Design Guide: TIDM-1010

BiSS-C Absolute Encoder, Master-Interface Reference Design for C2000™ MCUs

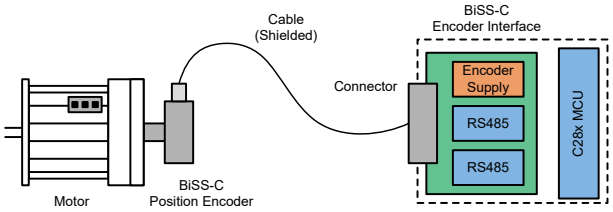
Description

C2000™ microcontroller (MCU) Position Manager technology offers an integrated solution to interface to the most popular digital and analog position sensors, which eliminates the necessity for external field-programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs). The Position Manager BoosterPack™ is a flexible, cost-effective platform intended for evaluating various encoder interfaces and designed to work with multiple C2000 MCU LaunchPad™ development kits. The software of this reference design specifically targets implementation of BiSS-C™, which is a digital, bidirectional interface for position encoders. The highly optimized and easy-to-use software library and examples included in this reference design enable BiSS-C position-encoder operation using the Position Manager BoosterPack.

Resources

TIDM-1010	Design Folder
LAUNCHXL-F28P65X	Tools Folder
SN65HVD78	Product Folder
TLV702	Product Folder
TPS22918-Q1	Product Folder

Ask the TI E2E™ support experts

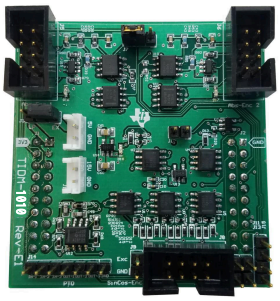


Features

- Flexible, Low-Voltage, BoosterPack Evaluation Platform for Position-Encoder Interfaces
- Integrated MCU Solution for BiSS-C Without Additional FPGA Requirements
- Easy Interface-to-BiSS-C Commands Through Driver Functions and Data Structure Provided by Library
- Library Support for Unpacking Received Data and Optimized Cyclic Redundancy Check (CRC) Algorithm
- Supports Clock Frequency up to 10 MHz and Verified Operation up to 100-m Cable Length
- Includes Evaluation Software Example Showcasing BiSS-C Software Library

Applications

- Industrial
- Motor Drives



1 System Description

Industrial drives, like servo drives, require accurate, highly-reliable, and low-latency position feedback. The BiSS protocol is designed for serial transfer of digital data between a sensor and a controller. BiSS stands for bidirectional serial synchronous. The BiSS interface was introduced by iC-Haus GmbH as an open-source protocol in 2002. BiSS-C mode is the continuous mode in which the BiSS-C interface reads out the position data cyclically.

The TIDM-1010 design implements a BiSS-C interface to a C2000 LaunchPad. BiSS-C is a pure serial, digital interface, based on the RS-485 standard. BiSS is capable of transmitting position values, along with other physical quantities, and allows reading and writing of the internal memory of the encoder. The transmitted-data types include absolute position, turns, temperature, parameters, and diagnostics. Figure 1-1 shows the major hardware blocks used in this design.

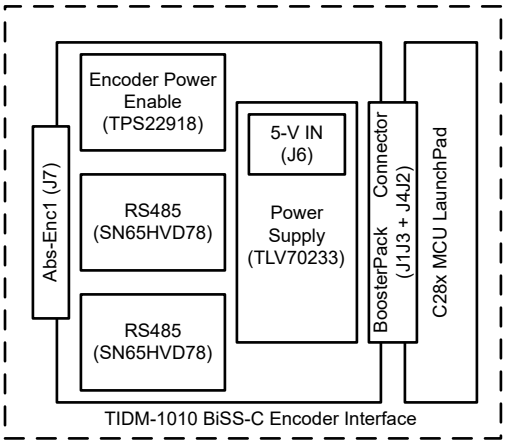


Figure 1-1. TIDM-1010 Hardware Blocks and Connectors

TIDM-1010 supports a point-to-point configuration which is typically used with the BiSS position or rotary encoders. The point-to-point topology is shown in Figure 1-2. In the point-to-point configuration, only one device with one or more sensors is operated on by the master. PM_bissc library only supports this configuration.

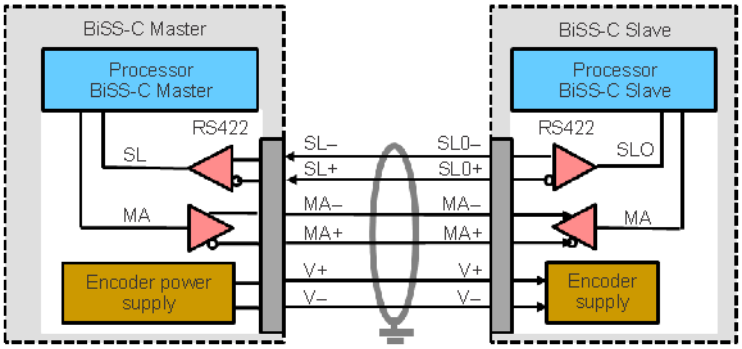


Figure 1-2. BiSS-C Point-to-Point Structure

The absolute position encoder connects to the TIDM-1010 through a 6-wire, shielded, cable. RS-485 is used as the physical layer. The 6 wires are:

- MA+ and MA-: Encoder clock (BiSS Master Clock) differential signals
- SL+ and SL-: Return data from the encoder (BiSS Slave return) differential signals
- V+ and V-: Power and ground supply for the encoder

The BiSS-C MA-clock frequency is variable and depends on the cable length (see Section 3.3.4). Depending on the encoder and the encoder cable, the maximum cable length or the maximum achievable clock frequency can vary. Because the quality of the cable has an impact on the communication performance, encoder manufacturers define these limits in their data sheets and recommend an appropriate cable for use with their encoders. More

details of the protocol and point-to-point configuration can be found at BiSS Interface: BiSS User Society and Internet Platform.

The Texas Instruments C2000 Position Manager BiSS-C (PM_bissc) encoder interface enables a BiSS implementation without external hardware such as an FPGA or CPLD. The reference implementation features:

- Up to a 10 MHz clock frequency with a 10 m cable length
- Integrated cable-propagation delay-compensation
- Software driver functions:
  - Perform a transaction with the encoder. This consists of sending the MA signal to the encoder and receiving the response.
  - Calculate a CRC
  - Compare the CRC received with the calculated CRC
  - Unpacking the response data

This reference implementation includes all of the source code. Any changes needed for the implementation can be made by users as required by their application.

Note

The library supports the basic-interface drivers for

- Single-cycle-data (position + error + warning + CRC) transaction
- Single register read access
- Single register write access

All the higher-level application software and BiSS features can be developed by users using the basic interface provided by this implementation.

1.1 Key System Specifications

Table 1-1. Key System Specifications

PARAMETER	SPECIFICATIONS	DETAILS
Input voltage	5 V(1)	Section 3.3.1
Output voltage (encoder)	5 V	Section 3.3.1
Protocol supported	BiSS-C point-to-point	BiSS
Maximum Frequency	10 MHz	Supported up to 10-m cable.Section 2.3.1.1
Encoder bits	BiSS-C protocol standard	BiSS
Position data CRC	x^6 + x + 1	Polynomial for position data (single cycle data) verification
Control data CRC	x^4 + x + 1	Polynomial for control data verification
Control data features	Single register read, single register write	
CPU cycles	Section 3.3.5	
Code size	Section 3.3.5	

(1) The encoder connected to the TIDM-1010 device determines the current limit required of this supply. TI recommends using a generic, bench-top, adjustable, power supply with an adjustable current limit instead of the TIDM-1010 generated voltage.

2 System Overview

The C2000 BiSS TIDM-1010 reference design is a combination of hardware and software. The core hardware components are a C2000 real-time microcontroller (MCU) and a RS-485 transceiver. The C2000 LaunchPad and the TIDM-1010 boosterPack, which contains the RS-485 transceiver, are the boards used in this implementation. The C2000Ware Motor Control SDK package contains the necessary software. The software includes a library, which implements key BiSS interface features, along with a system-level example to demonstrate BiSS-C communication.

The BiSS encoder interface leverages the C2000 CLB (Configurable Logic Block) and the SPI (Serial Peripheral Interface) modules. The CLB controls the MA clock, SPI clock, and compensates for cable propagation delay. The SPI module acts as the receive interface to the RS-485 physical layer. The firmware, which is written in C, runs on the C28x of the C2000 MCU.

The C2000 LaunchPad can provide power for the TIDM-1010 RS-485 transceiver and 5V for the encoder. 5V can also be supplied separately if the encoder specifications require a higher current than the LaunchPad can provide.

During start-up, the application running on the C28x initializes the MCU clocks and configures the pin-mux. The MCU's SPI and CLB are also configured as required to send and receive data.

2.1 Block Diagram

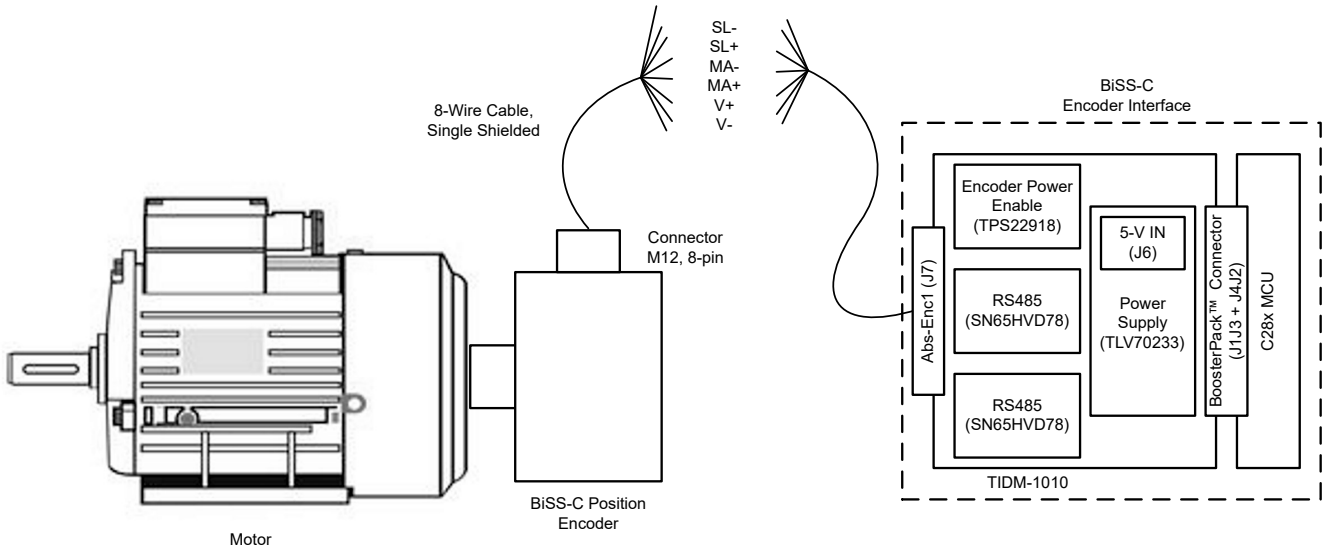


Figure 2-1. TIDM-1010 System Block Diagram

2.2 Highlighted Products

The TIDM-1011 reference design hardware consists of a C2000 LaunchPad plus a BOOSTXL-POSMGR BoosterPack. This section covers the key devices used. For more information on each of these devices, see their respective product folders at TI.com.

2.2.1 C2000 Real-Time MCU LaunchPad

The provided TIDM-1010 example is based on the TMS320F28P650DK9 MCU. The F28P65x provides 600 MIPS of total system performance between dual, 200-MHz, C28x CPUs and a real-time-control coprocessor (CLA). This powerful MCU contains 1MB of onboard flash and includes highly-differentiated peripherals, such as 16-bit or 12-bit analog-to-digital converters (ADCs), comparators, 12-bit digital-to-analog converters (DACs), delta-sigma sinc filters, HRPWMs, eCAPs, eQEPs, CANs, and more.

The F28P65x also features the Configurable Logic Block (CLB). The BiSS interface makes extensive use of the CLB. The CLB peripheral allows users to incorporate custom logic without the need for an external FPGA or CLPD. The CLB is composed of submodules that combine together to enable custom digital logic. Submodules include: Finite State Machines (FSM), Lookup Tables (LUT), and counters. The CLB also interfaces with existing

on-chip control peripherals to enhance functionality and provide design options. The BiSS interface can be ported to any C2000 device which includes the CLB.

The LAUNCHXL-F28P65X evaluation board is leveraged for the TIDM-1010 implementation. The LAUNCHXL-F28P65X LaunchPad is a low-cost development board for the F28P65x devices.

2.2.2 SN65HVD78

The SN65HVD78 device combines a differential driver and a differential receiver, which operate from a single, 3.3-V power supply. The driver differential outputs and the receiver differential inputs are internally connected to form a bus port suitable for half-duplex (two-wire bus) communication. These devices feature a wide, common-mode voltage range, which make the devices suitable for multipoint applications over long cable runs.

Find the full device features and specifications at the SN65HVD78 product folder.

2.2.3 TLV702

The TLV702 series of low-dropout (LDO) linear regulators are low-quiescent current devices with excellent line and load-transient performance. All device versions have thermal shutdown and current limit for safety. The devices regulate to specified accuracy with no output load.

Find the full device features and specifications at the TLV702 product folder.

2.2.4 TPS22918-Q1

The TPS22918-Q1 is a single-channel load switch, with configurable rise time and configurable quick-output discharge. The device contains an N-channel MOSFET that can support a maximum, continuous current of 2 A. The switch is controlled by an on and off input, which can interface directly with low-voltage control signals.

Find the full device features and specifications at the TPS22918-Q1 product folder.

2.3 Design Considerations

This section provides:

1. Overview of the BiSS-C interface protocol.
2. Overview of the C2000 BiSS-C encoder interface.
3. TIDM-1010 hardware (BOOSTXL-POSMGR BoosterPack) implementation.
4. C2000 MCU implementation, including the required input/output, CRC calculations, and the CLB design.
5. Overview of the C2000 BiSS-C encoder interface software library.

Note

This section provides implementation details only. For information related to:

- Hardware requirements, setup and testing: Refer to Section 3
- Software: Installing and running the software: Refer to: *C2000 BiSS-C Encoder Interface Software Guide* (HTML, PDF). The software guide includes documentation for:
  - Communication demonstration
  - BiSS-C application programmer interface (API)
  - Incorporating the library into your own solution
  - Porting the library to CPU2 on a dual-core device

2.3.1 BiSS-C Protocol

The BiSS-C interface specification describes a serial interface protocol to communicate with encoders and other sensors. BiSS-C allows the simultaneous transmission of position data and the control data over the same line. The interface is similar to the serial synchronous interface (SSI) protocol where the data transmission is synchronized with the controller clock signal.

