します (AM62x または AM62L)。
- アプリケーションに応じていずれかのフォルダを選択します (x_Byte_x_Channel_SPI)。
- AM62x コントローラの場合、A53 コア フォルダには C ファイルが含まれ、M4 コアには CCS プロジェクトが含まれています。
- ペリフェラルには、MSPM0 用の CCS プロジェクトがあります。
- GitHub リポジトリのクローンを作成する方法は次のとおりです。

```
HOST$ mkdir <Beyond-SDK-installation-path>
HOST$ cd <Beyond-SDK-installation-path>
HOST$ git clone https://github.com/TexasInstruments/Beyond-SDK.git
```

3.2 SK-AM62x ソフトウェア設定

3.2.1 A53 コア

- 『AM62x スターター キット評価基板クイック スタート ガイド』に掲載されている設定に従います。
- 以下の実験では、プロセッサ SDK Linux for AM62x バージョン 9.0 を使用しています。
- 次の手順に従ってカーネル デバイス ツリーを変更し、Linux で SPI ドライバを設定します。

1. k3-am625-sk.dts デバイス ツリー ファイルは、パス *<psdk-installation-path>/board-support/ti-linux-kernel/arch/arm64/boot/dts/ti* にあります。
2. 次のようにファイルを変更します。

Inside &main_pmx0{...} に以下を追加:

```
main_spi0_pins_default: main-spi0-pins-default {
        pinctrl-single,pins = <
             AM62X_IOPAD(0x01bc, PIN_OUTPUT, 0) /* (A14) SPI0_CLK */
             AM62X_IOPAD(0x01c0, PIN_INPUT, 0) /* (B13) SPI0_D0 */
             AM62X_IOPAD(0x01c4, PIN_OUTPUT, 0) /* (B14) SPI0_D1 */
             AM62X_IOPAD(0x01b4, PIN_OUTPUT, 0) /* (A13) SPI0_CS0 */
        >;
};
```

ファイルの末尾に以下を追加:

```
&main_spi0 {
     status = "okay";
     pinctrl-names = "default";
     pinctrl-0 = <&main_spi0_pins_default>;
     spidev@0 {
          spi-max-frequency = <16000000>;
          reg = <0>;
          compatible = "rohm,dh2228fv";
     };
};
```

- 『ユーザー ガイド — プロセッサ SDK AM62x』に記載されている手順を使用してカーネルを再コンパイルします。このページの手順を実行しながら、SPI カーネル ドライバで提供されているカーネル設定のセクションに従って、*menuconfig* を使用してカーネルをカスタマイズします。詳細については、以下の手順をご覧ください。

```
HOST$ cd <psdk-installaion-path>/board-support/ti-linux-kernel/
HOST$ make defconfig ti_arm64_prone.config
HOST$ make ARCH=arm64 menuconfig
```

```
      Device Drivers --->
          [*] SPI support
              <*> User mode SPI device driver support
#Save these changes to the .config file
HOST$ make Image dtbs modules
HOST$ sudo cp ./arch/arm64/boot/Image /media/<USER>/root/boot/
HOST$ sudo cp ./arch/arm64/boot/dts/ti/k3-am625-sk.dtb /media/root/boot/dtb/ti
HOST$ sudo -E env “PATH=$PATH” INSTALL_MOD_PATH=/media/<USER>/root make modules_install
HOST$ sync; sync
```

- ターゲットの C ファイルを SDK パスにコピーし、『Hello World プログラムのコンパイル例』で提供されている方法を使用して C プロジェクトファイルをコンパイルします。
- SD カードを SK-AM62x に再度挿入し、デバイスを再起動します。

3.2.2 M4F コア

- 以下に記載されている AM62x 向けの CCS (Code Composer Studio) の設定を行います。『AM62x の導入手順』。CCS のインストール中に、「Select Components」(部品の選択) ウィンドウで「MSPM0 32-bit Arm Cortex-M0+ General Purpose MCUs」(MSPM0 32 ビット Arm Cortex-M0+ 汎用マイコン) を選択します。
- CCS プロジェクトを Project Explorer にインポートします (File (ファイル) > Import (インポート) > Code Composer Studio > CCS Projects (CCS プロジェクト))。CCS プロジェクトは以下のパスにあります: *<Beyond-SDK-installation-path>/Beyond-SDK/collaterals/appnotes/slaaej0-MSPM0-ADC-Attach/am62x/<x_Byte_x_Channel_SPI>/Controller/AM62x-M4F_Core_MCU_Domain/*
- CCS の詳細なサポートについては、『CCS ユーザー ガイド』を参照してください。

