Application Note

F280039C 的 A-Format 绝对编码器主接口

Hang Yang *Industrial ASM*

摘要

本应用手册提供了利用实时 C2000 MCU 的可配置逻辑块 (CLB) 实现 A-Format 编码器接口的指导。在建议的实现方案中，CLB 模块用于管理 SPI 发送/接收序列，生成 SPI 时钟信号，并动态计算接收 (RX) 循环冗余校验 (CRC)。SPI 用于数据传输及接收。

本应用手册详细介绍了 SPI 及 CLB 的实现以及验证测试结果。

内容

商标

1 简介

在电机控制应用中，绝对编码器通常用于需要高精度电机位置感应的场景。A-Format 为 Nikon 绝对编码器的接口协议。

TI C2000 微控制器的重点之一是电机控制。C2000 具有独特的可配置逻辑块 (CLB) 模块，可支持电机控制应用的编码器接口。CLB 包括可以使用 CLB 工具链自定义的各种逻辑块。TI 提供了多种参考设计，这些参考设计利用 CLB 实现编码器接口，以简化客户应用开发。其中之一是 TIDM-1011 Tamagawa T-Format 接口参考设计（请参阅参考文献 [1]）。T-Format 接口类似于 A-Format 接口。A-Format 接口可以通过在不更改硬件的情况下修改 T-Format 接口的软件配置（包括 SPI 参数、CLB 逻辑和帧格式）来实现。

本应用手册旨在提供有关如何通过从 T-Format 接口执行修改来实现 A-Format 接口的指导。

2 系统说明

A-Format 接口实现依据 TIDM-1011 设计。此实现方案使用相同硬件平台进行了验证：LAUNCHXL-F280039C (C2000 LaunchPad)、BOOSTXL-POSMGR (Position Manager Booster Pack) 和 Nikon A-Format 编码器（型号：Nikon-MX50AHN00，替换 TIDM-1011 中的 Tamagawa 编码器）。调整 T-Format 接口为 A-Format 时，只需进行软件修改，包括：1) SPI 外设配置（FIFO 宽度/深度、时钟极性/相位）；2) CLB 逻辑自定义（CRC 计算、发送/接收序列控制）；3) T-Format 帧格式设定（18 位字段拆分、命令/响应解析）。

