参考資料

Application Note

F280039C 用 A-Format 絶対エンコーダ プライマリ インターフェイス

Hang Yang *Industrial ASM*

概要

このアプリケーション ノートでは、リアルタイム C2000 MCU の構成可能ロジック ブロック (CLB) を活用して、A-Format エンコーダ インターフェイスを実装するためのガイダンスを提供します。提案された実装では、CLB モジュールを使用して SPI 送受信シーケンスを管理し、SPI クロック信号を生成し、即座に受信 (RX) 側の巡回冗長検査 (CRC) を計算します。SPI は、データの転送と受信に使用されます。

このアプリケーション ノートには、SPI と CLB の実装の詳細、および検証テストの結果が記載されています。

目次

商標

1 はじめに

モーター制御アプリケーションでは、高精度のモーター位置検出が必要な場合、一般的に絶対エンコーダが使用されます。A-Format は、Nikon 絶対エンコーダのインターフェイス プロトコルです。

TI の C2000 マイコンの重点分野の 1 つに、モーター制御があります。C2000 には独自の構成可能ロジック ブロック (CLB) モジュールが搭載されており、モーター制御アプリケーションのエンコーダ インターフェイスをサポートします。CLB にはさまざまなロジック ブロックが含まれており、CLB ツール チェーンを使用してカスタマイズできます。TI では、CLB を活用してエンコーダ インターフェイスを実装したいくつかのリファレンス デザインを提供しており、お客様のアプリケーション開発を容易にしています。その 1 つは、TIDM-1011 Tamagawa T-Format インターフェイスのリファレンス デザインです (関連資料 [1] を参照)。T-Format インターフェイスは、A-Format インターフェイスと類似しています。A-Format インターフェイスは、ハードウェアを変更せずに T-Format インターフェイスのソフトウェア構成 (SPI パラメータ、CLB ロジック、フレーム フォーマットを含む) を変更することで実装できます。

このアプリケーション ノートの目的は、T-Format インターフェイスからの変更に基づき、A-Format インターフェイスの実装方法に関するガイダンスを提供することです。

2 システムの説明

A-Format インターフェイスの実装は、TIDM-1011 設計に基づいています。実装は、同じハードウェア プラットフォームを使用して検証されました。LAUNCHXL-F280039C (C2000 LaunchPad)、BOOSTXL-POSMGR (Position Manager BoosterPack)、Nikon A-Format エンコーダ (モデル：Nikon-MX50AHN00、TIDM-1011 の Tamagawa エンコーダの置き換え)。T-Format インターフェイスを A-Format に適応させる場合、以下のようなソフトウェアの変更のみが必要です。1) SPI ペリフェラル構成 (FIFO の幅 / 深度、クロック極性 / 位相)、2) CLB ロジックのカスタマイズ (CRC 計算、送受信シーケンス制御)、3) A-Format フレームのフォーマット (18 ビット フィールド分割、コマンド / 応答解析)。

A-Format インターフェイスは、データの転送と受信に SPI を使用します。CLB は、SPI の転送 / 受信シーケンスを管理するとともに、RX CRC を即座に計算するために使用されます。次の図に、インターフェイスの簡略化したブロック図を示します。