3.3 AM62L ソフトウェア設定

3.3.1 A53 コア

- 『AM62L 導入ガイド』に記載されている手順に従います。
- 以下の実験では、プロセッサ SDK Linux for AM62L バージョン 11.0.15 を使用しています。
- AM62L SDK の ti-linux-kernel リポジトリ内で以下のパッチを適用し、Linux で SPI ドライバを設定します。
- 注:以下に示すパッチでは、HDMI は無効になっています。これは、TMDS62LEVM では、HDMI が拡張ヘッダと多重化されており、一度に使用できるのは 1 個だけであるためです。

```
diff --git a/arch/arm64/boot/dts/ti/k3-am62l3-evm.dts b/arch/arm64/boot/dts/ti/k3-am62l3-evm.dts
index 2dd056ce0538..6cf837274c6e 100644
--- a/arch/arm64/boot/dts/ti/k3-am62l3-evm.dts
+++ b/arch/arm64/boot/dts/ti/k3-am62l3-evm.dts
@@ -36,6 +36,7 @@ memory@80000000 {

     hdmi0: connector-hdmi {
         compatible = "hdmi-connector";
+        status = "disabled";
         label = "hdmi";
         type = "a";

@@ -389,6 +390,15 @@ AM62PX_IOPAD(0x0188, PIN_INPUT, 0) /* (A9) MCASP0_AXR1 */
         >;
     };

+    main_spi1_pins_default: main-spi1-pins-default {
+        pinctrl-single,pins = <
+            AM62LX_IOPAD(0x008c, PIN_OUTPUT, 4) /* (H22) SPI1_CLK */
+            AM62LX_IOPAD(0x0080, PIN_INPUT, 4) /* (K22) SPI1_D0 */
+            AM62LX_IOPAD(0x0084, PIN_OUTPUT, 4) /* (J23) SPI1_D1 */
+            AM62LX_IOPAD(0x0088, PIN_OUTPUT, 4) /* (K23) SPI1_CS0 */
+        >;
+    };
+
     pmic_irq_pins_default: pmic-irq-default-pins {
         pinctrl-single,pins = <
             AM62LX_IOPAD(0x01e8, PIN_INPUT, 0) /* (C8) EXTINTn */
@@ -410,6 +420,17 @@ &main_uart0 {
     bootph-all;
 };
```

```
+&main_spi1 {
+    status = "okay";
+    pinctrl-names = "default";
+    pinctrl-0 = <&main_spi1_pins_default>;
+    spidev@0 {
+        spi-max-frequency = <24000000>;
+        reg = <0>;
+        compatible = "rohm,dh2228fv";
+    };
+};
+
 &main_uart1 {
     pinctrl-names = "default";
     pinctrl-0 = <&main_uart1_pins_default>;
@@ -488,11 +509,14 @@ exp2: gpio@23 {

     sii9022: bridge-hdmi@3b {
         compatible = "sil,sii9022";
+        status="disabled";
         reg = <0x3b>;
         interrupt-parent = <&exp1>;
         interrupts = <16 IRQ_TYPE_EDGE_FALLING>;
         #sound-dai-cells = <0>;
         sil,i2s-data-lanes = < 0 >;
+        pinctrl-names = "default";
+        pinctrl-0 = <&main_dpi_pins_default>;
         bootph-all;

         ports {
@@ -774,8 +798,6 @@ partition@7fe0000 {

 &dss {
     status = "okay";
-    pinctrl-names = "default";
-    pinctrl-0 = <&main_dpi_pins_default>;
     bootph-all;
 };

diff --git a/arch/arm64/configs/defconfig b/arch/arm64/configs/defconfig
index 94dfc265a61c..157d0d62fb53 100644
--- a/arch/arm64/configs/defconfig
+++ b/arch/arm64/configs/defconfig
@@ -1828,3 +1828,5 @@ CONFIG_CORESIGHT_STM=m
 CONFIG_CORESIGHT_CPU_DEBUG=m
 CONFIG_CORESIGHT_CTI=m
 CONFIG_MEMTEST=y
+CONFIG_UHID=y
+CONFIG_SPI_SPIDEV=y
\ No newline at end of file
```

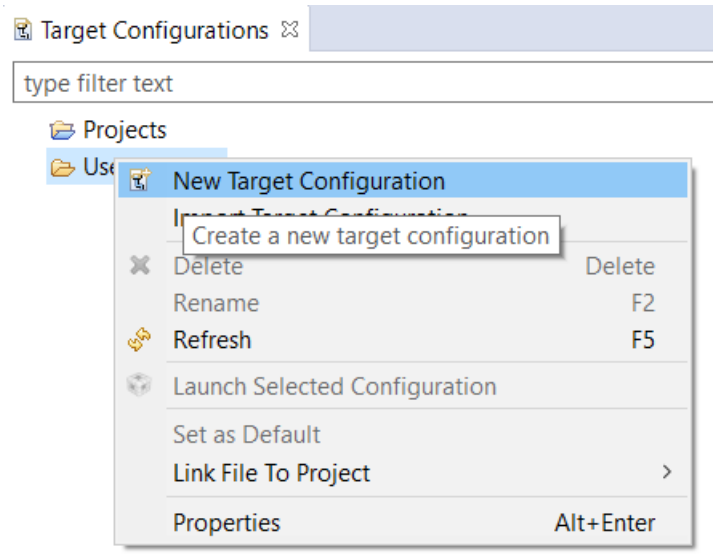
- Linux を再ビルドし、イメージ、dtbs、モジュールを SD/eMMC にインストールします。
- ここにリンクされている C ファイルを Linux のユーザー空間で使用してコンパイルします。
- マルチ バイトの例では、コントローラ側でデータを受信する方法は 2 つあります。1 つの方法は、MSPM0 からそれぞれ 2 クロック 8 ビットでデータを送信することです。もう 1 つのオプションは、ダブル期間の単一クロック バーストでデータ全体を送信することです。どちらの方法もテスト済みで、ここにリンクされている C ファイルから実装できます。
- **注:**より高い SPI 周波数を実現するために Linux ユーザー空間で SPI スクリプトを実行できない場合は、MSPM0 と AM62L のグランド ピンを接続します。