T-Format 接口使用 SPI 进行数据传输及接收。CLB 用于管理 SPI 的传输/接收序列并且动态计算 RX CRC。下图显示了该接口的简化方框图。

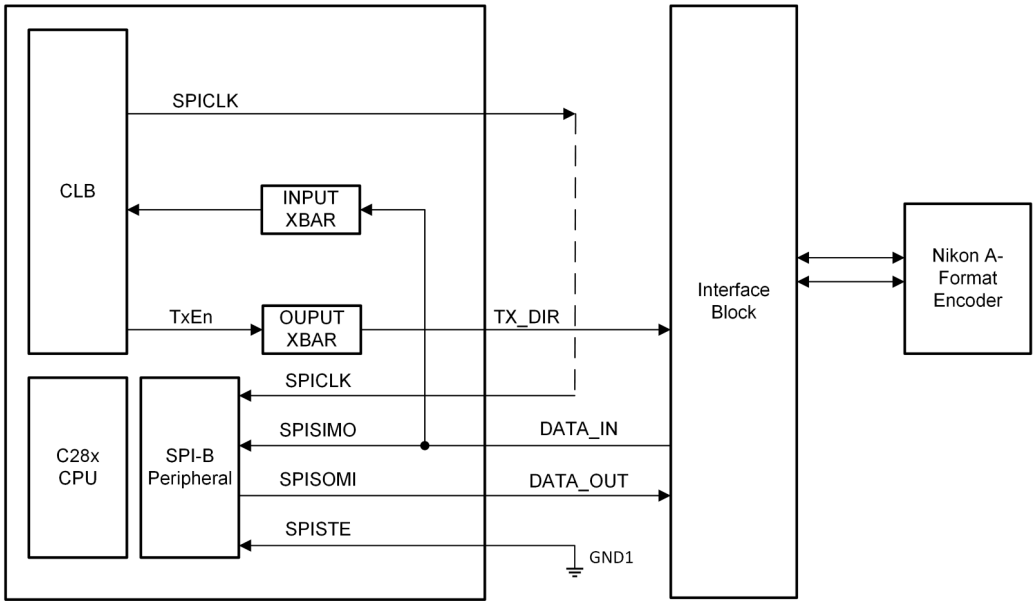

图 2-1. 系统图

2.1 SPI 实现方案

A-Format 帧通过 SPI 外设传输。对于每个通信周期，首先将命令字段填充到 TX FIFO 中，然后在 RX FIFO 中接收编码器响应。

A-Format 中的字段是 18 位。为了将 18 位字段填充到 SPI FIFO 中，每个字段被拆分为两个 9 位半字段。相应地，FIFO 的宽度设置为 9 位。下图示出了将 18 位 A-Format 字段拆分为 9 位 SPI FIFO 条目。以 CDF0 命令为例：控制器将一个 18 位字段发送到编码器，编码器以三个 18 位字段进行响应。因此，TX FIFO 的深度设置为 2，RX FIFO 深度的深度设置为 6 个条目

要传输与接收的字段数可能会因命令而异。用户应相应地更改 TX 及 RX FIFO 的深度。当根据不同的命令调整 FIFO 深度时，必须持续修改 SPI 外设配置（FIFO 深度寄存器）和 CLB 逻辑（发送/接收序列控制），以确保同步。

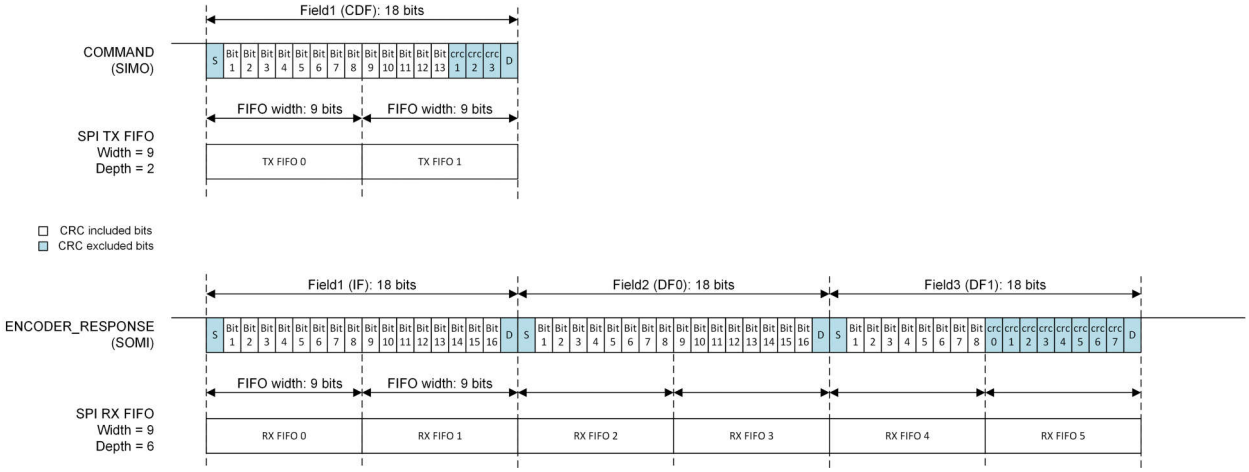

图 2-2. 将 A-Format 字段拆分为 SPI FIFO

2.2 CLB 实现方案

该解决方案中具有两个 CLB 逻辑块、一个控制 SPI 时序、另一个动态计算 RX CRC。SPI 时序 CLB 逻辑块将与 T-Format 解决方案中的逻辑块兼容，因此无需更改。但是，由于字段格式和 CRC 多项式不同，RX CRC CLB 逻辑块需要更改。下文介绍了用于 CRC 计算的 CLB 实现。对于其余的 CLB 函数，可以参考 T-Format 设计指南。