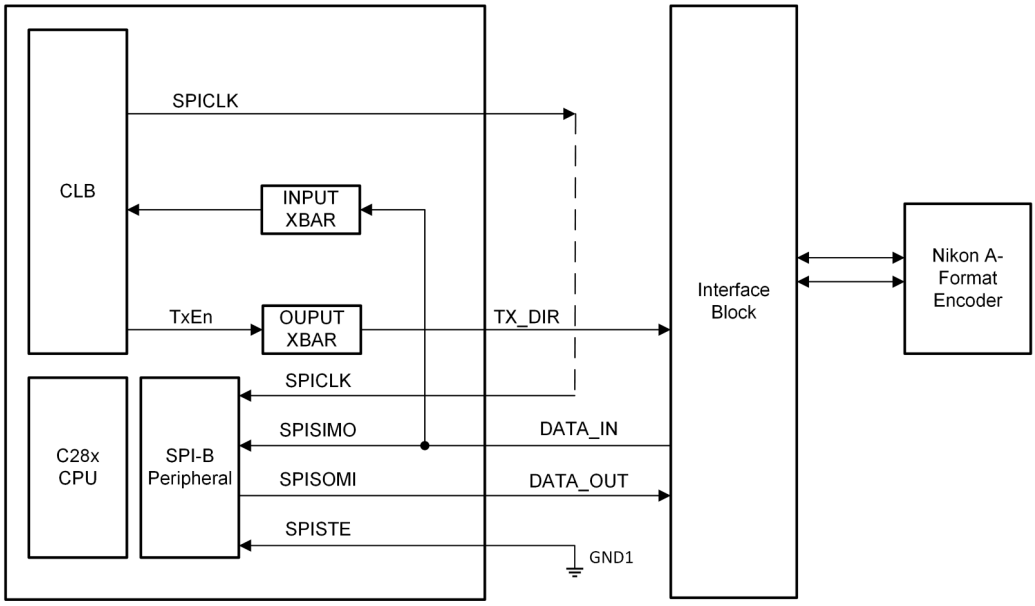

図 2-1. システム図

2.1 SPI の実装

A-Format フレームは、SPI ペリフェラル経由で転送されます。各通信サイクルについて、最初にコマンド フィールドが TX FIFO に入力され、その後エンコーダの応答が RX FIFO で受信されます。

A-Format のフィールドは 18 ビットです。SPI FIFO に 18 ビットのフィールドを入力するために、各フィールドは 2 つの 9 ビット ハーフ フィールドに分割されます。したがって、FIFO の幅は 9 ビットに設定されます。以下の図に、18 ビットの A-Format フィールドを 9 ビットの SPI FIFO エントリに分割する方法を示します。CDF0 コマンドを例として考えると、コントローラは 1 つの 18 ビット フィールドをエンコーダに送信し、エンコーダは 3 つの 18 ビット フィールドで応答します。したがって、TX FIFO の深度は 2 に設定され、RX FIFO の深度は 6 エントリに設定されます

転送および受信するフィールドの数は、コマンドによって異なる場合があります。ユーザーは、それに応じて TX と RX FIFO の深度を変更する必要があります。さまざまなコマンドに基づいて FIFO の深度を調整する場合、同期を確保するため、SPI ペリフェラル構成 (FIFO 深度レジスタ) と CLB ロジック (送受信シーケンス制御) を一貫して変更する必要があります。

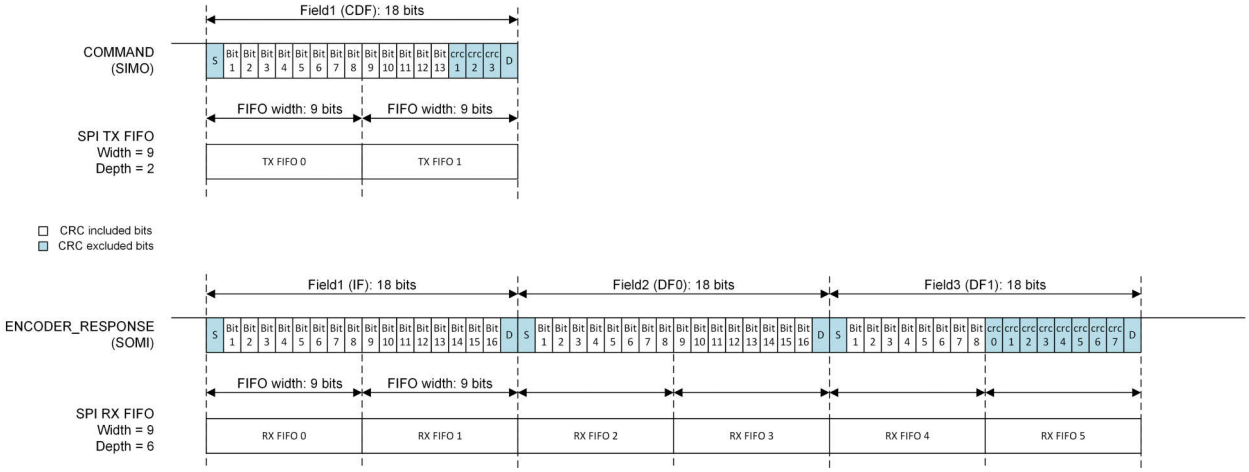

図 2-2. A-Format フィールドを SPI FIFO に分割する

2.2 CLB の実装

このソリューションには 2 つの CLB タイルがあり、1 つは SPI タイミングを制御し、もう 1 つは RX CRC を即座に計算します。SPI タイミング CLB タイルは、T-Format ソリューションのタイルと互換性があるため、変更は不要です。ただし、RX CRC CLB タイルは、フィールド フォーマットと CRC 多項式が異なるため、変更が必要です。CRC 計算用の CLB 実装を以下に示します。残りの CLB 機能については、T-Format 設計ガイドを参照することができます。