Table 2-1. SSI Compared to BiSS-C

SSI (Serial Synchronous Interface)	BiSS-C
2Mbits / s	Up to 10Mbits / s on 10 meter cable
N/A	Cable length propagation compensation

Table 2-1. SSI Compared to BiSS-C (continued)

SSI (Serial Synchronous Interface)	BiSS-C
Differential twisted pair cable (RS-422/485)	Differential twisted pair cable (RS-422/485)
Unidirectional. Only supports data transmission from encoder to controller.	Often used uni-directionally like SSI. Also supports bidirectional data transmission enabling the encoder configuration can be controlled by the drive.
Supports a simple parity check	Robust error checking by Cyclic Redundancy Check (CRC)
N/A	Encoder can request additional processing time
Point-to-point topology only	Typically point-to-point, but the protocol also supports daisy chained encoders. The C2000 design discussed in this document implements a point-to-point topology.

Note

This document describes some important concepts of the BiSS-C frame and control communication. For the detailed specifications, see *BiSS User Society and Internet Platform*

Though position encoders typically provide the feedback-position data of motors, this encoder allows closed-loop control of motors. This design implements the BiSS-C interface for point-to-point communication. In the point-to-point configuration, only one such encoder is interfaced to by the controller or drive.

In the BiSS-C protocol, the controller provides a clock (MA) to the encoder and then the encoder sends data back to the controller (SL). The communication data consists of single-cycle data (SCD) and control data (CD).

- SCD is the primary data transmitted from the encoder to the controller and is updated every BiSS cycle. The SCD includes the absolute position, error bit, warning bit and a CRC check. The absolute position can be single-turn only (one revolution) or can include both single-turn and multi-turn (number of revolutions) data.
- CD enables the controller to write and read encoder registers. The controller sends one control data bit (CDM) to the encoder, and the encoder returns one control data bit (CDS) back to the controller each cycle.

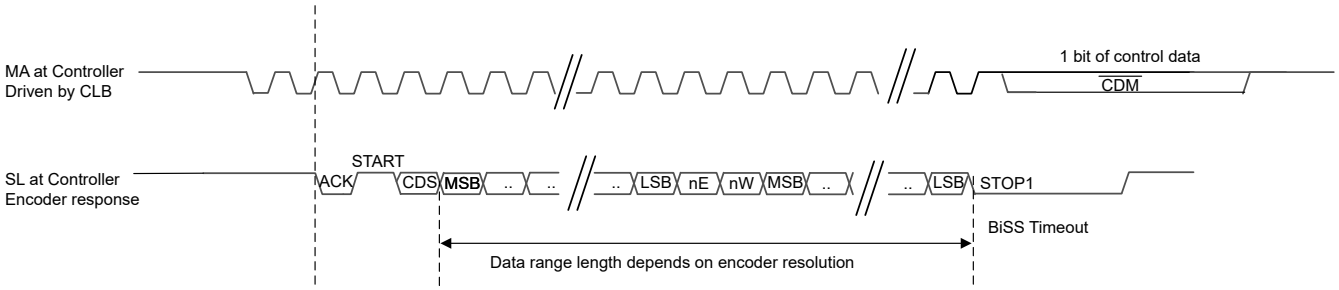


Figure 2-2. BiSS-C Frame

Figure 2-2 shows an example BiSS frame. In the reset state, both the MA and SL lines are active high. The controller starts the frame by sending the clock over the MA line. On the second rising edge of the MA clock, the encoder responds with a low signal to acknowledge (ACK) the BiSS frame. For the next MA clock cycle, a start bit is asserted by the encoder. Following START, the encoder sends a control data (CDS) bit, which is the response to the control CDM bit sent in the previous frame.

After the CDS bit, the encoder sends the position data starting with the most significant bits (MSBs). The position data is followed by an error bit (nE) and a warning bit (nW). The encoder then sends the cyclic redundancy check (CRC) bits with the MSB first. The CRC is sent inverted by the encoder over the SL line.

After sending all the bits, the encoder goes into the BiSS time-out state driving SL to a low signal level. SL then goes high when the encoder is ready for the next transmission or expiration of the BiSS time-out. The inverted state of the MA clock line during the BiSS time-out is the CDM bit for control communication. One control data bit is sent in each BiSS frame by the controller and one bit is sent back by the encoder.

Note

An encoder has an SLI (input) and an SLO (output) signal. In the point-to-point configuration, the SLO signal is directly connected to the SL of the controller.

2.3.1.1 Line Delay Compensation

In a real-world application environment, the encoder can be far from the controller. A long-cable connection between the encoder and the controller can delay transmission and physical noises.

Line delay is the propagation delay due to the length of the cable used in the transmission. When the controller starts sending the MA clock, some time is required for the clock to reach the encoder. When the encoder receives the clock, the encoder begins responding with SL data. The encoder's response also travels the reverse path to the controller through the cable. The time delay in the transmission of data over the wire is proportional to the length of the cable. With a cable length up to 100 meters, a cable delay of 1µs from the time the controller sends the clock until the controller receives the encoder's response is possible.

The BiSS-C interface has mechanism to compensate for line delay and avoid errors in the transmission of longer cables.

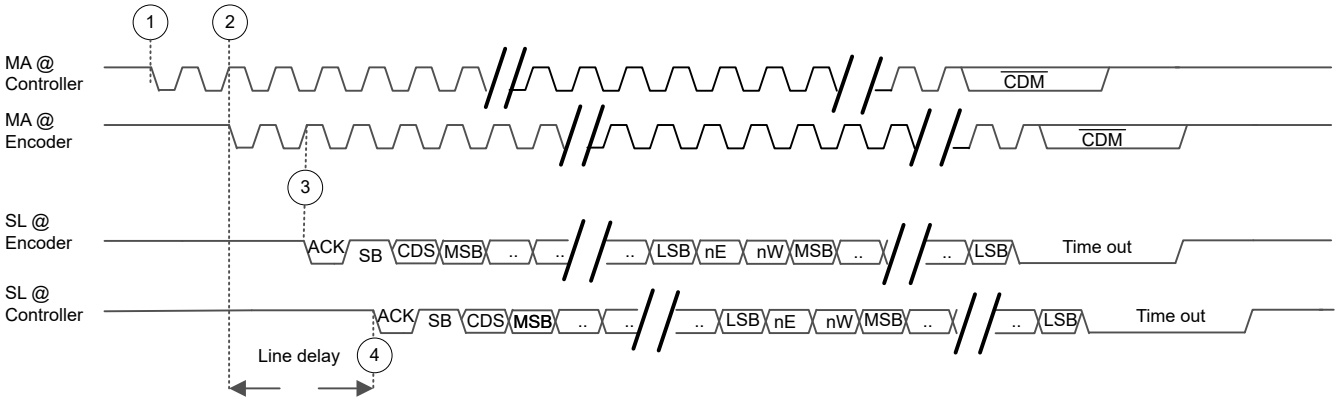


Figure 2-3. BiSS-C Line Delay

Figure 2-3shows the signals in two perspectives: at the controller and at the encoder.

Refer to Figure 2-3 marker (1):

The MA at Controller line shows how the clock looks at the BiSS-C interface. This is on the cold side where the drive controller is located. The controller begins the transaction by sending the MA signal.

Refer to Figure 2-3 marker (2):

Due to the line delay, the MA clock signal is delayed at the encoder (motor). The MA at Encoder line shows the delay.

Refer to Figure 2-3 marker (3):

The encoder responds to the second rising edge of delayed MA clock. The SLO at Encoder line shows the response of the encoder to MA at Encoder.

Refer to Figure 2-3 marker (4):

The response takes some time to travel back to the controller. Traveling to the controller is delayed as shown in the SLO at Controller signal. By measuring the time duration between second rising edge of the MA clock and the first falling edge of the SLO line, the total time delayed can be calculated. To avoid the transmission errors, BiSS-C interface compensates for this line delay.

2.3.1.2 Processing Time Request by Encoder

An encoder can request processing time before sending the sensor data. Extra time is required for operations like analog-to-digital conversion and memory access. The encoder indicates the processing time by delaying the

START bit (SB). The controller must check whether encoder is requesting processing time and provide additional clock cycles.

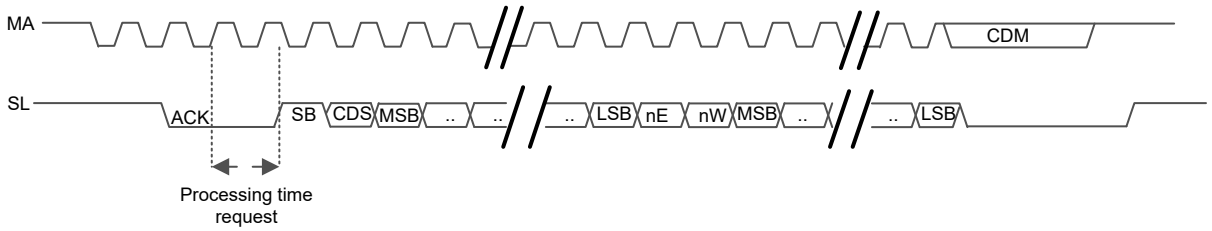


Figure 2-4. Encoder Processing Time Request

2.3.1.3 Control Communication

In BiSS-C communication, the controller can send a control frame over the MA line without interrupting the position-data communication. This is accomplished by sending one bit of the control frame within each BiSS frame.

As described in Section 2.3.1, the controller sends one data bit, known as CDM, per BiSS frame. Likewise, the encoder responds to these CDM bits with one-bit of the response, known as CDS, per BiSS frame. This is repeated until a complete control frame is sent, and response received, over several BiSS frames.

The BiSS-C control frame has two types:

- Register communication frame: a read or write of an internal register within the encoder
- A command frame: sends a command to the encoder

Note

As provided, TIDM-1010 does not implement the command frame. This feature can be added to the design if required by the system developer. In the command frame, the control select bit (CTS) is zero (CTS = 0). Using the command frame to support a multipoint connection is beyond the scope of this design. Therefore, this document focuses on the register communication frame only.

The following steps describe a read or write access. Refer to Figure 2-5 and Figure 2-6.

1. The controller sends at least 14 BiSS-C frames with CDM = 0
2. CDM = 1 indicates the start bit, S, of a control frame.
3. The next CDM bit is referred to as CTS (control select bit). For a register access CTS is 1.
4. The controller then sends a 3-bit ID to identify the slave being accessed.
5. The ID is followed by a 7 bit register address and a CRC.
6. The next 3 bits are a read bit (R), a write bit (W) and a start bit (S). R W S are defined as follows:
   - Write access: RWS equals 011b
   - Read access: RWS equals 101b
7. The controller either:
   - Holds the CDM bit low for a read access
   - Sends the 8-bit data + CRC to be written for a write access
8. A stop bit (P) indicates the end of the control frame.

Note

In step 6, the protocol allows the encoder to request additional processing time for the read or write. This is done by responding with S = 0 (instead of S = 1 as shown). This is not supported by the current implementation but can be added by the developer with updates to the CD state machine C code. This extra time is not required by all encoders. Refer to the specifications for your particular encoder.

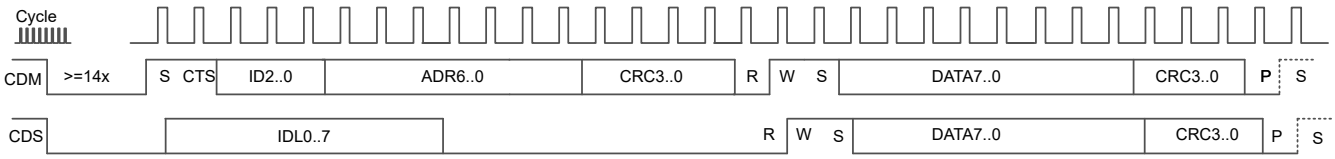


Figure 2-5. Control Frame: Register Read

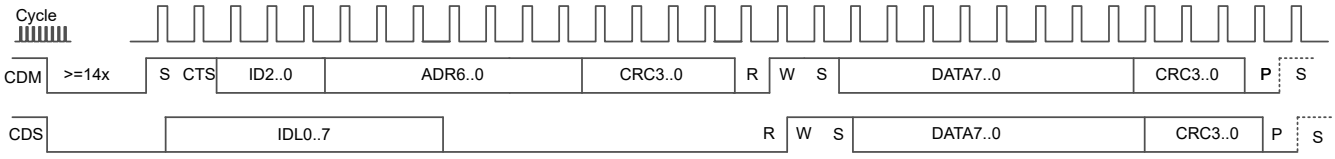


Figure 2-6. Control Frame: Register Write

Note

The BiSS protocol allows for back-to-back reads, or back-to-back writes, of consecutive registers. This is enabled when the controller sends a stop bit (P = 1) immediately followed by another start bit (S = 1). This feature is not implemented in TIDM-1010. Only a single read, or single write, per control frame is supported.