在线性反馈移位寄存器 (LFSR) 模式下用计数器块计算 RX CRC。RX 位在每个 SPI 时钟在 LFSR 中移位，一旦最后一位移入，就可以从 LFSR 读取 CRC 结果。图 2-3 标记应移入 LFSR 的位。CLB 逻辑实现了一个位掩码滤波器来选择需要移入 LFSR 进行 CRC 计算的特定位，根据以下两条规则生成掩码：

- 每 18 位的中间 16 位
- 前 n 位

其中，n 是总帧长度，不包括 CRC 字段和末尾的定界符。用于屏蔽 CRC 位的信号命名为 CRC_MASK 及 DATA_VALID。

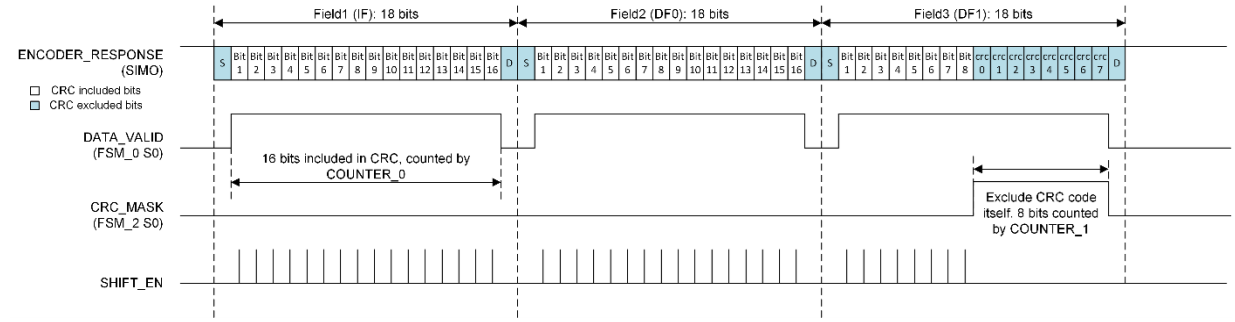

图 2-3. 在 CLB 内生成 CRC 计算掩码

由于屏蔽规则与 T-fomat 不同，因此用于 SHIFT_EN 信号生成的 CLB 逻辑也发生了变化。使用两个计数器及一个 FSM 来生成 SHIFT_EN 信号。图 2-4 展示了用于 CRC 计算的 CLB 逻辑块方框图。除了 SHIFT_EN 信号生成逻辑（包括 COUNTER0、COUNTER1 和 FSM2 和 LUT1）之外，方框图与 T-Format 实现基本相同。