3.4 LP-MSPM0L130x ソフトウェア設定

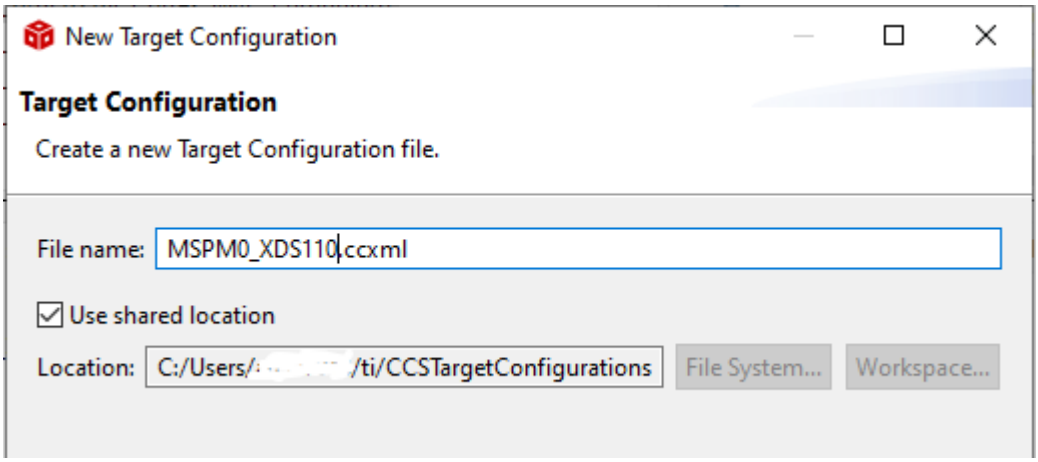
CCS (Code Composer Studio) は、MSPM0 Launchpad での開発に使用できます。CCS 関連の一般的なヘルプについては、『CCS ユーザー ガイド』を参照してください。MSPM0 デバイスでの開発を可能にするには、CCS のインストール中に「Select Components」(部品の選択) ウィンドウにある「MSPM0 32-bit Arm Cortex-M0+ General Purpose MCUs」(MSPM0 32 ビット Arm Cortex-M0+ 汎用マイコン) を選択します。