2.3.2 C2000 BiSS-C Encoder Interface Overview

Communication over a BiSS-C encoder interface is primarily achieved by the following components:

- CPU (C28x)
  - Configures the device, CLB and SPI
  - Initializes CLB counters to generate the proper the MA clock frequency and clock counts for the encoder's resolution
  - Packs and unpacks data
  - Calculates the single-cycle-data CRC and control frame CRC
  - Compares calculated CRC with received CRC
- Configurable logic block (CLB)
  - Sends the MA clock and CDM bit
  - Monitors the SPI PICO signal for the encoder's response. Controls the SPI clock to read the response
  - Measures, and compensates for, cable propagation delay as required by the interface
- Serial peripheral interface (SPI)
  - Receives the encoder's response
- Device interconnects (XBARs, CLB XBARS)
  - Routes signals into and out of the CLB and the device
- External interface block
  - TIDM-1010 board with RS-485 differential line driver

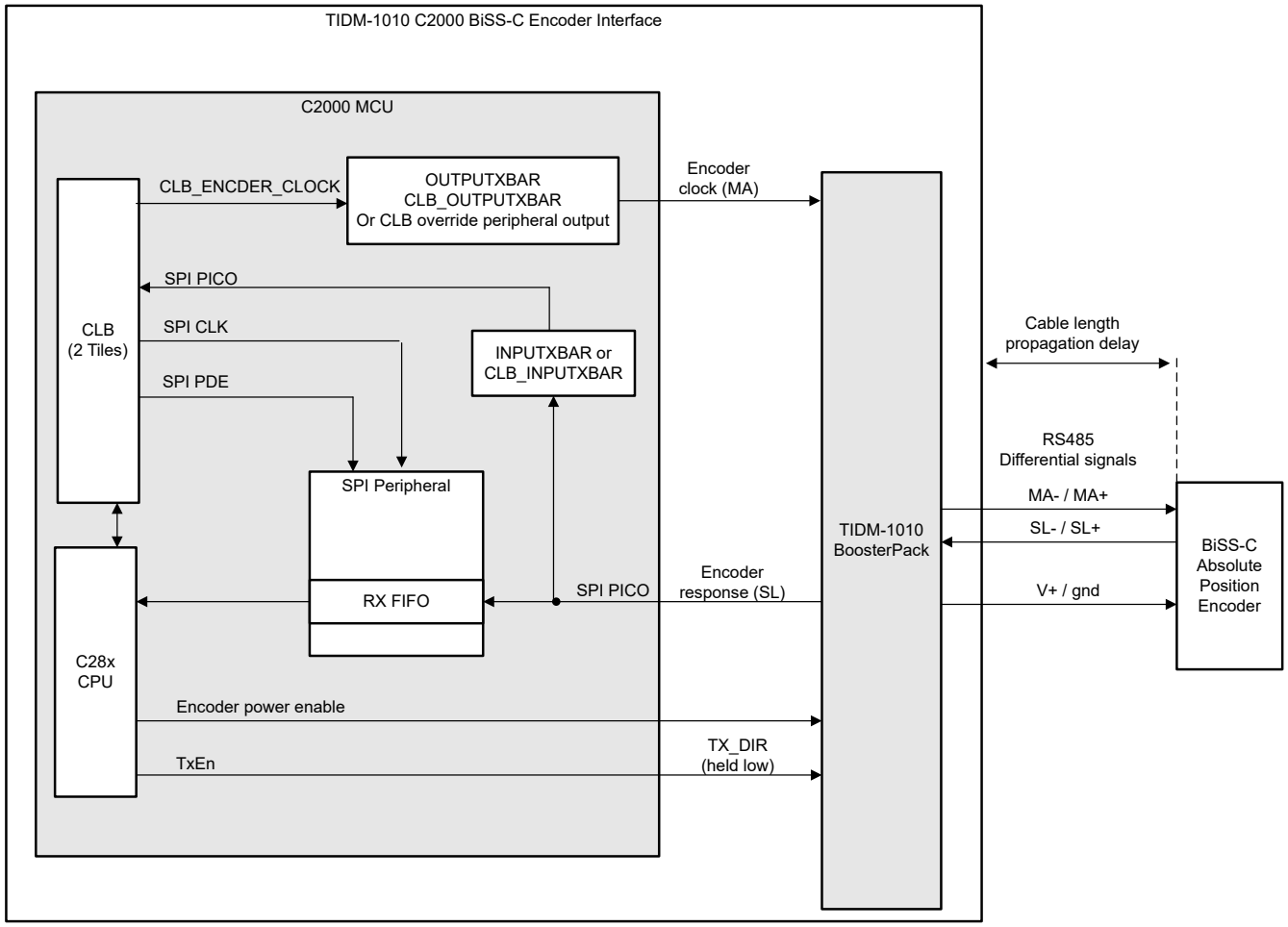


Figure 2-7. Encoder Interface Implementation Block Diagram

Note

On the F2837xD / F2837xS / F28007 devices the CLB can not override the SPI input signals directly. The TIDM-1010 hardware can route the CLB generated SPI clock back to the peripheral clock input pin and tie the SPI PDE pin to ground. Refer to the TIDM-1010 schematics for more information.

The remainder of this section describes the following aspects of the design:

- The TIDM-1010 hardware
- C2000 MCU resources including the CLB
- C2000 software used by the encoder interface

2.3.3 TIDM-1010 Board Implementation

The TIDM-1010 board provides the following:

- Differential line driver and receiver for RS-485 communication between a C2000 MCU and the encoder.
- TxEN signal routed from the MCU to the direction control of the RS-485 driver/receiver. For the BiSS-C implementation, this signal is held low.
- SPICLK signal routed to a GPIO where it can be controlled by the CLB peripheral. This connection is optional for all supported devices except the F2837xD, F2837xS and F28007x devices. On other devices, the CLB is capable of clocking the SPI peripheral within the device itself.

- **Note**

  The TIDM-1010 board is identical to the Position Manager BoosterPack (BOOSTXL-POSMGR), which means the TIDM-1010 board can interface with several other position encoder types. The board is fully populated by default for future compatibility. This reference design focuses on the BiSS-C, and the hardware blocks not mentioned in this document can be ignored.

Table 2-2 lists the connectors on the TIDM-1010 BiSS-C implementation and the functions of these connectors.

Table 2-2. TIDM-1010 Board and BOOSTXL-POSMGR Connectors

CONNECTOR	DESCRIPTION	USED BY TIDM-1010
Abs-Enc-1 (J7)	BiSS-C and other absolute encoders	Yes, LaunchPad Site 2
Abs-Enc-2 (J8)	BiSS-C and other absolute encoders	No
Abs-Enc-2 Breakout (J10)	Allows two absolute encoders at site two using jumpers	No
SinCos (J14)	SinCos encoder	No
Resolver (J14 and J15)	Resolver interface with 15-V excitation circuitry	No
PTO (J17)	Pulse-train output	No
J1, J3 and J4, J2	BoosterPack connector	Yes
J6	5-V DC supply input	Yes
J16	15-V DC resolver excitation input	No

Figure 2-8 describes the encoder support at each site of the LaunchPad development kit.

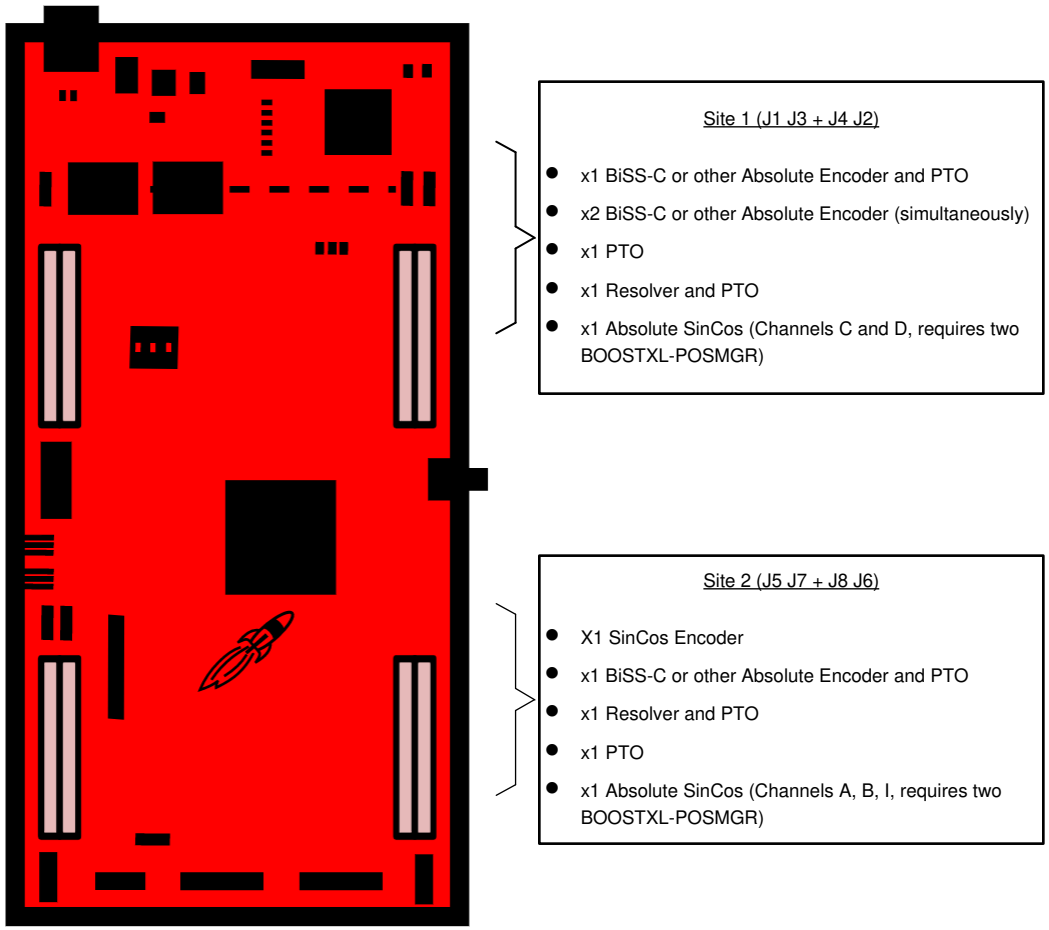


Figure 2-8. TIDM-1010 Board and BOOSTXL-POSMGR Encoder Support

As provided, TIDM-1010 uses LaunchPad Site 2 and BOOSTXL-POSMGR's Encoder 1 connections. Figure 2-9 shows the connections. The complete schematic of the TIDM-1010 BoosterPack can be downloaded from the BOOSTXL-POSMGR product page.

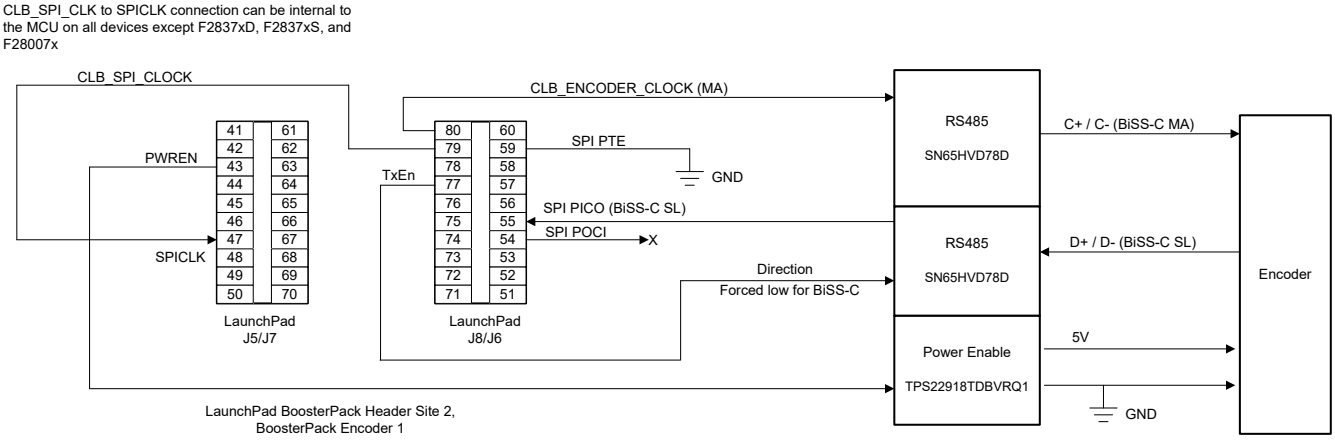


Figure 2-9. BoosterPack Block Diagram

2.3.4 MCU Resource Requirements

Table 2-3 lists the C2000 Real-Time MCU resources used by the TIDM-1010 reference design.

Table 2-3. TIDM-1010 Resource Usage

RESOURCE NAME and Quantity	TYPE	PURPOSE
CLB x 2	Type 1 or later	Provides the SPI clock, delay compensation, and CDM bit control. If the tile instance is changed, then routing in/out of the CLB must also be updated.
GPIO x 3	I/O	• CLB output, MA encoder clock • CLB output, RS-485 direction control (TxEN). For BiSS-C this signal is held low. • CPU encoder power control (PwrCtl). This signal is not required if the encoder power is controlled in another manner (example: externally powered).
GPIO x 1	I/O (F2837xD, F2837xS and F28007x only)	• CLB output of CLB_SPI_CLK • CLB Type 1: route this pin externally to the SPICLK input. • CLB Type 2, or later: clock the SPI module directly from the CLB. An external connection is not required, but can be useful for test and debug.
INPUTXBAR or CLB_INPUTXBAR x 1	Module, I/O	Connect the SPI PICO pin to the CLB input.
OUTPUTXBAR or CLB_OUTPUTXBAR or CLB override of peripheral output x 2	Module, I/O	• Connect a CLB output to ENCODER_CLOCK (MA) GPIO • Connects the CLB to TxEN GPIO
SPI x 1	Module and I/Os	One SPI instance to receive the RS-485 physical layer data signal. The SPI clock is controlled by the CLB.
CPU and Memory	Module	CPU and memory use for various functions.

2.3.4.1 Input, Output Signals, and CLB Tiles

This section describes the input or output and CLB tile connections used on the P65x device. When porting to another device, different routing or signal usage is sometimes required.

1. The specific GPIO pins and SPI module used depend on the device-specific LaunchPad pinout.
2. The connections into, and out of, the CLB depend on the features of that device and the pin. For example using a device INPUTXBAR instead of a CLB_INPUTXBAR.
3. The specific CLB tile instances depend on the capability of the tile to override other signals such as the SPICLK. For example, if SPI-D is used, then the design is best implemented on the tile that has access to SPI-D directly.

Note

In the input/output diagram:

- Letters in a colored circle indicates an off-page connection within a CLB tile diagram in a following section.
- Letters followed by _RE indicate *rising edge*. For example: D is the encoder clock. D_RE is the same signal after the CLB's rising edge filter is applied.
- G is an output from a finite state machine and referred to as the FRAME_STATE. G consists of two state signals: G.s0 and G.s1.