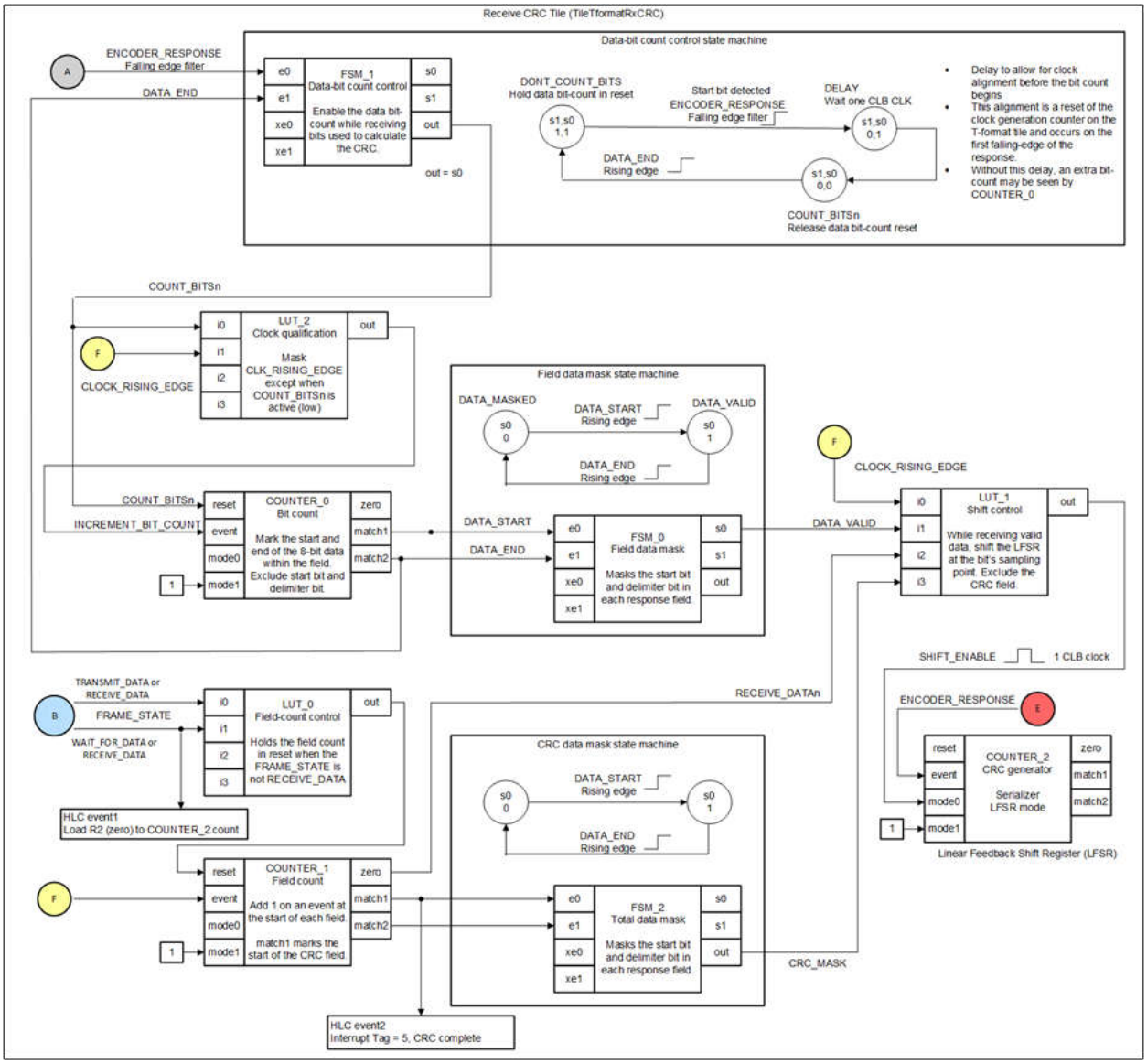

图 2-4. CRC 计算的 CLB 图

上述逻辑通过 TI CLB 工具链实现（修订版 B，请参阅参考文献[2]）。有关工具链的使用，请参阅 CLB 工具用户指南（修订版 B）。验证逻辑的一种快速方法是在工具链中进行仿真。图 2-5 展示了 CRC 计算的模拟示例。编码器的响应在波形中作为“边界输入 4”输入。当响应数据输入时，SHIFT_EN 信号作为“Counter2_mode0”生成。该信号标记包含在 CRC 计算内的位。在 SHIFT_EN 信号的每个脉冲上，响应数据被移入标记为“counter_2”的 LFSR。随着最后一个 SHIFT_EN 脉冲的出现，LFSR 保持值 0x1576608F，其中最终 8 位 0x8F 表示所需的 CRC 结果。