RX CRC は、リニア フィードバック シフト レジスタ (LFSR) モードのカウンタ ブロックを使用して計算されます。RX ビットは SPI クロックごとに LFSR でシフトインされ、最後のビットがシフトインされた時点で LFSR から CRC の結果を読み取ることができます。図 2-3 では、LFSR にシフトインするビットがマークされています。CLB ロジックはビット マスク フィルタを実装しており、CRC 計算のために LFSR にシフトインする必要のある特定のビットを選択できます。マスクは以下の 2 つのルールに基づいて生成されます。

- 18 ビットごとに中央の 16 ビット
- 先頭の n ビット

ここで、n は CRC フィールドと最後の区切り記号を除く合計フレーム長です。CRC ビットをマスクするために使用される信号は、CRC_MASK および DATA_VALID という名前が付けられています。

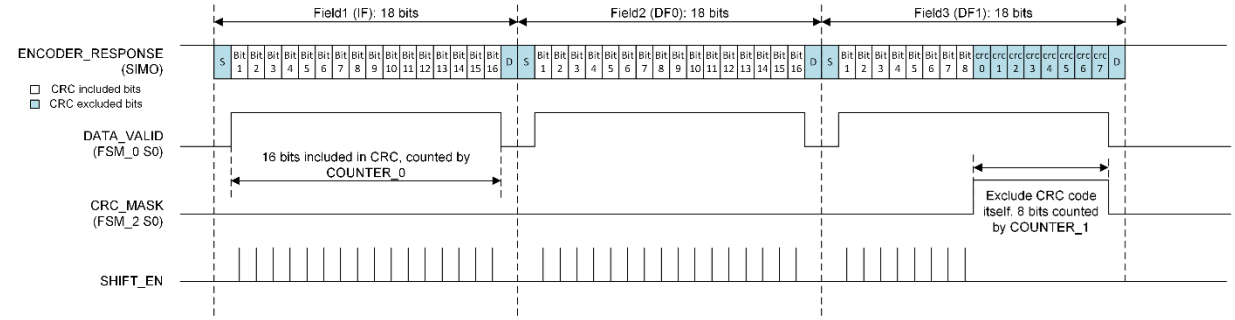

図 2-3. CLB での CRC 計算用マスクの生成

マスクのルールは T-Format と異なるため、SHIFT_EN 信号生成用の CLB ロジックも変更されます。2 つのカウンタと 1 つの FSM を使用して、SHIFT_EN 信号を生成します。図 2-4 に、CRC 計算用の CLB タイルのブロック図を示します。ブロック図は、COUNTER0、COUNTER1、FSM2、LUT1 を含む SHIFT_EN 信号生成ロジックを除いて、T-Format の実装とほとんど同じです。