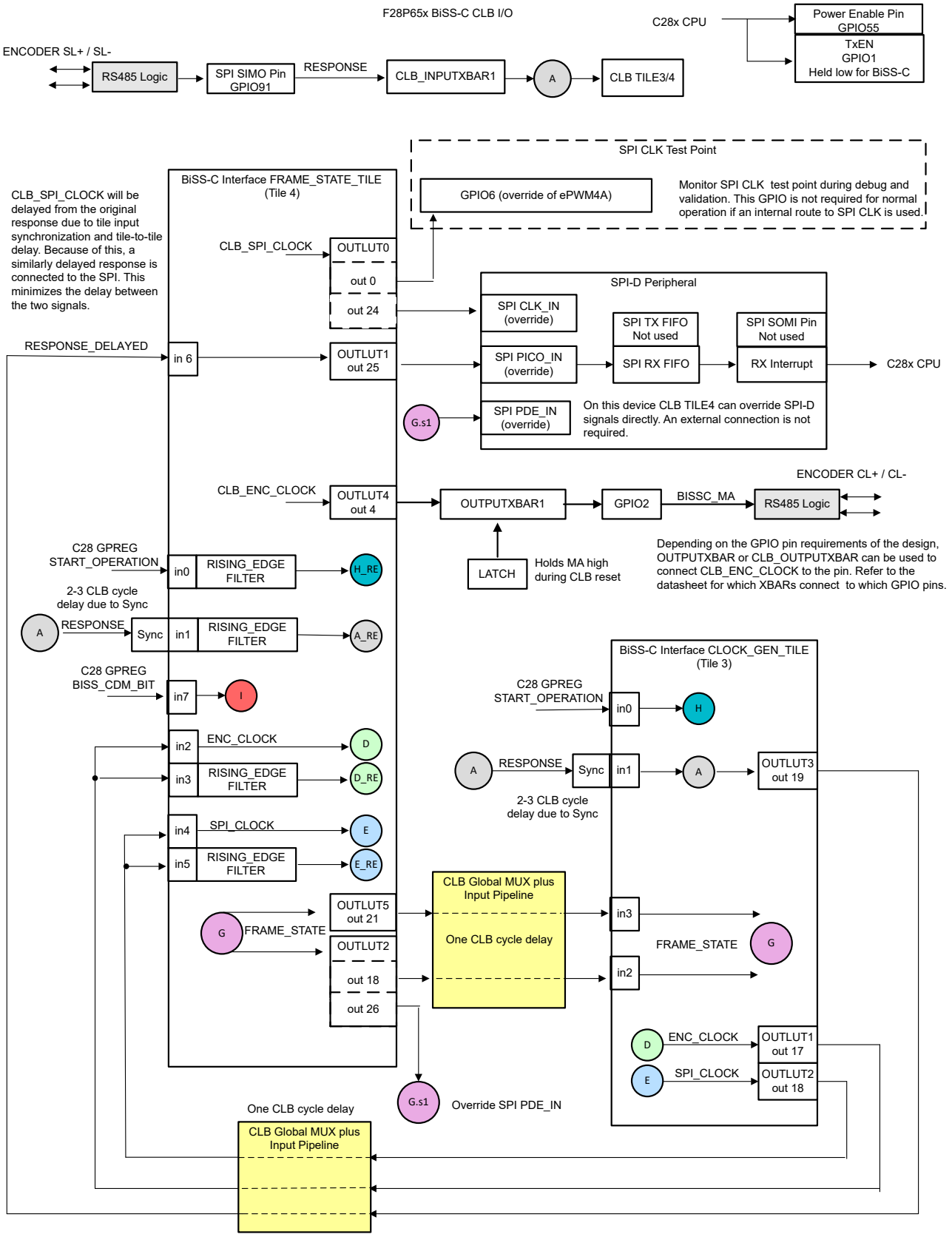


Figure 2-10. F28P65 Input, Output, and CLB Usage for BOOSTXL_POSMGR

2.3.5 CLB BiSS-C Implementation Details

The CLB is responsible for:

- Generating the encoder clock, MA, signal with the CDM bit.
- Monitoring the SPI PICO signal for a response from the encoder.
- Aligning the SPICLK to the incoming response.
- Clocking the SPI to receive the response.

This section describes the design of the CLB tiles using two approaches:

- Visualization of the CLB behavior during each phase of the transaction using waveforms.
- The CLB tile design including interconnect of the submodules.

2.3.5.1 Transaction Waveforms

When implementing a CLB design, first visualizing the required CLB behavior using waveforms can be helpful. To do this, first consider an an example transaction. Recall a BiSS-C transaction consists of the MA signal plus the encoder's response. A transmission can be broken up into FRAME_STATEs as shown in Figure 2-11. The first step is to map each element of the transaction to a CLB submodule. An example mapping is shown in Table 2-4.

Table 2-4. BiSS-C Transaction to CLB Mapping

Transaction Behavior	CLB Mapping
Track the FRAME_STATE	Finite State Machine (FSM): transitions to a new state given the previous state and current inputs.
Generate two clock signal of a specific width: one for MA and one for the SPI clock.	For each signal, this requirement maps to a COUNTER. Leverage the match values to generate the timing for the rising/falling edges. A LUT (Lookup Table) then generates the actual edges based on this timing.
Transmit a specific number of clock pulses for MA and for SPI clock.	For each clock, this maps to a COUNTER. On the edge of the clock the COUNTER increments and a MATCH indicates when the number is reached.
Align the SPI clock with the encoder's response	The COUNTER generating the SPI clock can be configured such that the edge transition is properly aligned with the encoder's response.
Only allow the SPI to be clocked when receiving the encoder's response	A LUT blocks the clock when the clock is not needed
During the BiSS timeout, hold the MA signal high or low (CDM bit)	Using GPREG input, the C28 can tell the CLB if the CDM must be high or low. This combined with a LUT then hold the value on MA after the specific number of clocks have been transmitted.
Control the TxEN	For BiSS-C, the TxEN is held low. The CLB output LUT can apply a constant "0" to an output.
Tell the CLB to start the transaction by sending MA	The C28x configures the COUNTER and SPI for the transaction. The CLB GPREG allows the C28x CPU to directly change a CLB input to start a transaction.

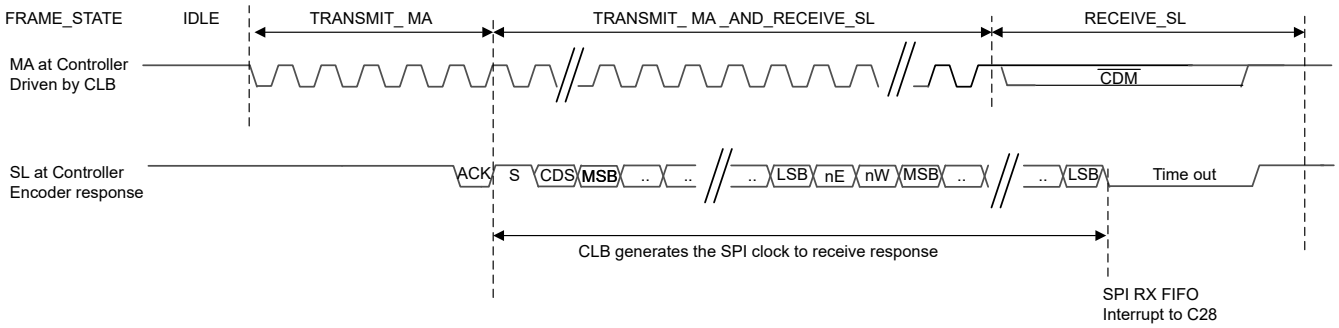


Figure 2-11. BiSS-C Single Cycle Data Transaction Example

The next step is visualization of the specific submodule behavior. Start with a quick sketch and then add additional detail as the design develops. Figure 2-12 shows an example waveform that was generated using the CLB SystemC simulation model with a custom input as the encoder's response. While Figure 2-12 was generated by the actual design, the figure's detail is similar to a preliminary sketch.

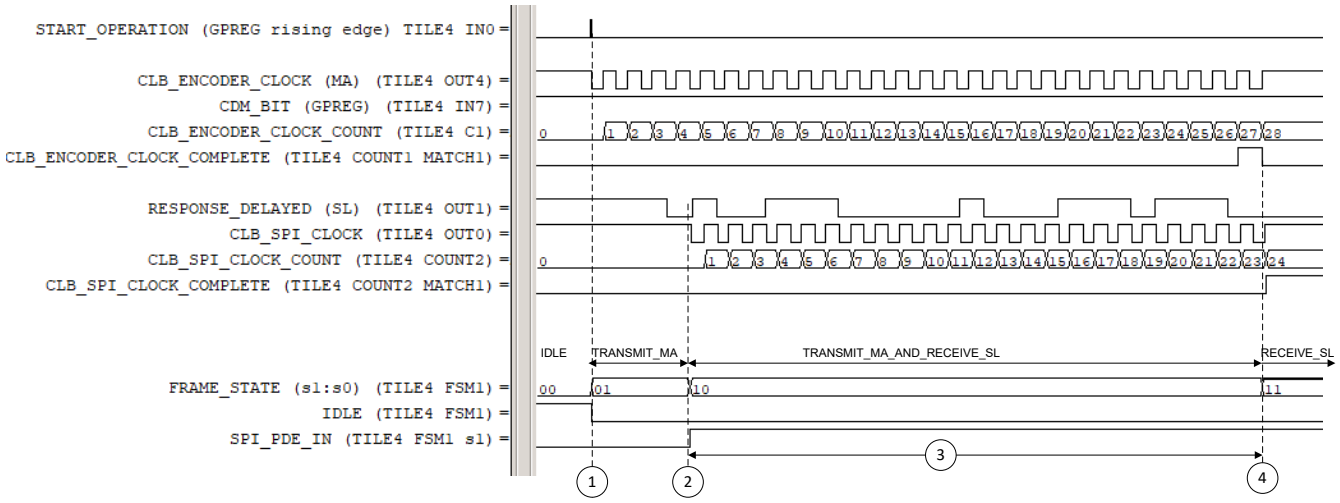


Figure 2-12. CLB Communication Waveform

Markers 1 - 4 in Figure 2-12 are used in the following sections to describe specific behavior of the design with respect to that marker. The markers are:

1. Transition from IDLE to TRANSMIT_MA
2. Transition from TRANSMIT_MA to TRANSMIT_MA_AND_RECEIVE_SL
3. During TRANSMIT_MA_AND_RECEIVE_SL
4. Transition to RECEIVE_SL

During IDLE there is no activity on the interface. The C28x must initiaite the transaction by:

- Configureing the CLB and SPI
- Pulling the START_OPERATION signal high through the CLB Tile's GPREG register. GPREG is the CLB's general purpose register which allows the C28x to directly control a tile's inputs.

Refer to: Figure 2-12**, marker (1).**

START_OPERATION remains high for one CLB_CLOCK because the CLB's rising edge filter is enabled for that input. At marker (1) the main state machine (FSM1 on Tile4) responds by moving FRAME_STATE from IDLE to the TRANSMIT_MA state.

During TRANSMIT_MA:the encoder interface sends the MA signal to the encoder. This signal is a clock with a specific duty cycle and defined number of clock cycles. The number of cycles depends on the resolution of the encoder. While clocking MA, the CLB monitors the SL line for an encoder response.

Refer to: Figure 2-12**, marker (2).**

To detect the encoder's response, the CLB monitors SL for a rising edge. The rising edge corresponds to the START bit after the ACK. The time required to detect the response can be any number of clocks and depends on the state of the encoder and the cable length. In addition, if the encoder requires extra time to respond, the ACK state is extended.

When the encoder's START bit is detected, the FRAME_STATE transitions to send MA and receive SL.

Refer to: Figure 2-12**, marker (3).**

The CLB starts clocking the SPI peripheral to receive the response.

- The CLB aligns CLB_SPI_CLOCK to the response. The response is sampled on the rising-edge of the CLB generated clock.

- The CLB_SPI_CLOCK is connected to SPI_CLK_IN by directly overriding the input to the peripheral.
- A counter tracks the number of SPI clocks generated. Match1 of this counter indicates when the required SPI clocks have been sent. Additional SPI clocks, beyond what is required by the response length, are sometimes needed to fill the FIFO of the SPI to the FIFO interrupt level.

Refer to: Figure 2-12**, marker (4).**

2.3.5.2 FRAME_STATE Generation

This section discusses in depth the FRAME_STATE generation. The FRAME_STATE is responsible for determining which of clocks are generated at any given time within the transaction.

The FRAME_STATE (FSM_1 s1, s0) transitions through 4 states:

- IDLE:
  - No activity
  - The CLB waits for the rising edge of the START_OPERATION signal to begin transmitting the encoder clock.
- TRANSMIT_MA:
  - Transmit the encoder clock
  - Monitor the response for the ACK from the encoder
- TRANSMIT_MA_AND_RECEIVE_SL
  - Continue to transmit the encoder clock
  - Start the SPI clock to receive the response
  - During this time both the encoder clock and the SPI clock are active.
- RECEIVE_SL:
  - The transmission of encoder clock is complete
  - The CDM bit is active on the MA signal
  - Finish receiving the response by clocking the SPI.
  - Depending on the length of the data and the SPI width used, additional SPI clocks can be generated to trigger a SPI interrupt.

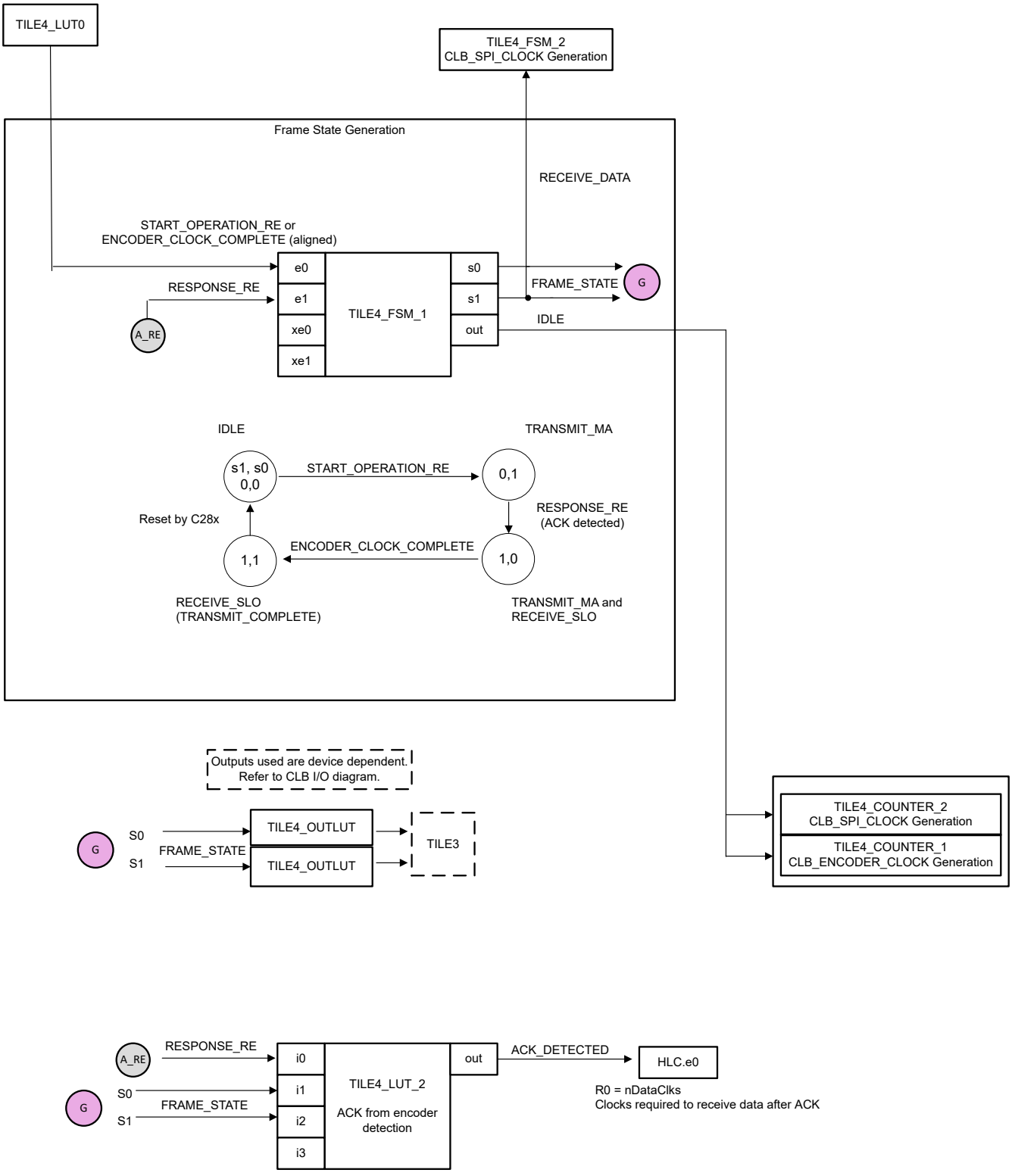


Figure 2-13. FRAME_STATE Generation

One method to derive the corresponding equations is to use a Karnaugh map (Table 2-5, and Table 2-6). The resulting equations are combined by an OR operator and entered into the CLB tool. The equations do not need to be reduced to the simplest form.