MSPM0 の新しいターゲット構成を追加するには、次の手順に従います。

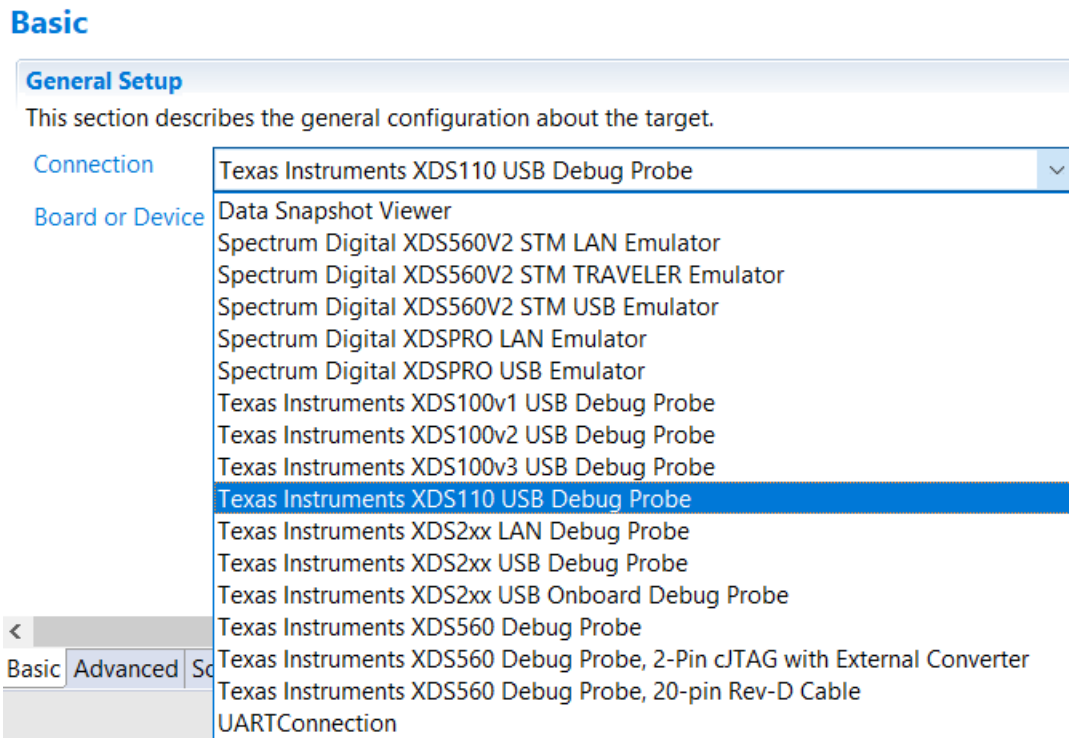
- 新しいターゲット構成を作成します。



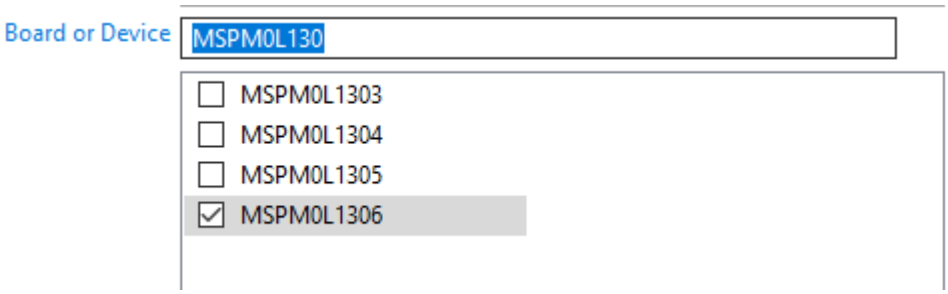
- 新しいターゲット構成に適切な名前を付けます (通常は {soc name}_{JTAG type})



- 接続として「XDS110 USB Debug Probe」(XDS110 USB デバッグ プローブ) を選択します。



- 「Board or Device」(ボードまたはデバイス) で「MSPM0L130」と入力し、「MSPM0L1306」を選択します。



- 「Save」(保存) をクリックして、新しく作成したターゲット構成を保存します。

4 実行手順

このセクションでは、コントローラとペリフェラルの両方でプロジェクトを実行する手順について説明します。

4.1 LP-MSPM0L130x でプロジェクトを実行

ターゲット構成を作成した後で、MSPM0 バイナリを構築してオンチップ フラッシュに書き込むことができます。

1. CCS プロジェクトをワークスペースにインポートします。プロジェクトへのパスは以下のとおりです。
   a. *<Beyond-SDK-installation-path>/Beyond-SDK/collaterals/appnotes/slaaej0-MSPM0-ADC-Attach/<controller>/<x_Byte_x_Channel_SPI>/Peripheral/MSPM0/*
2. インポートした CCS プロジェクトをビルドします。
   a. 「**Duplicate Name of GPIO Pin**」(GPIO ピン名の重複) エラーが発生した場合、次の手順を実行します。
      i. .syscfg ファイル拡張子で示されるシステム構成ファイルを開きます。
      ii. ADC12 構成を開きます。
      iii. 「Pin Configuration」(ピン構成) セクションを参照し、ADC12 チャネル 7 のピンを開きます
      iv. ピン名を「ti_driverlib_gpio_GPIOPinGeneric0」から「ti_driverlib_gpio_GPIOPinGeneric6」に変更します。
      v. 変更を保存し、プロジェクトを再ビルドします。
3. ビルドされたバイナリは、*<Beyond-SDK-installation-path>/.../MSPM0/Debug/<file-name>.out* にあります。
4. 「Target Configurations」(ターゲット構成) ウィンドウで、「*MSPM0_XDS110.ccxml*」を右クリックします。
5. 「Launch Selected Configuration」(選択した構成を起動) を選択します。
6. 「Debug」(デバッグ) ウィンドウで「Texas Instruments XDS110 USB Debug Probe_0/CORTEX_M0P」(テキサス インスツルメンツ XDS110 USB デバッグ プローブ_0/CORTEX_M0P) をクリックします
7. 「Run」(実行) ->「Connect Target」(ターゲットの接続) の順に選択します
8. 「Run」(実行) ->「Reset」(リセット) ->「Subsystem Reset」(サブシステム リセット) の順に選択します。
9. 「Run」(実行) ->「Load」(ロード) ->「Load Program」(プログラムのロード) の順に選択します。
10. MSPM0 プロジェクトのバイナリを参照し、「OK」をクリックします。
11. これにより、バイナリでフラッシュに書き込まれます。
12. 「Run」(実行) ->「Resume」(再開) の順に選択します。

新しい MSPM0 SDK バージョンでは、ビルド エラーが発生する可能性があることに注意してください。これを解決するには、CCS の Resource Explorer を使用し、新しい「spi_peripheral_echo_interrupts_LP_MSPM0L1306_nortos_ticlang」プロジェクトをインポートして、以下を実行します。