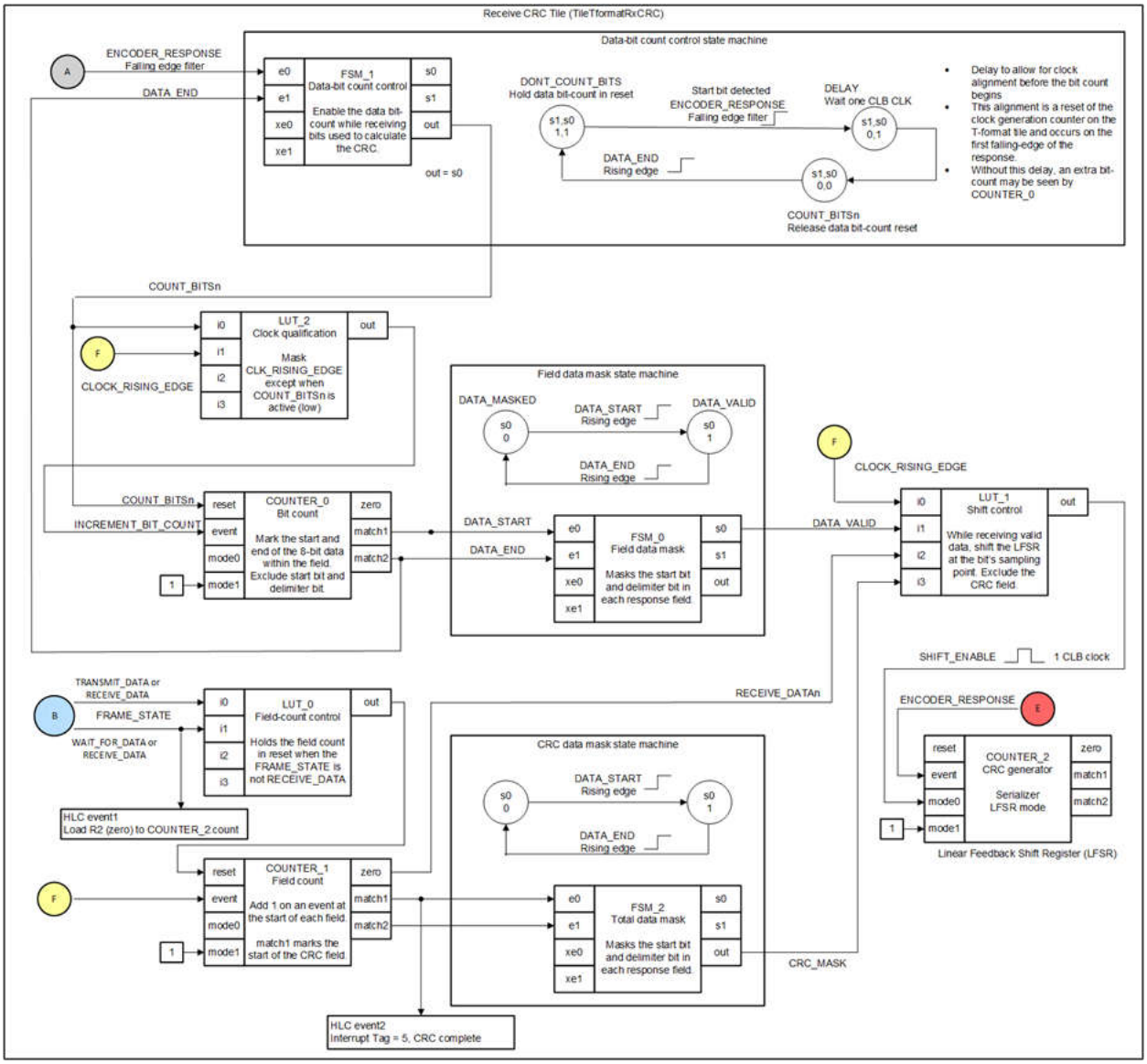

図 2-4. CRC 計算用の CLB 図

上記のロジックは、TI の CLB ツール チェーンを使用して実装されています (Rev. B、関連資料 [2] を参照)。ツール チェーンの使用方法については、「CLB ツール ユーザー ガイド (Rev. B)」を参照してください。ロジックを検証する簡単な方法は、ツール チェーンの一部であるシミュレーションを使用することです。CRC 計算のシミュレーションの例を、図 2-5 に示します。エンコーダからの応答は、波形の「境界入力 4」として入力されます。応答データが受信されると、SHIFT_EN 信号は「Counter2_mode0」として生成されます。この信号は、CRC 計算に含まれるビットを示します。応答データは LFSR にシフトインされ、SHIFT_EN 信号の各パルスに応じて「counter_2」とマークされます。最後の SHIFT_EN パルスが入力されると、LFSR は 0x1576608F の値を保持します。ここで、最後の 8 ビットである 0x8F は、必要な CRC 結果を表します。

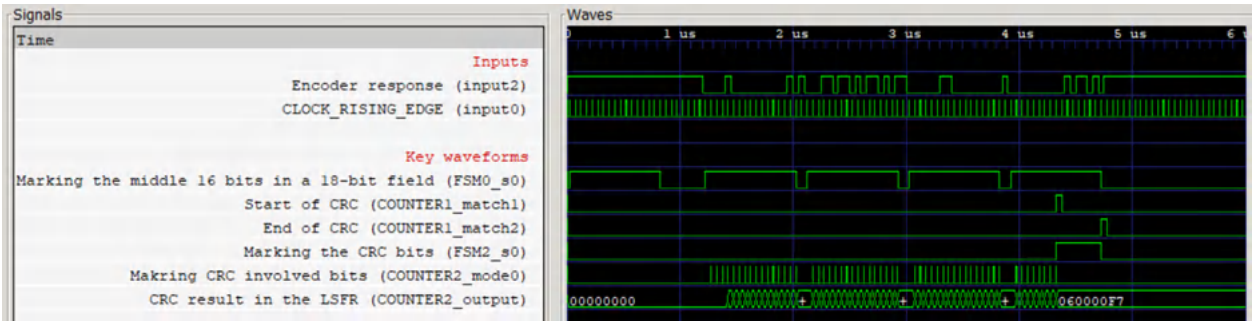

図 2-5. CLB の CRC 計算ロジックのシミュレーション

2.3 ソフトウェアの変更点

T-Format ソリューションのソフトウェアのほとんどは、A-Format インターフェイスで再利用できます。上記の SPI および CLB 実装に基づき、ソフトウェアに変更点があります。