Table 2-5. FRAME_STATE FSM_1 Karnaugh Map, State s0

		Current Input (e1, e0) RESPONSE_RE, START_OPERATION or ENCODER_CLOCK_COMPLETE			
		0,0	**0,1**	**1,1** (1)	**1,0**
Previous State s1, s0	**0,0 IDLE**	0	1 (2)	1 (2)	0
	0,1 TRANSMIT_MA	1 (3)	1 (3)	0	0
	1,1 RECEIVE_SL	1 (4)	1 (4)	1 (4)	1 (4)
	1,0 TRANSMIT_MA RECEIVE_SL	0	1 (5)	1 (5)	0

(1) Corresponds to an invalid, or unexpected, e1:e0 combination. The system designer decides the behavior of the state machine in such cases.
(2) !s1 & !s0 & e0
(3) !s1 & s0 & !e1
(4) s1 & s0. The C28x forces the transition from state 1,1 back to IDLE.
(5) s1 & !s0 & e0

Table 2-6. FRAME_STATE FSM_1 Karnaugh Map, State s1

		Current Input (e1, e0) RESPONSE_RE, START_OPERATION or ENCODER_CLOCK_COMPLETE			
		0,0	**0,1**	**1,1**	**1,0**
Previous State s1, s0	**0,0 IDLE**	0	0	0	0
	0,1 TRANSMIT_MA	0	0	1 (1)	1 (1)
	1,1 RECEIVE_SL	1 (2)	1 (2)	1 (2)	1 (2)
	1,0 TRANSMIT_MA RECEIVE_SL	1 (3)	1 (3)	1 (3)	1 (3)

(1) !s1 & s0 & e1
(2) s1 & s0. The C28x forces the transition from state 1,1 back to IDLE.
(3) s1 & !s0

The OUT signal from FSM_1 corresponds to the IDLE state.

Detection of the encoder's response is another key component of the design. LUT_2, shown in Figure 2-13, is responsible for detecting the encoder's ACK. If the FRAME_STATE is WAIT_FOR_ACK (0,1) and ENCODER_RESPONSE_RE goes high, then ACK has been detected. This results in the equation: i0 & (i1 & !i2):

- i0 == 1: ENCODER_RESPONSE_RE goes high
- (!i2 & i1) == FRAME_STATE == WAIT_FOR_ACK (0,1)

2.3.5.3 CLB_SPI_CLOCK Generation

The CLB is responsible for detecting the encoder's response and then clocking the SPI to receive the data. First consider the duty and frequency of the clock. Figure 2-14 shows the CLB logic which generates the SPI clock. The corresponding simulation waveform is shown in Figure 2-15.

The first step is to generate a clock (called SPI_CLOCK) with the specificed frequency and duty. In the generation of SPI_CLOCK:

- TILE3_COUNTER_1: is responsible for the duty and frequency (width) of the clock. The time between the high transition and the low transition of the SPI_CLOCK is measured in the number of CLB clocks. The following match signals are used:
  - zero match: corresponds to the low transition of the clock
  - match1: corresponds to the high transition of the clock
  - match2: corresponds to the width of the clock
- TILE3_FSM_1: Generates the clock edges based on the COUNTER_1 match values: match1 and zero match
- TILE3_LUT_0: Resets the counter when the clock width has been reached as indicated by the COUNTER_1 match2 output

The second step is to align the clock with the received data. This is required for the SPI to receive the encoder's response correctly. There are two challenges to aligning the clock:

1. The response can arrive at any time due to cable propagation delay as described in Section 2.3.1.1.
2. The encoder can also delay the acknowledgment, ACK, to request additional processing time as described in Section 2.3.1.2.

To align the clock, LUT_0 monitors the RESPONSE signal for a rising edge. The first rising edge corresponds to the START bit after the acknowledge (ACK) bit. When this edge is detected, LUT_0 resets COUNTER_1 which aligns the SPI_CLOCK to the response.

The third step is to generate the CLB_SPI_CLOCK based on the internal SPI_CLOCK. As shown in Figure 2-15:

- SPI_CLOCK is generated for a longer duration then required for the SPI to receive the response
- CLB_SPI_CLOCK only outputs the number of clocks required to receive the response

This is achieved by using TILE4_FSM_2 to generate a SPI_CLOCK_OUTPUT_ENABLE signal. This signal is used by the OUTLUT to allow/block the SPI_CLOCK as required.

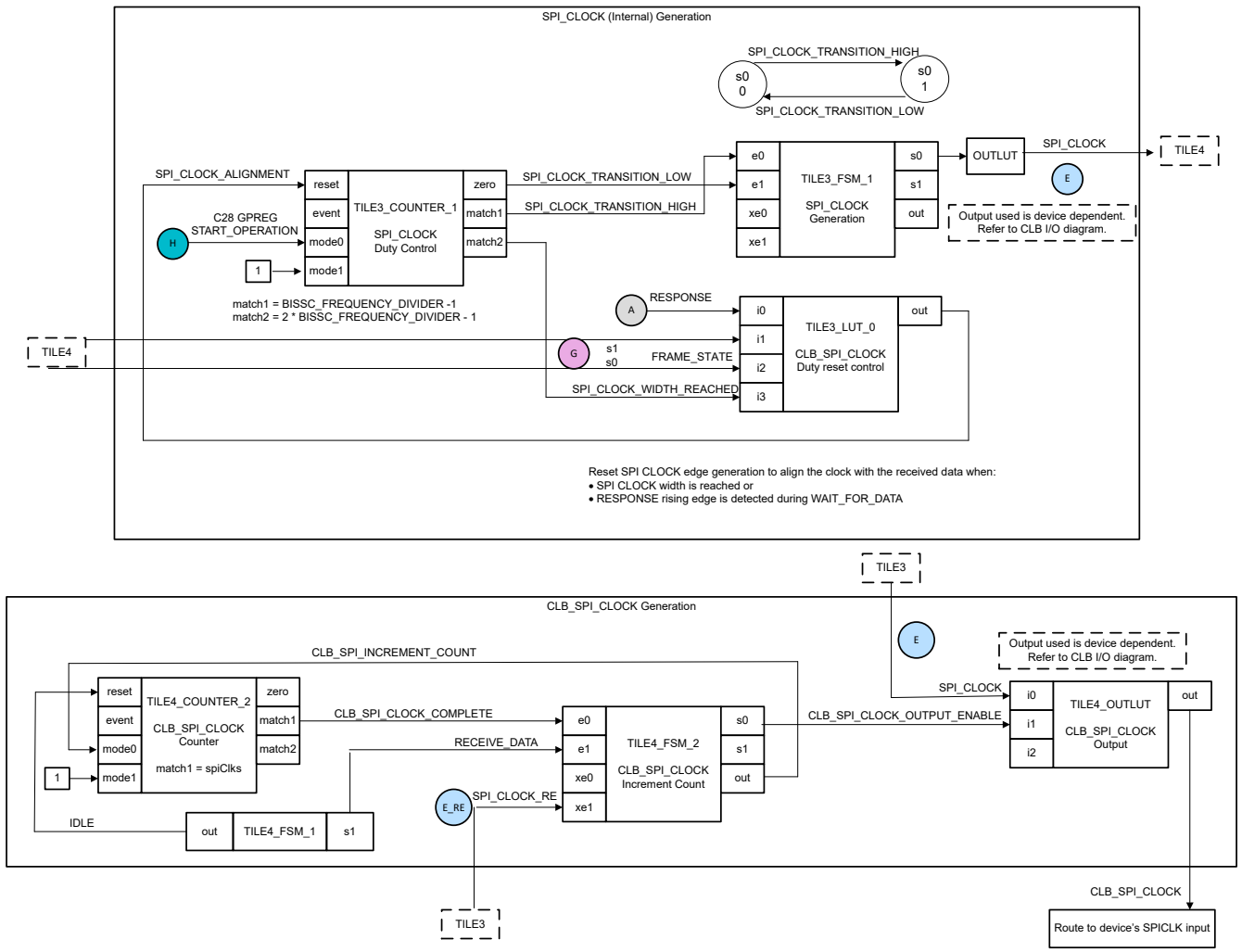


Figure 2-14. CLB_SPI_CLOCK Generation

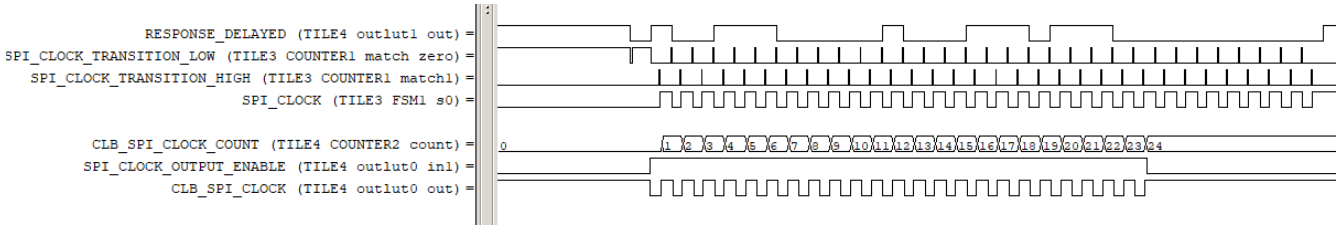


Figure 2-15. SPI Clock Generation Simulation Waveform

2.3.5.4 ENCODER_CLOCK (MA) Generation

The ENCODER_CLOCK, or BiSS MA, generation is similar to the CLB_SPI_CLOCK generation shown in the previous section. There are two key differences:

- The total number of MA clocks generated depend on when the response is received.
- The end of the MA signal must be held high or low to indicate the CDM bit.

The marker (1) in Figure 2-18 indicates the point when the encoder's ACK has been detected by the logic shown in Figure 2-17. Prior to marker (1) an unknown number of MA clocks has been generated. From marker (1) forward, the number of additional MA clocks required is X + 4 + 6 where:

- X bits: total POSITION bits (Single Turn + Multi Turn)
- 4 bits: START bit + CDM bit + ERROR bit + WARNING bit
- 6 bits: CRC bits

This number is stored by the application in TILE4 HLC register R0. HLC adjusts t the total number of clocks (COUNTER_1 match1) using this value along with the current counter value. For example, refer to Figure 2-18):

1. If X = 13, then the application stores 23 in HLC R0
2. When ACK is detected at marker (1), the HLC program is triggered.
3. HLC reads the current CLB_ENCODER_CLOCK_COUNT (4) and adds R0 (23) = 27
4. HLC loads 27 into TILE4 COUNT_1 match1
5. When the CLB_ENCODER_CLOCK_COUNT reaches match1, the generation of MA clocks is complete.