1. Beyond SDK から「spi_peripheral_echo_interrupts.c」を新しいプロジェクトにコピーします。
2. 新規プロジェクトで、ADC12、SPI、TIMER、EVENT の sysconfig 設定を、Beyond SDK にならってコピーします。

CCS に関する追加サポートが必要な場合は、『Code Composer Studio ユーザー ガイド』を参照してください。

4.2 SK-AM62x/AM62L 評価基板でプロジェクトを実行

このセクションでは、コントローラとして使用されるコアに応じた実行手順について説明します。AM62L 評価基板では、同じ A53 コア手順に従うことができます。

4.2.1 A53 コア

『AM62x スターター キット評価基板クイック スタート ガイド』および『AM62L クイック スタート ガイド』を通じて取得したシリアル モニター上で、実行可能ファイルの格納場所に移動し、以下のコマンドを使用して実行します。

```
root@am62xx-evm:~#./<executable_name> -D <spidriver_name_from_/dev_folder> -s <speed> -v
```

例:

```
root@am62xx-evm:~#./spidev_adc_multibyte_multichannel -D /dev/spidev1.0 -s 16000000 -v
```

4.2.2 M4F コア

オンチップ フラッシュに事前に組み込まれている SK-AM62x バイナリを書き込みます。

1. CCS プロジェクトを MSPM0 ワークスペースとは別のワークスペースにインポートします。
   a. *<Beyond-SDK-installation-path>/Beyond-SDK/collaterals/appnotes/slaaej0-MSPM0-ADC-Attach/am62x/<x_Byte_x_Channel_SPI>/Controller/AM62x-M4F_Core_MCU_Domain/Debug/<file-name>.out*
   b. これにより、デバイスごとに 2 種類のデバッグ セッションが可能となります。
2. インポートした CCS プロジェクトをビルドします。
3. 「Target Configurations」(ターゲット構成) ウィンドウで、「AM62x_XDS110.ccxml」を右クリックします。
4. 「Launch Selected Configuration」(選択した構成を起動) を選択します。
5. 「Debug」(デバッグ) ウィンドウで「Texas Instruments XDS110 USB Debug Probe_0/BLAZAR_Cortex-M4F_1」(テキサス インスツルメンツ XDS110 USB デバッグ プローブ_0/BLAZAR_Cortex_M4F_1) をクリックします
6. 「Run」(実行) ->「Connect Target」(ターゲットの接続) の順に選択します
7. 「Run」(実行) ->「Reset」(リセット) ->「Subsystem Reset」(サブシステム リセット) の順に選択します。
8. 「Run」(実行) ->「Load」(ロード) ->「Load Program」(プログラムのロード) の順に選択します。
9. AM62x プロジェクトに対応する事前ビルド済みのバイナリを参照し、「OK」をクリックします。
10. これにより、バイナリでフラッシュに書き込まれます。
11. 「Run」(実行) ->「Resume」(再開) の順に選択します。

5 結果

使用されるアプリケーション コード例は以下のとおりです。

- シングル バイトは、8 ビット ADC データ転送を指します。
- マルチ バイトとは、設定された数のバイトを送信できる SPI 転送を指します。例では、2 バイトのデータが構成されています。MSPM0 の ADC の最大分解能は 12 ビットであるため、転送される 16 ビットのうち、ADC データが含まれるのは下位 12 ビットのみです。
- シングル チャネルとは、ADC で監視されるアナログ信号が 1 個のみであることを意味します。コントローラはトランザクションを開始するためにダミー コマンドを送信する必要がありますが、ペリフェラルでコマンド値をチェックする必要はありません。
- マルチチャネルとは、ADC が複数のアナログ信号をシーケンシャルに変換することを意味します。トランザクションを開始し、対応するチャネル データを受信するために、コントローラは有効なコマンドを送信する必要があります。
- AM62x コントローラと AM62L コントローラは、同様の出力を供給します。

5.1 シングル バイト シングル チャネル

以下の結果を得るために、8 ビット ADC に対して以下のアナログ入力を考慮したことに注意してください。

コマンド 0x00：ADC チャネル 7：正弦波信号 (3.3Vpp、1.65V DC オフセット、@2Hz)

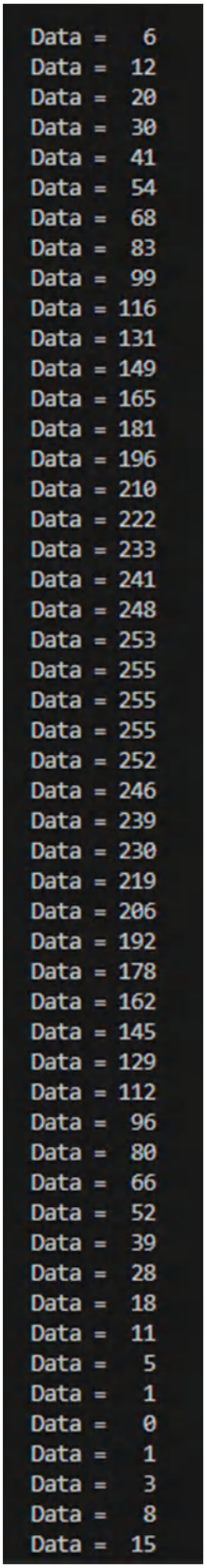


図 5-1 シングル バイト シングル チャネル正弦波の結果

コマンド 0x00：ADC チャネル 7：方形波信号 (3.3Vpp、1.65V DC オフセット、@2Hz、50% デューティ)

```
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
```