1 つ目は tformat_configureCLBLen() API です。この関数は、コマンドによって異なる場合がある適切な転送クロック数および受信クロック数で CLB カウンタを構成するために使用されます。A-Format CLB の実装 (CRC 計算用) では、CRC マスクはエンコーダ応答のクロック数に依存するため、この API には 2 つの追加構成が追加されます。構成された 2 つのライン、COUNTER_1_MATCH_1 と COUNTER_1_MATCH_2 は、FSM_2 を正しくトリガし、エンコーダ応答の CRC をマークします。この構成の効果については、図 2-5 を参照してください。

```
static inline void
tformat_configureCLBLen(uint16_t transmitClocks, uint16_t receiveClocks)
{
    CLB_writeInterface(PM_TFORMAT_CLB_BASE,
                       CLB_ADDR_COUNTER_1_MATCH1, transmitClocks);
    CLB_writeInterface(PM_TFORMAT_CLB_BASE,
                       CLB_ADDR_COUNTER_1_MATCH2, transmitClocks);
    CLB_writeInterface(PM_TFORMAT_CLB_BASE,
                       CLB_ADDR_HLC_R0, receiveClocks);


    // Below are the changes
    CLB_writeInterface(PM_TFORMAT_RX_CRC_BASE,
                       CLB_ADDR_COUNTER_1_MATCH1, receiveClocks - 9);
    CLB_writeInterface(PM_TFORMAT_RX_CRC_BASE,
                       CLB_ADDR_COUNTER_1_MATCH2, receiveClocks - 1);

    return;
}
```

2 つ目は、PM_tformat_setupCommand() 関数に渡されるパラメータでの変更点です。

```
PM_tformat_setupCommandReadoutOrReset(uint16_t commandID0_1_2_3_7_8_C,
                                      uint16_t tformatRXClocks,
                                      uint16_t tformatRXFields,
                                      uint16_t tformatTXClocks,
                                      uint16_t tformatFIFOLevel)
```

これらのパラメータには、フレーム データ、転送および受信データのフィールド / クロック数、および FIFO LEVEL が含まれます。これらのパラメータは、インターフェイス プロトコル自体に関連しており、SPI 実装セクションで説明した FIFO 分割の影響を受けます。以下に、これらのパラメータを正しく計算する方法の例を示します。この例は、A-Format CDF0 に基づいています。

```
    #define AFORMAT_FRAME_LEN 9
    #define AFORMAT_TX_FRAMES 1
    #define AFORMAT_CFID0RES_RX_FIELDS 4

    uint16_t aformatTXClocks = (AFORMAT_TX_FRAMES*AFORMAT_FRAME_LEN);
    uint16_t aformatRXClocks = (2u*AFORMAT_CFID0RES_FIELDS*AFORMAT_FRAME_LEN);
    uint16_t aformatRXFields = (2u*AFORMAT_CFID0RES_FIELDS);
    uint16_t aformatFIFOLevel = (2u+2u*AFORMAT_CFID0RES_FIELDS);
```

A-Format の 18 ビット フィールドは 2 つの 9 ビット ハーフ フィールドに分割されるため、FIFO に書き込む際は、上位のハーフ フィールドが先に入力され、次に下位のハーフ フィールドが入力されます。T-Format ソリューションと同様に、残りの TX FIFO には 0xFFFF が書き込まれ、受信中に TX ラインが High に保持されます。以下は FIFO 書き込みの例です。

```
    HWREGH(PM_TFORMAT_SPI + SPI_O_TXBUF) = cdf_msh;  // Change
    HWREGH(PM_TFORMAT_SPI + SPI_O_TXBUF) = cdf_lsh;  // Change

    HWREGH(PM_TFORMAT_SPI + SPI_O_TXBUF) = 0xFFFF;
    HWREGH(PM_TFORMAT_SPI + SPI_O_TXBUF) = 0xFFFF;
    HWREGH(PM_TFORMAT_SPI + SPI_O_TXBUF) = 0xFFFF;
    HWREGH(PM_TFORMAT_SPI + SPI_O_TXBUF) = 0xFFFF;

    HWREGH(PM_TFORMAT_SPI + SPI_O_TXBUF) = 0xFFFF;
    HWREGH(PM_TFORMAT_SPI + SPI_O_TXBUF) = 0xFFFF;
    HWREGH(PM_TFORMAT_SPI + SPI_O_TXBUF) = 0xFFFF;
    HWREGH(PM_TFORMAT_SPI + SPI_O_TXBUF) = 0xFFFF;
    HWREGH(PM_TFORMAT_SPI + SPI_O_TXBUF) = 0xFFFF;
    HWREGH(PM_TFORMAT_SPI + SPI_O_TXBUF) = 0xFFFF;
```