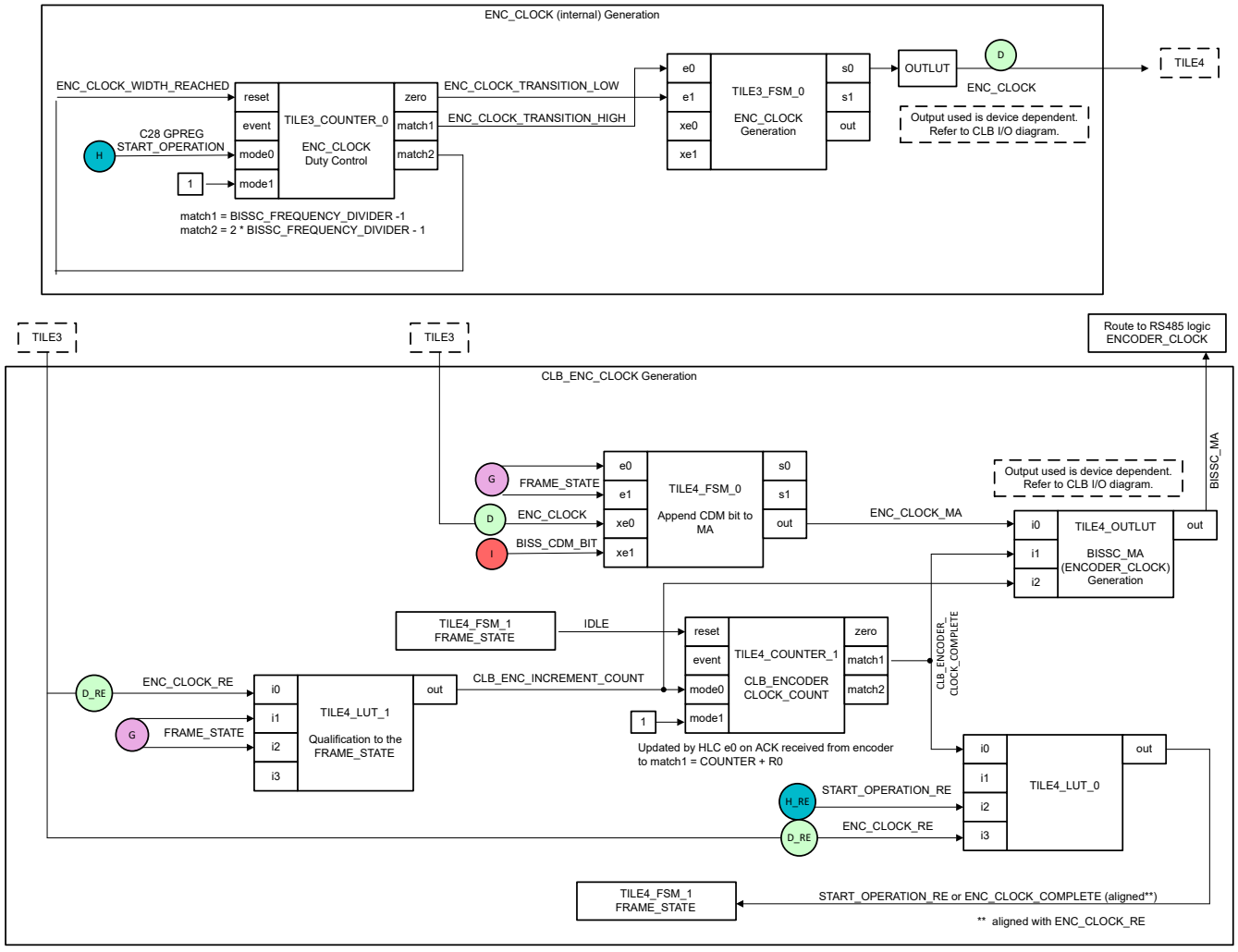


Figure 2-16. ENCODER_CLOCK (MA) Generation

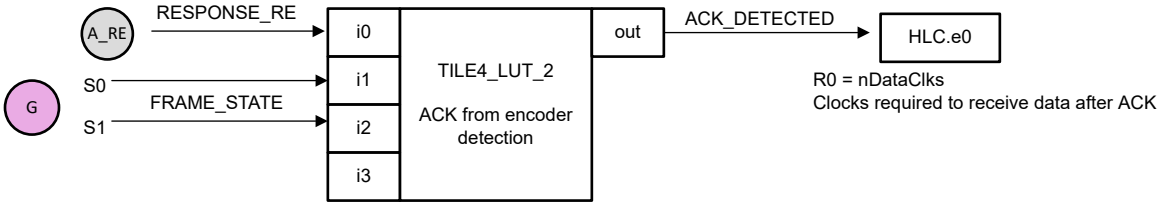


Figure 2-17. Encoder ACK Detection Logic

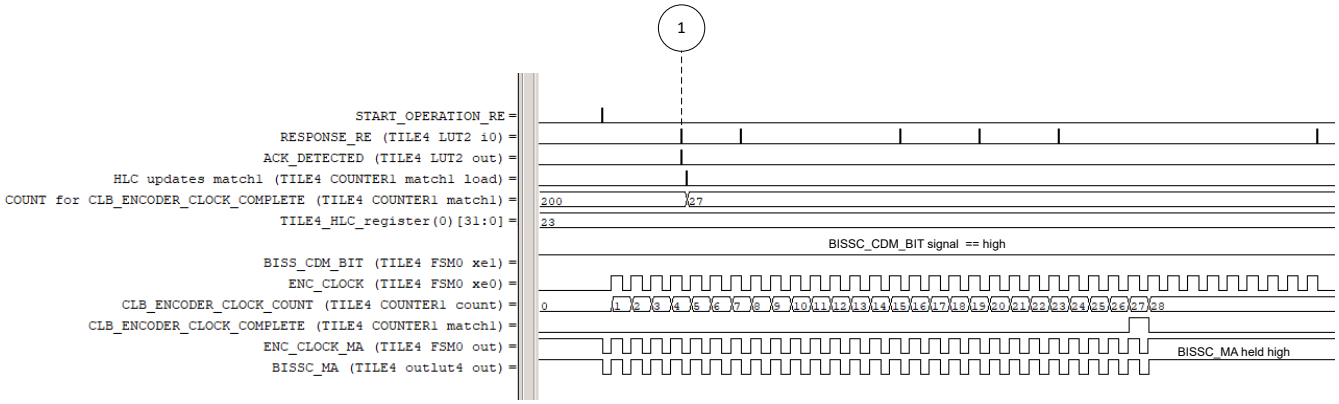


Figure 2-18. MA Generation Simulation Waveform: CDM Signal High

The CDM bit is one-bit of control data transferred to the encoder each BiSS-C frame. The end of the MA signal must be held high or low to indicate the current CDM bit. Figure 2-18 shows a simulation waveform when the level of BISSC_CDM_BIT is high. Figure 2-19 shows a simulation waveform for BISSC_CDM_BIT signal low. The BISS_CDM_BIT signal is controlled by the C28x CPU writing to the tile's GPREG.

Note

For this discussion the state of the signal, high or low, refers to the level of the BISS_CDM_BIT signal. The actual CDM bit value is interpreted by the encoder as the inverse of signal level. For example, if the BISS_CDM_BIT signal is high, the encoder detects the CDM bit value as 0. The software library accounts for this inversion.

Tile 4 FSM_0 appends the BISS_CDM_BIT level to the MA signal when FRAME_STATE changes due to CLB_ENCODER_CLOCK_COMPLETE (Figure 2-16). In the case where the level of BISSC_CDM_BIT is low, an extra edge is removed from ENC_CLOCK_MA by the output (Figure 2-19)

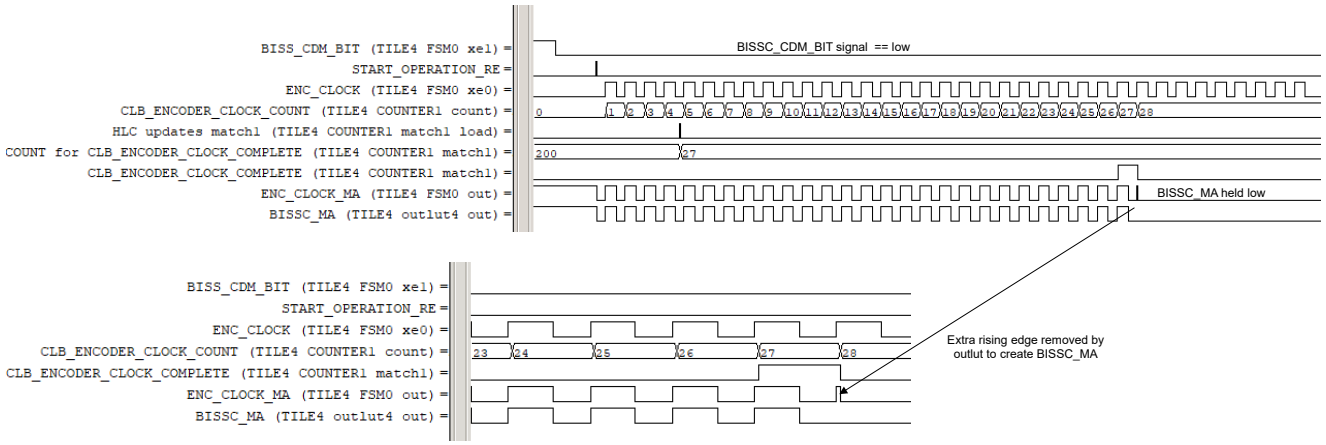


Figure 2-19. MA Generation Simulation Waveform: CDM Signal Low

2.3.6 PM BiSS-C Interface Library

The PM BiSS-C encoder interface library provides:

- The CLB logic implementation.
- A well-defined application programming interface (API) to enable C2000 devices to communicate with BiSS-C position encoders.

This section provides a high-level overview of the functions in the API.

2.3.6.1 PM BiSS-C Library Functions

The BiSS-C Library consists of the following functions, which enable the user to interface with BiSS-C encoders. For a detailed description of the API, refer to the: *C2000 BiSS-C Encoder Interface Software Guide* (HTML, PDF).Table 2-7 provides a high-level overview of the API.

Table 2-7. BiSS-C Library Functions

NAME	DESCRIPTION
Initialization Functions	
Generate CRC Table	Generates a lookup table for a given CRC polynomial with a specified number of bits. Two tables are used. One for the single-cycle data CRC calculation and one for the command data CRC calculation.
Setup Peripheral	Configures the BiSS-C sub-system realized by the CLB logic.
Initialize Parameters	Initialize parameters required to extract the position and CRC from the single-cycle data.
Run-time Functions	
Start Operation	Initiates a BiSS-C transfer by starting the MA signal.
Process Command Data	Command data state machine which handles the CDM output and CDS input each BiSS-C frame.
Receive Position	Extract the position, error and warning and CRC from the SCD. Calculates a CRC and compares it to the received CRC.
Setup a SCD Transaction	Set up the SPI and CLB for a new single-cycle data transfer.

3 Hardware, Software, Testing Requirements, and Test Results

This section details the test procedure and results. Included are:

- Hardware requirements
- Software requirements
- Test results
- Benchmarks
- Troubleshooting guide

3.1 Hardware

To experiment with the TIDM-1010, the following hardware components are required:

- TIDM-1010 BoosterPack (also known as the BOOSTXL-POSMGR)
- External, 5-V, DC power supply (see Table 1-1)
- F28P65x LaunchPad development kit (LAUNCHXL-F28P65X) and USB cable
- BiSS-C Encoder
- Cable to connect the encoder to TIDM-1010
- Custom adapter to connect the female-terminated cable to the wire-leads adapter
- Computer with CCS installed

3.1.1 TIDM-1010 Jumper Configuration

Figure 3-1 shows the jumper configuration for the TIDM-1010/BOOSTXL-POSMGR board.

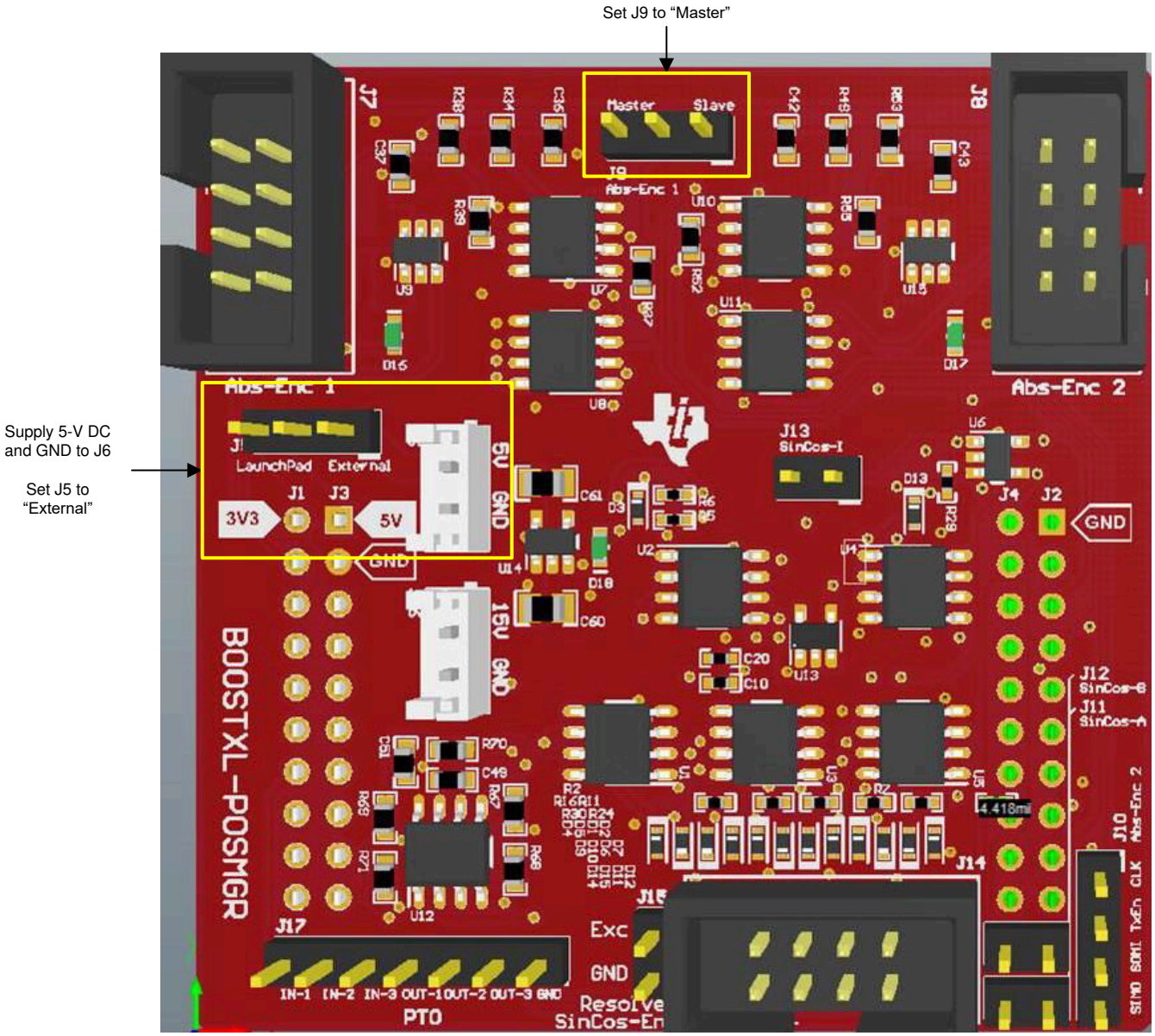


Figure 3-1. TIDM-1010 Board Jumper Configuration

Table 3-1 lists the jumper configuration for the TIDM-1010 board.

Table 3-1. TIDM-1010 Board Jumper Details

JUMPER	FUNCTION	POSITION
J5	TIDM-1010, 5-V, power-plane source selection	*External* (1)
J9	Abs-Enc-1 master-slave mode selection	*Master* (2)
J11	Sine-Cosine encoder-A signal enable	Open
J12	Sine-Cosine encoder-B signal enable	Open
J13	Sine-Cosine encoder-index signal enable	Open

(1) This configuration requires users provide an external power source to J6, as shown in Figure 3-1.
(2) Do not use the slave mode option. There is an error in the BoosterPack logic for this mode.

3.2 Software

This section provides an overview of the software used by TIDM-1010. For comprehensive documentation, refer to the *C2000 BiSS-C Encoder Interface Software Guide* (HTML, PDF).

The software guide includes:

- Documentation of the system demonstration code
- Documentation of the BiSS-C application programmer interface (API)
- Incorporating the library into your own solution
- Guidance for porting the solution from C28x CPU1 to CPU2 on dual-CPU devices.
- Software change history

Figure 3-2shows the software architecture implemented in this reference design. The software is implemented in a modular and portable manner. The main components include the C2000 Driver Library, the BiSS-C Encoder Interface Library, the SysConfig GUI Device Configuration Tool, and the CLB Tool.