図 5-2. シングル バイト シングル チャネル方形波の結果

5.2 シングル バイト マルチチャネル

以下の結果を得るために、8 ビット ADC に対して以下のアナログ入力を考慮したことに注意してください。

コマンド 0x00：ADC チャネル 7：正弦波信号 (3.3Vpp、1.65V DC オフセット、@2Hz)

コマンド 0x01：ADC チャネル 8：DC 信号 (3.3V)

```
CommandID =  0, Data = 102      CommandID =  1, Data = 255
CommandID =  0, Data =  72      CommandID =  1, Data = 255
CommandID =  0, Data =  44      CommandID =  1, Data = 255
CommandID =  0, Data =  23      CommandID =  1, Data = 255
CommandID =  0, Data =   8      CommandID =  1, Data = 255
CommandID =  0, Data =   0      CommandID =  1, Data = 255
CommandID =  0, Data =   2      CommandID =  1, Data = 255
CommandID =  0, Data =  11      CommandID =  1, Data = 255
CommandID =  0, Data =  29      CommandID =  1, Data = 255
CommandID =  0, Data =  52      CommandID =  1, Data = 255
CommandID =  0, Data =  81      CommandID =  1, Data = 255
CommandID =  0, Data = 112      CommandID =  1, Data = 255
CommandID =  0, Data = 146      CommandID =  1, Data = 255
CommandID =  0, Data = 177      CommandID =  1, Data = 255
CommandID =  0, Data = 206      CommandID =  1, Data = 255
CommandID =  0, Data = 229      CommandID =  1, Data = 255
CommandID =  0, Data = 246      CommandID =  1, Data = 255
CommandID =  0, Data = 255      CommandID =  1, Data = 255
CommandID =  0, Data = 255      CommandID =  1, Data = 255
CommandID =  0, Data = 248      CommandID =  1, Data = 255
CommandID =  0, Data = 233      CommandID =  1, Data = 255
CommandID =  0, Data = 210      CommandID =  1, Data = 255
CommandID =  0, Data = 183      CommandID =  1, Data = 255
CommandID =  0, Data = 151      CommandID =  1, Data = 255
CommandID =  0, Data = 119      CommandID =  1, Data = 255
CommandID =  0, Data =  87      CommandID =  1, Data = 255
CommandID =  0, Data =  57      CommandID =  1, Data = 255
CommandID =  0, Data =  33      CommandID =  1, Data = 255
CommandID =  0, Data =  14      CommandID =  1, Data = 255
CommandID =  0, Data =   3      CommandID =  1, Data = 255
CommandID =  0, Data =   0      CommandID =  1, Data = 255
CommandID =  0, Data =   6      CommandID =  1, Data = 255
CommandID =  0, Data =  19      CommandID =  1, Data = 255
CommandID =  0, Data =  40      CommandID =  1, Data = 255
CommandID =  0, Data =  66      CommandID =  1, Data = 255
CommandID =  0, Data =  97      CommandID =  1, Data = 255
CommandID =  0, Data = 129      CommandID =  1, Data = 255
CommandID =  0, Data = 162      CommandID =  1, Data = 255
CommandID =  0, Data = 192      CommandID =  1, Data = 255
CommandID =  0, Data = 218      CommandID =  1, Data = 255
CommandID =  0, Data = 238      CommandID =  1, Data = 255
CommandID =  0, Data = 251      CommandID =  1, Data = 255
CommandID =  0, Data = 255      CommandID =  1, Data = 255
CommandID =  0, Data = 253      CommandID =  1, Data = 255
CommandID =  0, Data = 241      CommandID =  1, Data = 255
CommandID =  0, Data = 222      CommandID =  1, Data = 255
CommandID =  0, Data = 197      CommandID =  1, Data = 255
```

図 5-3. シングル バイト マルチチャネルの結果

5.3 マルチ バイト シングルチャネル

以下の結果を得るために、12 ビット ADC に対して以下のアナログ入力を考慮したことに注意してください。

コマンド 0x00：ADC チャネル 7：正弦波信号 (3.3Vpp、1.65V DC オフセット、@2Hz)

```
Data = 3879
Data = 4061
Data = 4095
Data = 4021
Data = 3810
Data = 3487
Data = 3070
Data = 2587
Data = 2062
Data = 1540
Data = 1065
Data =  641
Data =  307
Data =   97
Data =    6
Data =   52
Data =  232
Data =  527
Data =  915
Data = 1381
Data = 1896
Data = 2422
Data = 2915
Data = 3356
Data = 3719
Data = 3970
Data = 4095
Data = 4086
Data = 3953
Data = 3694
Data = 3326
Data = 2879
Data = 2377
Data = 1857
Data = 1338
Data =  884
Data =  497
Data =  209
Data =   43
Data =   13
Data =  111
Data =  346
Data =  683
Data = 1113
Data = 1603
Data = 2120
Data = 2639
Data = 3116
Data = 3527
```