最後に、デコード関数 PM_tformat_receiveDataID0_1_7_8_C() を変更する必要があります。この関数は SPI バッファを読み取り、インターフェイス プロトコルに従って未加工データを包括的なデータ構造にデコードします。A-Format における CDF0 のデコード応答の例を以下に示します。RX FIFO も分割されることに注意してください。

```
void aformat_decode(){
    // for now, limited to the response of CF0
    uint16_t fullData[4] = {0};

    // full data: get rid of start bit 0 and end bit 1
    fullData[0] = (tformatRxData[3]<<7)|(tformatRxData[4]>>2);
    fullData[1] = (tformatRxData[5]<<7)|(tformatRxData[6]>>2);
    fullData[2] = (tformatRxData[7]<<7)|(tformatRxData[8]>>2);
    fullData[3] = (tformatRxData[9]<<7)|(tformatRxData[10]>>2);
    aformatData.fullData = ((uint64_t)fullData[0]<<48)|
                           ((uint64_t)fullData[1]<<32)|
                           ((uint64_t)fullData[2]<<16)|
                           ((uint64_t)fullData[3]);

    // crc data
    aformatData.crcIncludedData = aformatData.fullData>>8; //for cmd ID 0 only!!

    // crc recived from encoder
    aformatData.crcRecv = (aformatData.fullData & 0x00000000000000ff);

    // crc by CLB
    aformatData.crcByCLB = aformat_getRxCRCbyCLB();
    aformatData.crcBothRecvAndCLB = ((aformatData.crcRecv&0xff)<<8) | (aformatData.crcRecv&0xff);

    // single turn data fullData[16:34] = ST[0:18] (19bit)
    aformatData.stData = (aformatData.fullData & 0x0000ffffd0000000)>>29;
    aformatData.stData = (__flip32(aformatData.stData)>>13)&0x7FFFF;

    // multi turn data fullData[35:51] (17bit)
    aformatData.mtData = (aformatData.fullData & 0x000000001ffff000)>>12;
    aformatData.mtData = (__flip32(aformatData.mtData)>>15)&0x1FFFF;

    // err code
    aformatData.errCode = (aformatData.fullData & 0x000f000000000000)>>48;

    // command ID
    aformatData.cmdID =  (aformatData.fullData & 0x03e0000000000000)>>53;

    // encoder address
    aformatData.encoderAddress = (aformatData.fullData & 0x1c00000000000000)>>58;

    encoderStatus.position = (float)aformatData.stData/524288.0*360;
    encoderStatus.turns = aformatData.mtData;
}
```