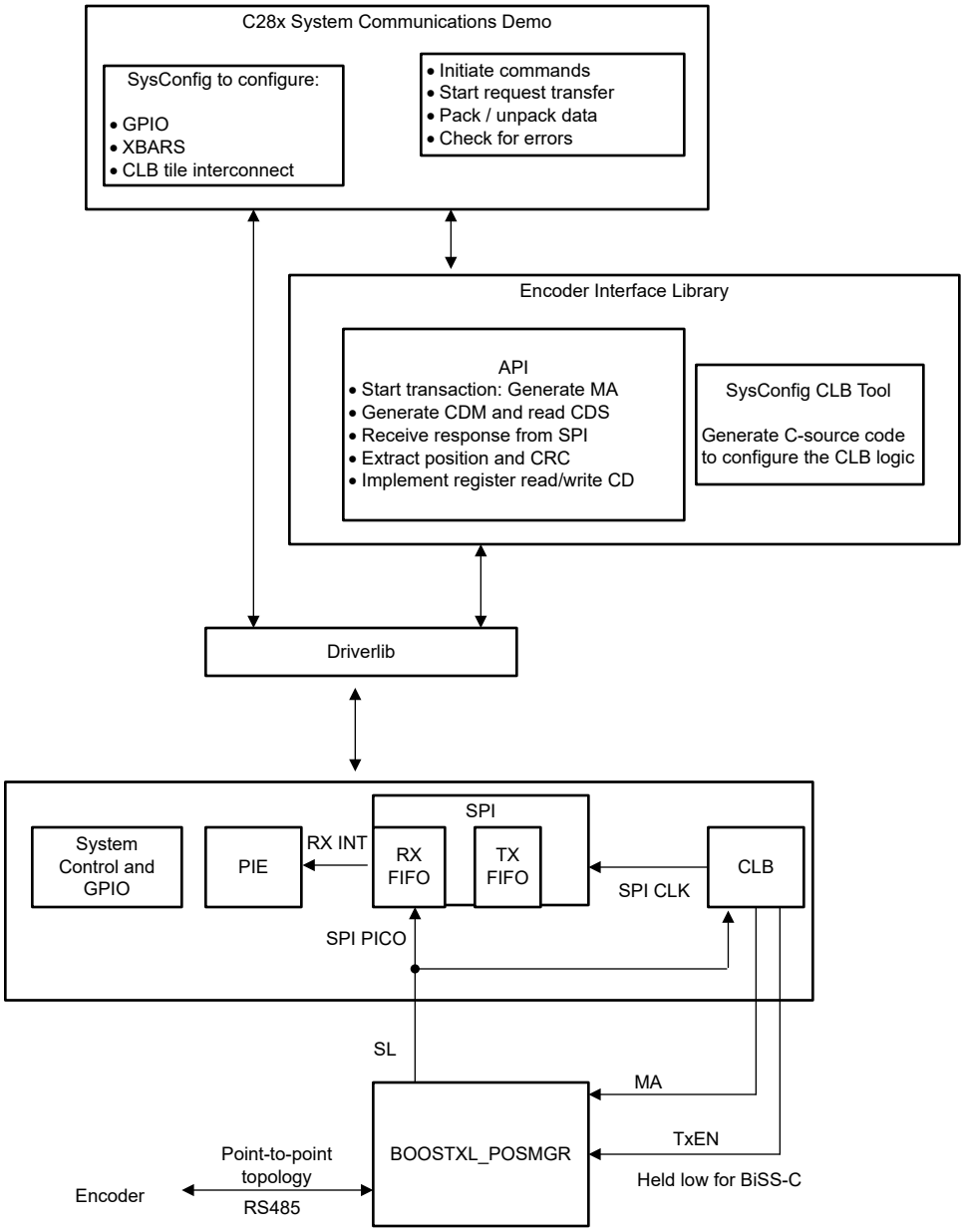


Figure 3-2. Software Architecture

A flowchart for the C2000 BiSS-C communications demonstration is shown in Figure 3-3. The example application configures the C2000 device, creates control data frames, sends the MA signal, unpacks responses, and checks the CRC.

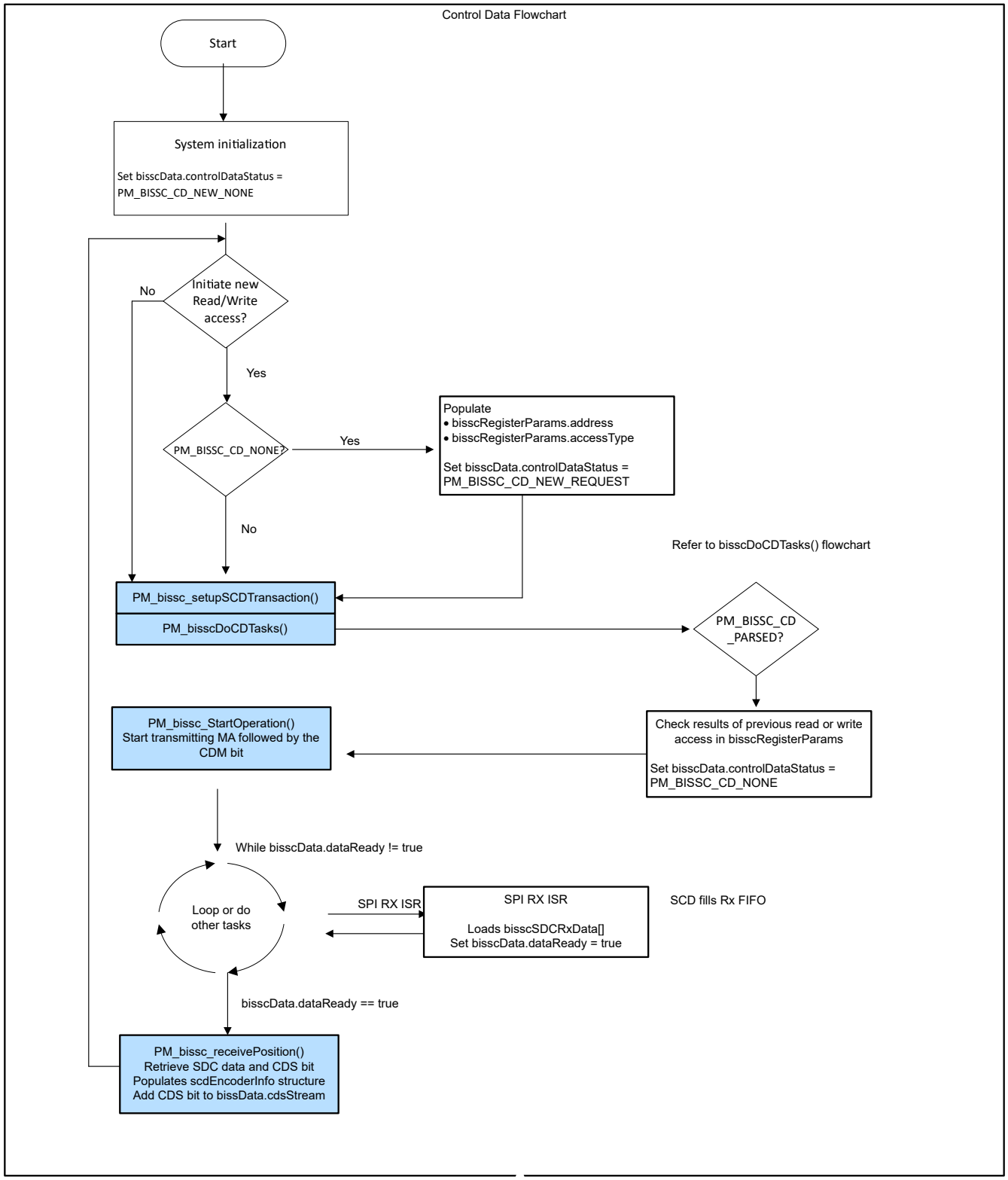


Figure 3-3. Communication Demonstration Flowchart

3.2.1 C2000 Driver Library (DriverLib)

The C2000 Driver Library (Driverlib) is a set of low-level APIs for C2000 device families. Driverlib provides easy-to- use function calls for configuring memory-mapped peripheral registers. Full source for Driverlib is

provided within C2000Ware and the C2000Ware Motor Control SDK. For more information, refer to the DriverLib section of the C2000 Software Guide.

3.2.2 C2000 SysConfig

C2000 SysConfig is a graphical user interface tool for configuring the C2000 Real-Time Control MCUs. SysConfig auto-generates embedded software that interfaces to Driverlib. In this reference design, the SysConfig tool is used to generate code to configure the SPI, GPIO, INPUTXBAR/OUTPUTXBAR and CLB MUX. For more information, refer to the C28x Academy: SysConfig module.

3.2.3 C2000 Configurable Logic Block Tool

The C2000 CLB Tool enables configuration of the CLB Logic through a graphical interface. The CLB Tool is an easy-to use GUI built into Code Composer studio and makes use of the C2000 SysConfig plug-in. In this reference design the CLB Tool is used to configure the tiles for the BiSS-C Encoder Interface as described in the design description. For more information, refer to the C28x Academy: Configurable Logic Block module.

3.2.4 Installing Code Composer Studio™ and C2000WARE-MOTORCONTROL-SDK

1. Install CCSTUDIO IDE v12.7.1 or later, if CCSTUDIO is not already on the computer.
2. Install C2000WARE-MOTORCONTROL-SDK v5.03.00.00 or later, if the software is not already installed on the computer
3. After installation, refer to the C2000 BiSS-C Encoder Interface Software Guide (HTML, PDF) for further instructions.

Note

The CLB tool is included Code Composer Studio (sysconfig) and the C2000Ware sub-component of the SDK (support utilities). To run CLB-based simulations requires installation of additional tools which are documented in the CLB Tool User's Guide.

3.2.5 Locating the Reference Software

The software included in this reference design consists of two parts:

- A system example that illustrates the usage of the encoder interface. The location of the example project source files is shown in Table 3-2.
- The encoder interface library. The location of the library source files is shown in Table 3-3.

For comprehensive documentation, refer to the *C2000 BiSS-C Encoder Interface Software Guide* (HTML, PDF).

Table 3-2. Location of System Examples

`C:\ti\c2000\C2000Ware_MotorControl_SDK_[version]\`	Default install location for the SDK. (`[SDK]`)
`[SDK]\solutions\boostxl_posmgr\`	Device-specific base install directory (`[pm_base]`)
`[pm_base]\[device]\ccs\bissc`	Code Composer Studio (CCS) projectspec files. Used to import the project into your CCS workspace.

Table 3-3. Location of the Encoder Interface Library

`C:\ti\c2000\C2000Ware_MotorControl_SDK_[version]`	Default install location for the SDK. (`[SDK]`)
`[SDK]\libraries\position_sensing\bissc`	Library base install directory (`[lib_base]`)
`[lib_base]\ccs\[device]`	Code Composer projectspec file for the reference library. Use these projects to re-build the library for each device.

3.3 Testing and Results

This section details the test procedure, results, and benchmarks. A troubleshooting guide is also provided.

3.3.1 Hardware Configuration

1. Make sure that the jumper configuration of the TIDM-1010 device is as described in Table 3-1.
2. Connect the TIDM-1010 device to the LaunchPad using the BoosterPack site-two connector (J1, J3 and J4, J2). Make sure the TIDM-1010 device is connected to site two of the LaunchPad, as shown in Figure 3-4.

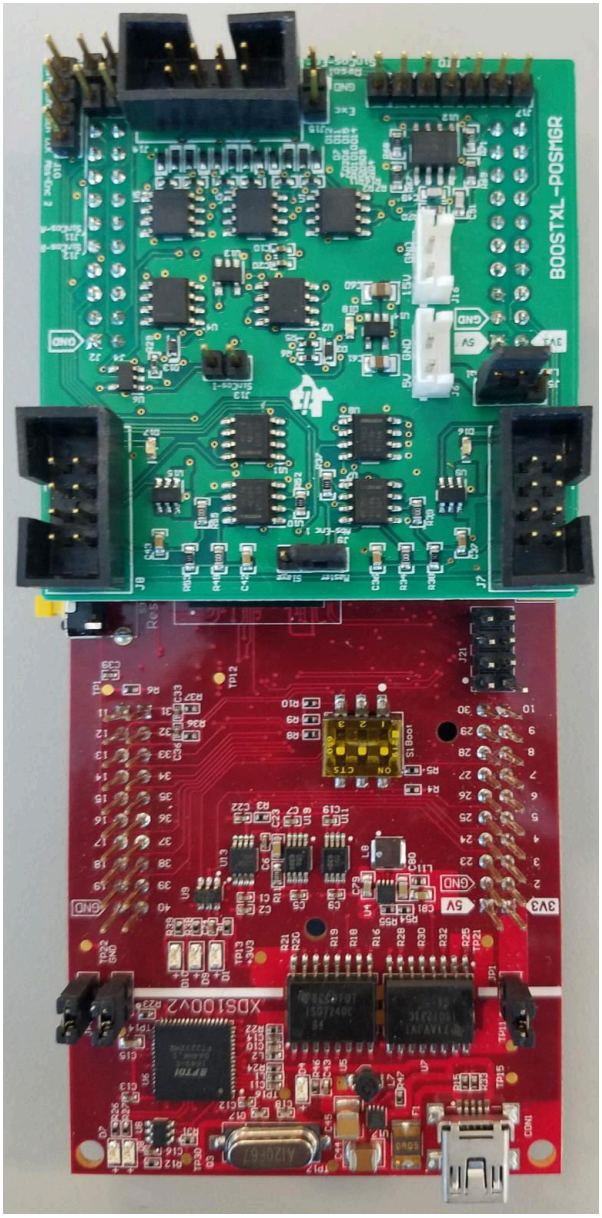


Figure 3-4. TIDM-1010 Board Connected to Site Two of LaunchPad™

3. Connect the USB cable to the LaunchPad.
4. Connect to the encoder:
   a. Prepare an adapter to connect the cable to the BiSS-C interface using the circular female to wire the leads adapter (see the BOM for the header used for the encoder connector, J7).
   b. Connect CLK+/CLK- to the BiSS-C MA+/MA- signal of the encoder.
   c. Connect DATA+/DATA- to the encoder's response SL+/SL-.
   d. Insert the header of the adapter created in the previous step to connect to Abs-Enc-1 (J7). The female end of the cable connects to the encoder. Figure 3-5 shows the pinout of J7.

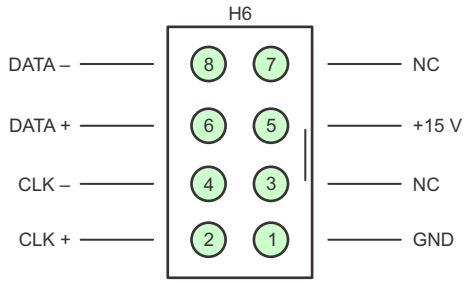


Figure 3-5. Abs-Enc-1 (J7) Pinout on TIDM-1010 Board

5. Supply 5-V DC and GND to J6, as shown in Figure 3-1. The board now looks like Figure 3-6. BoosterPack LED D18 turns on, which shows that the board has power.

Note

For some encoders the BoosterPack may not provide an adequate current at power up. If the encoder fails to respond, try connecting a power supply external to the BoosterPack to the encoder. If this is done, connect a common ground to the BoosterPack.