図 5-4. マルチ バイト シングルチャネル正弦波の結果

コマンド 0x00：ADC チャネル 7：方形波信号 (3.3Vpp、1.65V DC オフセット、@2Hz、50% デューティ)

```
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data =    8
Data =    9
Data =    8
Data =    7
Data =    6
Data =    6
Data =    7
Data =    8
Data =    7
Data =    3
Data =    8
Data =    6
Data =    0
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data =   13
Data =    9
Data =   13
Data =    9
Data =   12
Data =    7
Data =    8
Data =    7
Data =    7
Data =    0
Data =    4
Data =   11
Data = 4095
```

図 5-5. マルチ バイト シングルチャネル方形波の結果

5.4 マルチ バイト マルチチャネル

以下の結果を得るために、12 ビット ADC に対して以下のアナログ入力を考慮したことに注意してください。

コマンド 0x00：ADC チャネル 7：正弦波信号 (3.3Vpp、1.65V DC オフセット、@2Hz)

コマンド 0x01：ADC チャネル 8：DC 信号 (3.3V)

```
CommandID =  0, Data =  2501        CommandID =  1, Data =  4094
CommandID =  0, Data =  3421        CommandID =  1, Data =  4095
CommandID =  0, Data =  3993        CommandID =  1, Data =  4092
CommandID =  0, Data =  4079        CommandID =  1, Data =  4088
CommandID =  0, Data =  3657        CommandID =  1, Data =  4095
CommandID =  0, Data =  2822        CommandID =  1, Data =  4091
CommandID =  0, Data =  1798        CommandID =  1, Data =  4095
CommandID =  0, Data =   840        CommandID =  1, Data =  4095
CommandID =  0, Data =   191        CommandID =  1, Data =  4095
CommandID =  0, Data =    16        CommandID =  1, Data =  4089
CommandID =  0, Data =   359        CommandID =  1, Data =  4088
CommandID =  0, Data =  1128        CommandID =  1, Data =  4095
CommandID =  0, Data =  2132        CommandID =  1, Data =  4095
CommandID =  0, Data =  3123        CommandID =  1, Data =  4090
CommandID =  0, Data =  3840        CommandID =  1, Data =  4086
CommandID =  0, Data =  4095        CommandID =  1, Data =  4093
CommandID =  0, Data =  3859        CommandID =  1, Data =  4086
CommandID =  0, Data =  3154        CommandID =  1, Data =  4086
CommandID =  0, Data =  2172        CommandID =  1, Data =  4095
CommandID =  0, Data =  1160        CommandID =  1, Data =  4088
CommandID =  0, Data =   370        CommandID =  1, Data =  4089
CommandID =  0, Data =    20        CommandID =  1, Data =  4090
CommandID =  0, Data =   172        CommandID =  1, Data =  4093
CommandID =  0, Data =   807        CommandID =  1, Data =  4092
CommandID =  0, Data =  1767        CommandID =  1, Data =  4090
CommandID =  0, Data =  2789        CommandID =  1, Data =  4094
CommandID =  0, Data =  3632        CommandID =  1, Data =  4089
CommandID =  0, Data =  4074        CommandID =  1, Data =  4095
CommandID =  0, Data =  4002        CommandID =  1, Data =  4092
CommandID =  0, Data =  3443        CommandID =  1, Data =  4090
CommandID =  0, Data =  2535        CommandID =  1, Data =  4084
CommandID =  0, Data =  1509        CommandID =  1, Data =  4090
CommandID =  0, Data =   618        CommandID =  1, Data =  4095
CommandID =  0, Data =    88        CommandID =  1, Data =  4089
CommandID =  0, Data =    63        CommandID =  1, Data =  4095
CommandID =  0, Data =   549        CommandID =  1, Data =  4087
CommandID =  0, Data =  1410        CommandID =  1, Data =  4095
CommandID =  0, Data =  2440        CommandID =  1, Data =  4089
CommandID =  0, Data =  3376        CommandID =  1, Data =  4087
CommandID =  0, Data =  3974        CommandID =  1, Data =  4092
CommandID =  0, Data =  4089        CommandID =  1, Data =  4092
```