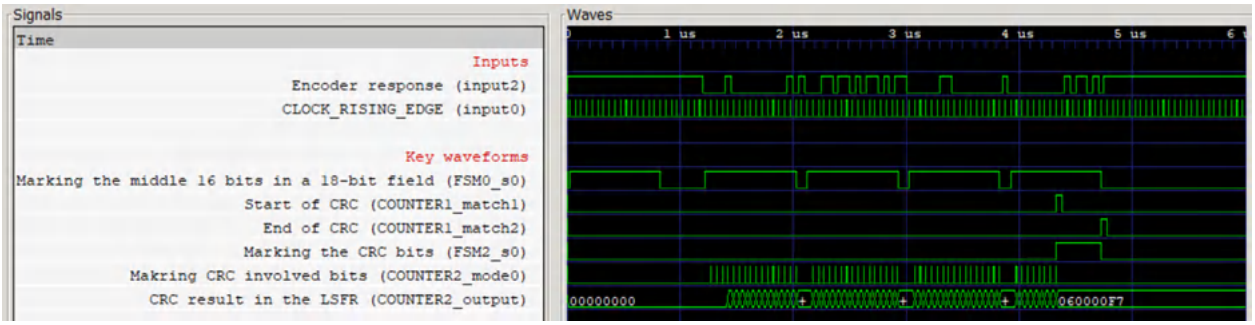

图 2-5. CLB 中 CRC 计算逻辑的仿真

2.3 软件更改

T-Format 解决方案中的大多数软件都可以在 A-Format 接口中重复使用。根据上述 SPI 和 CLB 实现，软件也有变化。

第一个是 tformat_configureCLBLen() API。此函数用于为 CLB 计数器配置适当数量的传输和接收时钟，这些时钟可能因命令而异。在 A-Format CLB 实现（用于 CRC 计算）中，CRC 掩码取决于编码器响应的时钟数，因此向该 API 添加了两个额外的配置。两行配置的 COUNTER_1_MATCH_1 和 COUNTER_1_MATCH_2 可以正确触发 FSM_2 并在编码器响应中标记 CRC。有关配置影响的详细信息，请参阅图 2-5。

```
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;
}
```

传递到 PM_tformat_setupCommand() 函数的参数的第二个更改。

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

这些参数包括帧数据、传输及接收数据的字段/时钟数以及 FIFO LEVEL。这些参数与接口协议本身相关，并受 SPI 实现一节中提到的 FIFO 分离的影响。下面示出了如何正确计算这些参数的示例。此示例基于 A-Format CDF0。

```
    #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);
```

由于 A-Format 中的 18 位字段被拆分为两个 9 位半字段，因此在填充 FIFO 时，最高有效半部分将首先填充，然后是最低有效半部分。与 T-Format 解决方案相同，TX FIFO 的其余部分会填充 0xFFFF，以便在接收期间将 TX 线路保持在高电平。下面是 FIFO 填充的示例。

```
    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;
```