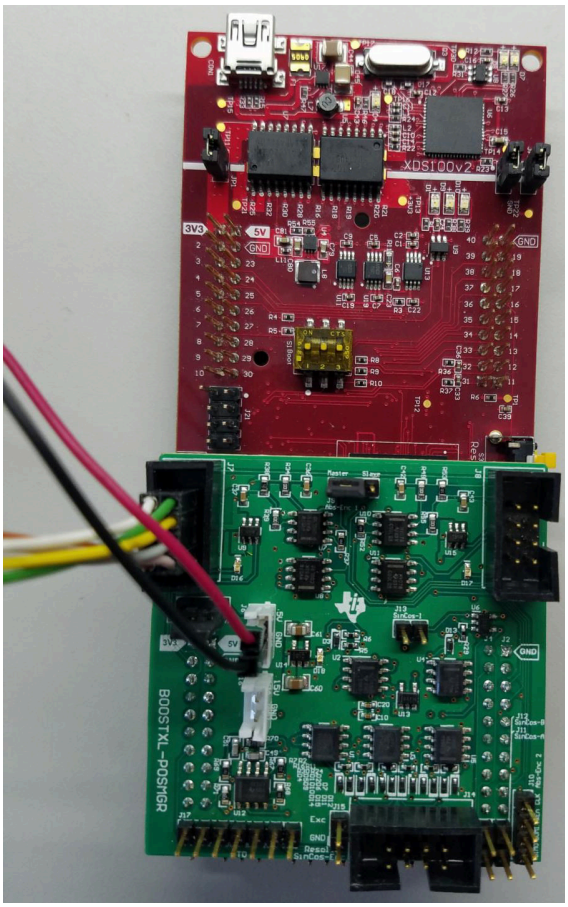


Figure 3-6. TIDM-1010 Board Connected to BiSS-C Encoder

3.3.2 Building and Loading Project

Follow the instructions in the C2000 BiSS-C Encoder Interface Software Guide (HTML, PDF) to load and run the system solution. Refer to the System Solution section of the software guide.

These directions include:

- Importing the projects into Code Composer Studio (CCS) for your device
- Configuring the library and system example
- Selecting the build configuration
- Populating the watch (expressions) window
- Running the code

3.3.3 Running Example Code

The BiSS-C system solution is a communications-only demonstration. The demo sends a MA signal to the encoder, receives the response, and checks for errors. This pattern is repeated in a while(1){} loop. In addition, the demo periodically sends a read/write access to an encoder register. As provided, the bank select register is used. While running the demo, you can monitor output signals of the MCU with a logic analyzer or scope (Figure 3-7) while manually turning the motor, or encoder's shaft.

Note

Only the F2837xD requires an external connection between the CLB and the SPI clock. Other devices have an internal connection between the CLB and the SPICLK. For devices with an internal connection, the SPICLK can also be brought out to a pin for monitoring. The test connection for SPI CLK is shown in the device input/output diagrams in Section 2.3.4.1.

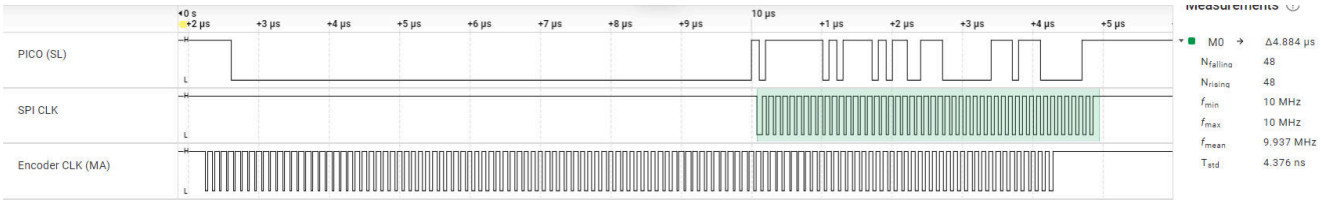


Figure 3-7. BiSS-C Waveform

The waveform shown in Figure 3-7 is:

- For an encoder with 32 position bits
- 10-m cable with 10 MHz MA signal
- The SPI clock was brought out to a pin for monitoring
- The SPI FIFO is configured for a 12-bit word with an interrupt when level 4 is full
- 48 clocks required to assert an interrupt
- Captured by a logic analyzer with a sampling frequency of 100 MHz

3.3.4 Encoder Test

Table 3-4 lists tests with various types of encoders. These tests were performed at Texas Instruments. The tests include reading position, reading registers, writing to registers. A 10 MHz operation is possible with up to 10-m in cable length. Longer cable lengths have also been tested at reduced frequency.

The maximum MA frequency depends on both the encoder and the encoder cable. The quality of the cable has an impact on the encoder's digital communication performance and achievable reach. For example, the power wires in the encoder cable may have a thicker gague to minimize the voltage drop for long cables. Typically, the encoder vendor will recommend the corresponding cable for their encoder. This cable quality is critical to work at 100-m or more cable lengths.

For guidelines for MA clock Frequencies vs Cable length, refer to the document *BiSS Interface: AN15: BiSS C MASTER OPERATION DETAILS* available from the BiSS Interface website.

Table 3-4. Encoder Test Summary

ENCODER MANUFACTURER	ENCODER NAME	TYPE	RESOLUTION (BITS)
Lika	HS58S18	Rotary	18 bits
Lika	EH036-20-12	Rotary	32 bits (12 MT + 20 ST)
iCHaus	EVAL MHM1D	Evaluation board	28 bits (16 MT + 12 ST)

3.3.5 Benchmarks

Table 3-5 lists the C28x CPU cycles required to execute BiSS-C library functions from RAM. This data was collected using the ERAD module and the following compiler settings:

- C2000 Codegen Tools V22.6.1.LTS
- -O2 -mf2
- float_support: fpu32
- tmu_support: tmu0
- fp_mode: relaxed
- abi: eabi

Note

These functions can be further optimized depending on the application. For example, if the resolution of the encoder is known at compile time, instead of run time, some code can be eliminated.

Table 3-5. Cycle-Count Benchmarks

Function	Cycles -O2 -mf2	Notes
PM_bissc_setupSCDTransaction	369	
bissc_setupCDTest	23	
PM_bissc_receivePosition	552	Depends on the resolution of the encoder. An encoder with 12 multiturn and 20 single turn bits was used for this measurement.
PM_bissc_doCDTasks	39 (min), 225 (max)	The maximum cycles occur when a register read or register write has completed.
PM_bissc_startOperation	39	
PM_bissc_setFreq	73	
bissc_getCRC	90	
bissc_spiRxISR	497	

Table 3-6 lists the code-size, in 16-bit words, corresponding to each of the library source files. The C28x lookup table takes 256 words of RAM or Flash that is not reflected in this table.

Table 3-6. Code-size in 16-bit Words

Source File	Code Size -O2 --mf2
pm_bissc_source	910
pm_bissc_crc	49
clb_config	463

3.3.6 Troubleshooting

Examine the following waveforms can assist in troubleshooting. Refer to the I/O diagrams in the design description.

- The CLB generated SPI clock. This is brought out on a test pin for observation.
- The response from the encoder at the SPI input pin.
- The response from the encoder (SL+/SL-) between the RS485 line driver and the encoder. Note: The data is a differential signal. Therefore, observation requires a special probe.
- The TxEN signal. Confirm this singal remains low. BiSS-C design does not pull TxEN high.
- The encoder clock (MA+/MA-) between the RS485 line driver and the encoder. Note: The data is a differential signal. Therefore, observation requires a special probe.

1. If the MA signal is not transmitted:
   - Determine if the issue is before, or after, the RS485 line drivers.
   - Confirm that TxEN is held low if the issue is between the line drivers and the encoder.
2. If the MA frequency is not as expected:
   - Check the BISSC_MA_CLOCK definition and BISSC_FREQ_DIVIDER definition in bissc.h
3. If the encoder response is not seen:
   - Check the power connections to the encoder.
   - Check the power supply current against the encoder specifications.
   - Reduce the frequency of MA and try again. This can indicate a cable, or connector, issue.
   - Confirm that the cable design, and length, meets the requirements of the encoder manufacture.
   - Determine if the issue is before or after the RS485 line drivers.
4. If the response waveform is observed, but not captured by the SPI:
   - This can occur if the CLB design was moved to a different tile or ported to a different device.
     – Check the SPI clock from the CLB. This clock is started when the CLB sees the response from the encoder. Confirm the internal XBAR connections properly route the response to both the SPI and the appropriate CLB input.
     – If the SPI clock is seen, confirm that the internal connection from the CLB output goes to the SPI instance used. This changes depending on the CLB tile used.

4 Design Files

To download the design files, see the product page at TIDM-1010.

5 Software Files

The software source files are included in the MotorControl software development kit (SDK) for C2000™ MCUs.

6 Related Documentation

1. iCHaus. iC-Haus, website.
2. BiSS Interface BiSS Interface, website.
3. C2000 BiSS-C Encoder Interface Software Guide (HTML, PDF)
4. Wikipedia, Cyclic Redundancy Checik, page.
5. Texas Instruments: Code Composer Studio page.
6. Texas Instruments: C28x Academy delivers easy-to-use training modules that span a wide range of topics for all C28x devices.
7. Texas Instruments: C2000 Software Guide includes an overview of C2000 software, software development kits and development tools.
8. Texas Instruments: C2000 DesignDRIVE, software for Industrial Drives and Motor Control
9. Texas Instruments: *C2000 Position Manager SinCos Library*, user's guide

Trademarks

7 Terminology

C28x	Refers to devices with the C28x CPU core.
API	Application Programming Interface. The definition of the library which enables the encoder interface protocol.
BiSS	An open-source digital interface for sensors and actuators. In this document, BiSS Interface or BiSS refer to the BiSS-C protocol introduced in 2007. BiSS-C stands for Bidirectional Serial Synchronous Continuous mode. For more details on the protocol and implementation aspects, see the *BiSS Interface* website.
BiSS frame	Single-cycle data plus one bit of control-data. The BiSS frame is sent as a response every BiSS cycle.
BiSS license	The purchase of TI MCUs does not automatically include a license to use the BiSS IPs. Users of the BiSS Interface have to apply for their own license from BiSS Interface.
CD	From the BiSS-C specification. Control Data. Data transmitted one bit per BiSS frame. For example, the CD permits the reading and writing of encoder registers.
CDM	From the BiSS-C specification. Control Data Master. One bit of control data, CD, sent each Biss frame by the controller to the encoder.
CDS	From the BiSS-C specification. Control Data Slave. One bit of control data, CD, sent each BiSS frame by the encoder back to the controller.
CLB	Configurable Logic Block peripheral on C2000 devices.
CRC	Cyclic redundancy check
CTS	From the BiSS-C specification. ConTrol Select bit. A BiSS-C control data (CD) frame begins with a start bit followed by the CTS bit. If CTS is 0, the control frame is a command frame. If CTS is 1, then the control frame is a register access frame.
MA	From the BiSS-C specification. Master clock. This pulse train output is sent from the controller to the encoder/sensor. It consists of the encoder clock plus one bit of control data from the master.

Position Manager BoosterPack	Future EVM for interfacing with various position encoders. The TIDM-1010 board is identical to the Position Manager BoosterPack EVM (see Section 2.3.3)
PM	Position Manager – foundational hardware and software on C2000 devices for position encoder interfaces
PM_bissc or PM_BISSC	Prefix used for all the library functions.
Point-to-point Topology	A configuration in which only one device (encoder or sensor) is interfaced to the master.
SSI	Serial Synchronous Interface
SL	From the BiSS-C specification. Slave return. Data input to the BiSS-C master from the slave. This is the encoder response including position and one bit of CD data each BiSS frame.
SPI	Serial peripheral interface peripheral on C2000 devices.

8 About the Authors

LORI HEUSTESS has been a member of the C2000 team for many years. Her areas of interest have been in CPU and peripheral validation, software development, and industrial applications. Lori currently works on the Application Specific MCU (ASM) Industrial Applications team.

SUBRAHMANYA BHARATHI AKONDY worked on the architecture definition and design of several C2000 MCU products and control peripherals. His interests include MCU architecture, applications, and design aspects.

SHEENA PATEL works on the Industrial Drives team of the C2000 MCU group as a Product Marketing Engineer.

9 Revision History

Changes from Revision * (April 2018) to Revision A (November 2024) **Page**

- Updated the numbering format for tables, figures, and cross-references throughout the document 1
- Updated launchpad information..... 1
- Added link to BiSS Interface website. Added library high-level features..... 2
- Added introduction to *System Overview* section..... 4
- Updated block diagram..... 4
- Added introduction to *Highlighted Products* section..... 4
- Added hyperlinks to the devices in the *Highlighted Products* section..... 4
- Changed MCU and LaunchPad information to the F28P65x..... 4
- Added introduction to *Design Considerations* section..... 5
- Added *BiSS-C Protocol* section..... 5
- Added *Line Delay Compensation* section..... 7
- Added *Processing Time Request by Encoder* section..... 7
- Added *Control Communication* section..... 8
- Added *C2000 BiSS-C Encoder Interface Overview* section..... 9
- Added functionality overview. Updated table column 3 *TIDM-1010 Board and BOOSTXL-POSMGR Connectors* indicating usage specifically by TIDM-1010. Added *BoosterPack Block Diagram* 10
- Added *MCU Resource Requirements* section..... 13
- Added *Input, Output Signals, and CLB Tiles* section..... 13
- Added *CLB BiSS-C Implementation Details* section..... 15
- Added *Transaction Waveforms* section..... 15
- Added *FRAME_STATE Generation* section..... 17
- Added *CLB_SPI_CLOCK Generation* section..... 19
- Added *ENCODER_CLOCK (MA) Generation* section..... 21
- Added introduction to *PM BiSS-C Interface Library* section..... 23
- Deleted information detailed in the *C2000 BiSS-C Encoder Interface Software Guide*. Added software guide hyperlink. Added summary of the library functions..... 24
- Added introduction to *Hardware* section..... 25
- Updated the *TIDM-1010 Board Jumper Details* table (note 2 and deleted unused note 3)..... 26
- Added summary and hyperlink to *C2000 BiSS-C Encoder Interface Software Guide*. Added *Software Architecture* and *Communication Demonstration Flowchart* figures. 27
- Added *C2000 Driver Library (DriverLib)* section..... 29
- Added *C2000 SysConfig* section..... 30
- Added *C2000 Configurable Logic Block Tool* section..... 30
- Changed *Installing CCS and controlSUITE*section with *Installing Code Composer Studio™ and C2000WARE-MOTORCONTROL-SDK* section..... 30
- Added *Locating the Reference Software* section..... 30
- Deleted instructions in the *C2000 BiSS-C Encoder Interface Software Guide*. Added overview of the software guide..... 33
- Changed instructions in the *C2000 BiSS-C Encoder Interface Software Guide* with a summary of the example application..... 33
- Updated table data for encoders tested. Updated the cable length recommendations..... 33
- Added *Benchmarks* section..... 34
- Added *Troubleshooting* section..... 34
- Updated hyperlink..... 36
- Added hyperlinks to iCHaus, BiSS Interface, Code Composer Studio, C28x Academy, and the C2000 Software Guide. Updated hyperlink to the BiSS-C Software Guide..... 36
- Added additional definitions for API, BiSS frame, BiSS license, CD, CDM, CDS, CTS, MA, SLA, SL, and more..... 36
- Added Lori Heustess to the list of authors..... 37

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265