3 検証

このインターフェイス ソリューションは、Nikon A-Format エンコーダでテストされています。テスト設定には、以下のハードウェアが含まれます。

- LAUNCHXL-F280039C
- BOOSTXL-POSMGR
- フォーマット エンコーダー Nikon-MX50AHN00

図 3-1 に、テスト設定の実機写真を示します。

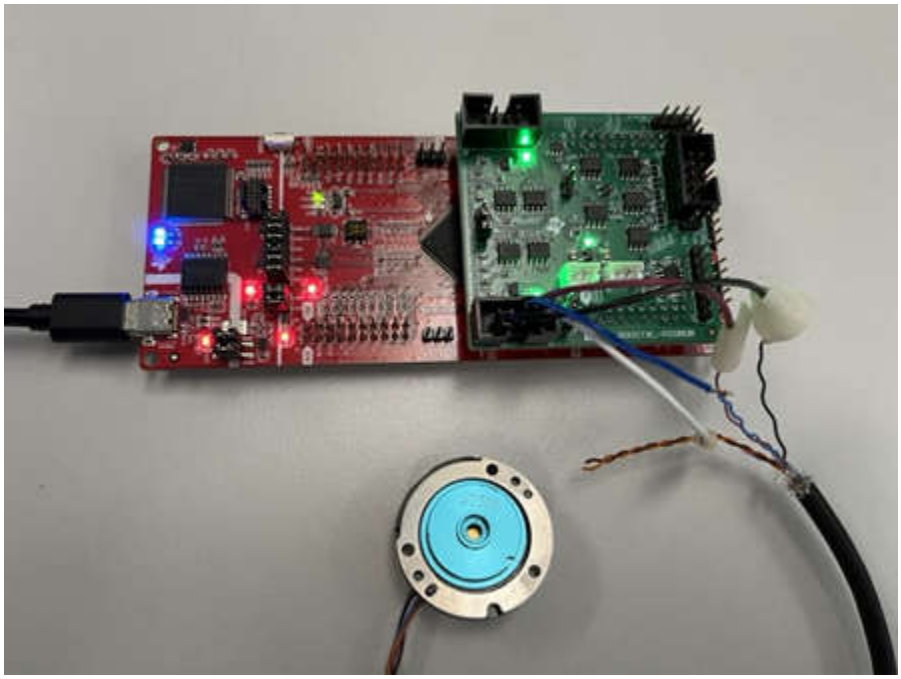

図 3-1. 検証の設定

このテスト設定の目的は、C2000 コントローラが A-Format フレームを正しく送受信できることを検証することです。このテストの目標は、CRC チェックを通過できるエンコーダ応答を取得することです。エンコーダの角度精度 (機械的性能) はこの検証の範囲を超えているため、エンコーダは実際のモーターには取り付けられていないことに注意してください。

目的のテスト目標を達成するため、コマンド フレーム CDF0 がテスト用に選択されています。このコマンド フレームは、エンコーダに対して、40 ビットの位置データ全体を返すよう要求します。テスト プログラムは、まず CDF0 コマンドをエンコードして SPI バッファに書き込み、CLB 開始信号を発行してコマンドを送信し、応答の CRC を検証します。結果は、CCS 式ウィンドウを使用して変数として表示されます。

まず、図 3-2 に示すように、SPI 受信バッファ データ (未加工の 9 ビット エントリ) をオシロスコープの波形 (POCI ピンにキャプチャ) と比較します。SPI バッファは、上記のように、3 ワードのダミー データで始まり、各ワードは 9 ビットです。ダミー データの後にエンコーダの応答が続きます。このエンコーダの応答には 8 ワードが含まれ、各ワードは 9 ビットで、有効なデータは各 9 ビット エントリの最下位ビット (LSB) に格納されます。バッファは右側の波形と整合しています。この比較は、A-Format フレームを転送および受信するために SPI および CLB が正しく設定されていることを示しています。

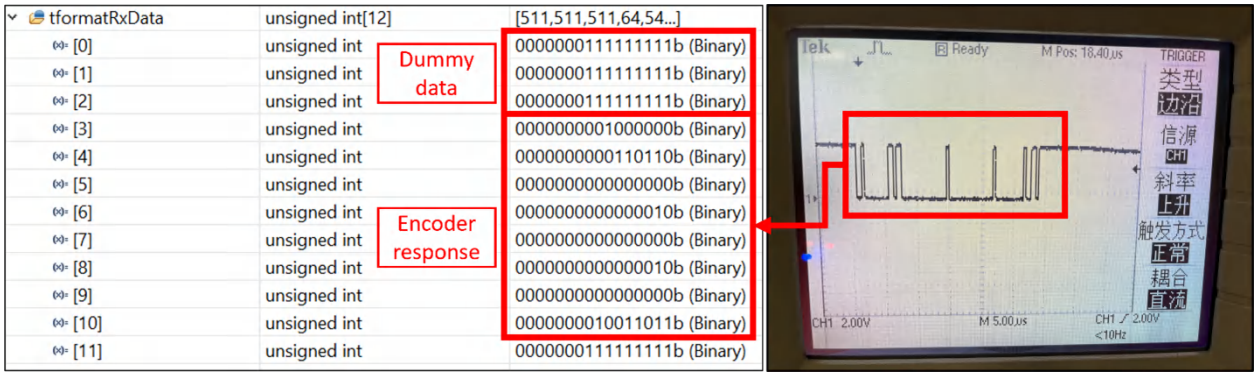

図 3-2. RX FIFO のデータおよび波形

次に、図 3-2 でデコードされたデータがさらにチェックされます。

デモのため、受信した CRC と計算された CRC を 1 つの 32 ビット変数に結合し、CCS の式ウィンドウにその変数を表示することで CRC を検証します。これにより、CCS が 1 つの通信サイクルで CRC 変数をサンプリングできるようになります。この結果として、CSS 式ウィンドウのスクリーン ショットを図 3-3 に示します。このウィンドウには、受信したフレームの未加工データ、デコードされたフレーム、および受信した CRC が計算されたものと一致していることを確認できます