最后，必须更改解码函数 PM_tformat_receiveDataID0_1_7_8_C()。此函数根据接口协议读取 SPI 缓冲器并且将原始数据解码为综合的数据结构。下文提供了 A-Format 中 CDF0 的解码响应示例。请注意，RX FIFO 也是拆分的。

```
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 验证

该接口解决方案在 Nikon A-Format 编码器上进行了测试。测试设置中包括以下硬件：

- LAUNCHXL-F280039C
- BOOSTXL-POSMGR
- 格式编码器 Nikon-MX50AHN00

图 3-1 中显示了测试装置的实物照片。

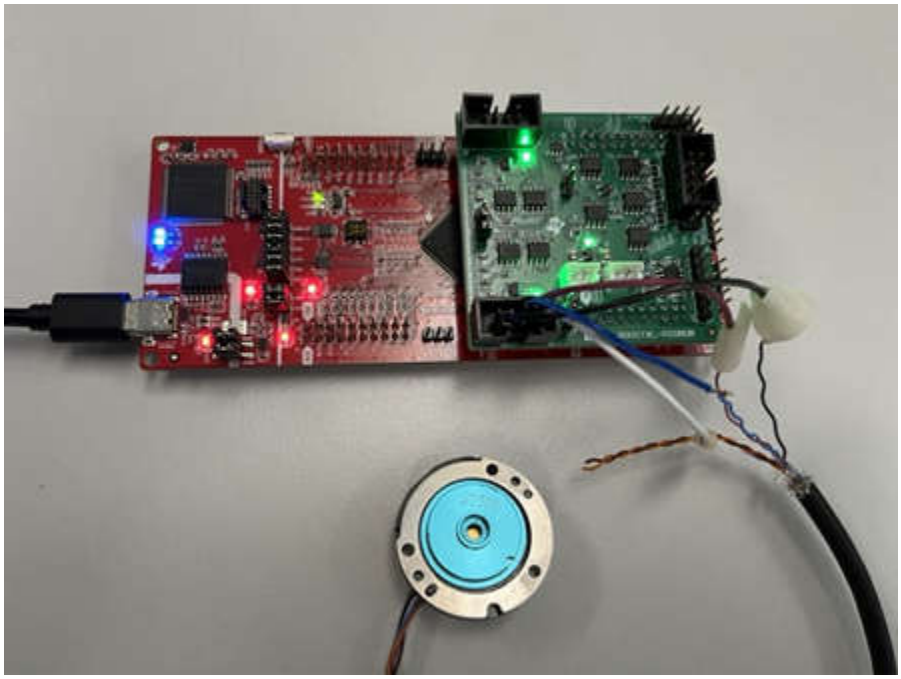

图 3-1. 验证设置

此测试设置的目标是验证 C2000 控制器是否可正确发送和接收 A-Format 帧。该测试的目的是获取可通过 CRC 校验的编码器响应。请注意，编码器的角度精度（机械性能）超出了该验证的范围，因此编码器未安装在实际电机上。

为了实现所需的测试目标，选择了命令帧 CDF0 进行测试。该命令帧请求编码器返回完整 40 位的位置数据。测试程序将首先对 CDF0 命令进行编码并填充 SPI 缓冲区，发出 CLB 启动信号来发送命令，并验证响应的 CRC。使用 CCS 表达式窗口将结果显示为变量。

首先，将 SPI 接收缓冲区数据（原始 9 位条目）与示波器波形（在 POCI 引脚上捕获）进行比较，如图 3-2 所示。SPI 缓冲器从三个字的虚拟数据开始，每个字为 9 位，如上所述。虚拟数据后跟编码器响应，其中包括 8 个字，每个字为 9 位，有效数据存储在每个 9 位条目的最低有效位 (LSB) 中。缓冲器同右侧的波形对齐。这种比较表明，SPI 和 CLB 已正确设置为传输及接收 A-Format 帧。