図 5-6. マルチ バイト マルチチャネルの結果

6 まとめ

AM62x および AM62L は、幅広い組込みアプリケーションにとって理想的な選択肢となります。多くの組込みアプリケーションでは、センサから実際のアナログ信号を収集する必要があります。このドキュメントでは、MSM0 を使用して AM62x および AM62L プロセッサに 8 ビットと 12 ビットの ADC を統合する手順を説明します。この統合は、AM62x プロセッサにオンボード ADC 機能がないという重要な制限を解決します。それに対し、AM62L プロセッサは約 10 ENOB ADC を搭載しており、特定のアプリケーションでは 12 ビットの分解能を必要とする場合があります。

ソリューション アーキテクチャでは、AM62x/AM62L プロセッサ (A53 または M4F コアを使用) を SPI コントローラとして、MSPM0L130x マイコンをペリフェラル デバイスとして配置します。通信は、コントローラでは 50MHz までの速度で全二重 SPI を介して行われますが、MSPM0L ペリフェラルでは 16Mbit/s に制限されます。このシステムは、シングル バイトまたはマルチ バイト転送とシングル チャネルまたはマルチチャネル構成を組み合わせた複数の動作モードをサポートしており、複数のアナログ入力を同時に監視できます。ADC サンプリングはタイマ トリガで、連続的なデータ更新によりリアルタイム性能を維持します。

ハードウェアの設定には、拡張ヘッダを介して SPI 接続により LP-MSPM0L1306 LaunchPad に接続された SK-AM62x スターター キットまたは AM62L 評価基板に加え、テスト用のアナログ信号源が必要です。ソフトウェアの実装には、SPI ドライバをサポートするための Linux カーネルの変更、適切なピン多重化を行うためのデバイス ツリーの設定、MSPM0 ペリフェラルと Am62x M4F コア両方に向けた CCS プロジェクト、ならびに A53 コア上の Linux ユーザースペースで実行するためのコンパイル済み実行ファイルが含まれます。このソリューションは、SK-AM62x および AM62L の両コントローラーに対し、正弦波および方形波のテスト入力 (周波数 2Hz で 3.3Vpp) を用いて正常に検証されました。これにより、すべての構成モードにおいて正確な ADC データ取得が可能であることが実証されるとともに、サポートされる最大 SPI 速度においても信頼性の高い性能が確認されました。

7 参考資料

1. テキサス・インスツルメンツ、*AM625*、製品ページ
2. テキサス インスツルメンツ、『*AM62L*』
3. テキサス・インスツルメンツ、*MSPM0L1306*、製品ページ
4. テキサス インスツルメンツ、『AM625 Sitara プロセッサ』、データシート。
5. テキサス インスツルメンツ、『*AM62Lx Sitara プロセッサ*』データシート。
6. テキサス インスツルメンツ、『MSPM0L130x ミックスド シグナル マイコン』、データシート。
7. テキサス インスツルメンツ、『SK-AM62 ユーザー ガイド』ユーザー ガイド。
8. テキサス インスツルメンツ、『AM62L ユーザー ガイド』、ユーザー ガイド。
9. テキサス インスツルメンツ、『LP-MSPM0L1306 ユーザー ガイド』、ユーザー ガイド。
10. テキサス インスツルメンツ、『SK-AM62x クイック スタート ガイド』、クイック スタート ガイド。
11. テキサス インスツルメンツ、『AM62L クイック スタート ガイド』、クイック スタート ガイド。
12. テキサス インスツルメンツ、『MSPM0 クイック スタート ガイド』、クイック スタート ガイド。
13. テキサス インスツルメンツ、『カーネル：基本的なコンポーネント：プロセッサ SDK Linux AM62x』、ユーザー ガイド。
14. テキサス インスツルメンツ、『カーネルの基本コンポーネント：プロセッサ SDK Linux AM62L』、ユーザー ガイド。
15. テキサス インスツルメンツ、『AM62x 用プロセッサ SDK Linux』、Web ページ。
16. テキサス インスツルメンツ、『AM62L 用 Linux SDK』、Web ページ。
17. テキサス インスツルメンツ、『「Hello World」プログラムのコンパイル』 Web ページ。
18. テキサス インスツルメンツ、『SK-AM62x 概要導入ガイド』、ユーザー ガイド。
19. テキサス インスツルメンツ、『AM62L 概要導入ガイド』、ユーザー ガイド。
20. テキサス インスツルメンツ、『MSPM0L1306 LaunchPad 開発キット』、ユーザー ガイド

8 改訂履歴

Changes from Revision * (November 2023) to Revision A (April 2026) **Page**

重要なお知らせと免責事項

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

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

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

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

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

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

重要なお知らせと免責事項

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

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

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

TI の製品は、 TI の販売条件 、 TI の総合的な品質ガイドライン 、 ti.com または TI 製品などに関連して提供される他の適用条件に従い提供されます。TI がこれらのリソースを提供することは、適用される TI の保証または他の保証の放棄の拡大や変更を意味するものではありません。 TI がカスタム、またはカスタマー仕様として明示的に指定していない限り、TI の製品は標準的なカタログに掲載される汎用機器です。

お客様がいかなる追加条項または代替条項を提案する場合も、TI はそれらに異議を唱え、拒否します。

最終更新日：2025 年 10 月