一方、応答からデコードされた角度およびマルチターン データの動作は、エンコーダを手動で回転させ、デコードされた角度位置とマルチターン データの変化を観察することで検証されます (手動によるエンコーダの回転と一致)。

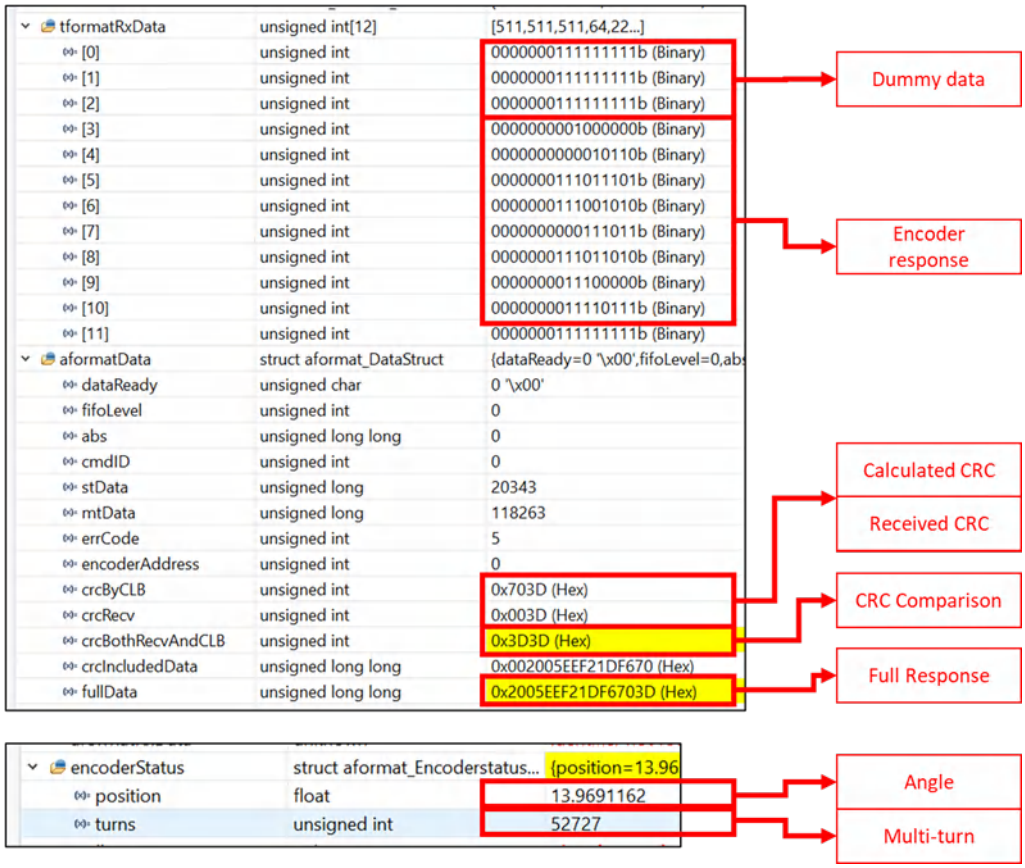

図 3-3. エンコーダ応答データのフル デコード

4 まとめ

このアプリケーションノートでは、C2000 F280039C マイコンの構成可能ロジックブロック (CLB) を活用して、Nikon A-Format 絶対エンコーダ インターフェイスを実装するためのガイダンスを提供します。主な実装は次のとおりです。1) 18 ビット フィールド分割用の SPI ペリフェラル構成 (9 ビット FIFO 幅)、2) SPI タイミング制御および A-Format 固有の CRC 計算用の CLB ロジックのカスタマイズ 3) コマンド / 応答処理用のソフトウェア フレーム解析。検証結果により、インターフェイスが A-Format フレームを正しく送受信し、CRC チェックを通過し、位置データを正確にデコードすることが確認されました。このソリューションは、TIDM-1011 T-Format のリファレンス デザインに基づくソフトウェア変更のみを必要とするため、開発者の開発労力を軽減できます。

5 参考資料

1. TIDM-1011 Tamagawa T-Format エンコーダ インターフェイスのリファレンス デザイン。テキサス インスツルメンツ、2022 年。
2. CLB ツール ユーザー ガイド (テキサス インスツルメンツ)
3. TMS320F280039C マイコン データシート (テキサス インスツルメンツ)

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