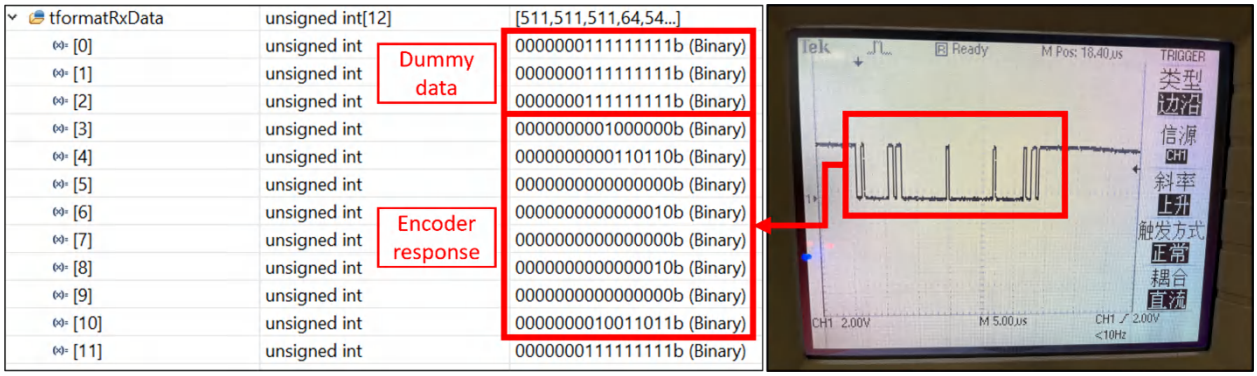

图 3-2. RX FIFO 数据和波形

接下来，将在图 3-2 中进一步检查解码的数据。

出于演示目的，通过将接收到的 CRC 和计算出的 CRC 组合成一个 32 位变量并在 CCS 的表达式窗口中显示该变量来验证 CRC。这样将确保 CCS 在一个通信周期内对 CRC 变量进行采样。因此，图 3-3 示出了 CSS 表达式窗口的屏幕截图，其中可以查看接收到的帧和解码的帧的原始数据，并且接收到的 CRC 与计算出的 CRC 对齐

同时，通过手动旋转编码器并观察解码后的角度位置和多圈数据的变化（与手动编码器旋转一致），验证从响应中解码的角度和多圈数据的行为。

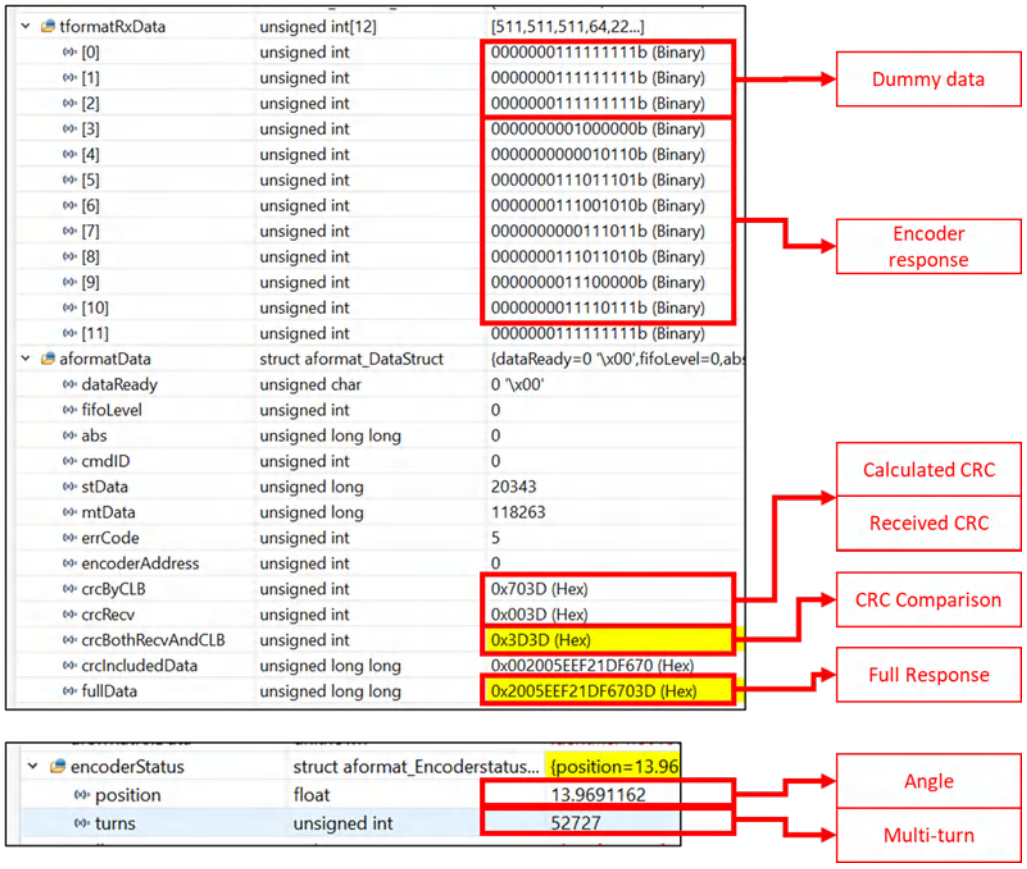

图 3-3. 完整解码编码器响应数据

4 总结

本应用手册提供有关利用 C2000 F280039C MCU 的可配置逻辑块 (CLB) 实现 Nikon A-Format 绝对编码器接口的指导。主要实现包括：1) 18 位字段拆分的 SPI 外设配置（9 位 FIFO 宽度）；2) SPI 时序控制的 CLB 逻辑定制及特定于 A-Format 的 CRC 计算 3) 用于命令/响应处理的软件帧解析。验证结果可确认接口正确发送/接收 A-Format 帧、通过 CRC 校验并且准确解码位置数据。该解决方案只需根据 TIDM-1011 T-Format 参考设计进行软件修改，即可减少开发人员的开发工作量。

5 参考资料

1. TIDM-1011 Tamagawa T-Format 编码器的接口参考设计。德州仪器 (TI)，2022。
2. CLB 工具用户指南（德州仪器 (TI)）
3. TMS320F280039C 微控制器数据表（德州仪器 (TI)）

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月