C2000™ Position Manager T-Format Library Module User's Guide

User's Guide

Literature Number: SPRUI71
September 2016

Contents

List of Figures

List of Tables

C2000™ Position Manager T-Format Library Module User's Guide

1 Introduction

1.1 T-Format Interface

T-Format, from Tamagawa, is designed for serial transfer of digital data between linear, rotary or angle encoders, touch probes, accelerometers and the subsequent electronics, such as numerical controls, servo amplifiers and programmable logic controllers. For more details on the protocol and implementation aspects, see the Tamagawa website.

1.2 Abbreviations/Acronyms

Table 1. Abbreviations/Acronyms

Type	Description
C28x	Refers to devices with the C28x CPU core
CLB	Configurable Logic Block
CRC	Cyclic Redundancy Check
PM	Position Manager – Foundation hardware and software on C28x devices for position encoder interfaces
PM_tformat	Prefix used for all the library functions
SPI	Serial Peripheral Interface
Subsequent Electronics	tformat Master implementation
tformat	tformat specification of encoder interface protocol by Tamagawa

1.3 System Description

Industrial drives (like servo drives), require accurate, highly reliable, and low-latency position feedback. A simplified system block diagram of a servo drive using an absolute position encoder is shown in Figure 1. The tformat encoder interface from Tamagawa is a digital, bidirectional interface standard for position or rotary encoders. The interface transmits position values or additional physical quantities. It also allows reading and writing of the encoder's internal memory. The type of data transmitted like absolute position, turns, temperature, parameters, diagnostics, and so on is selected through mode commands that the subsequent electronics sends to the encoder. The interface is a pure serial digital interface based on RS-485 standard.

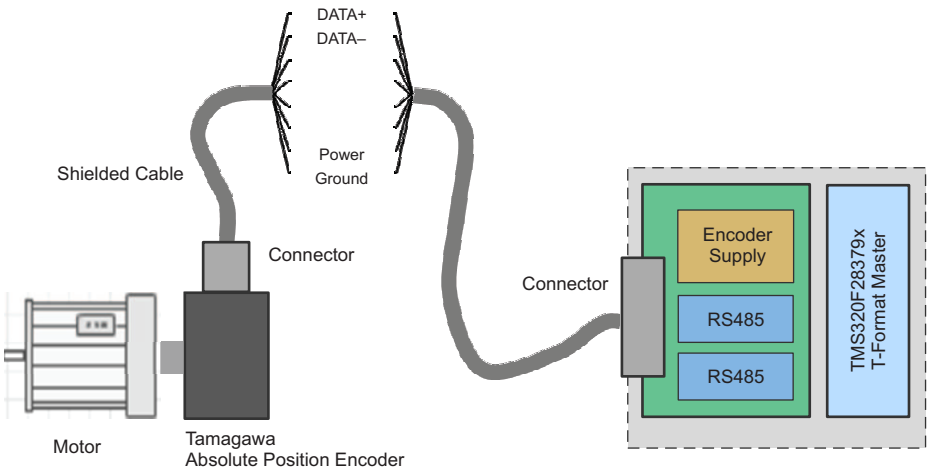

Figure 1. Industrial Servo Drive With T-Format Position Encoder Interface

Figure 1 shows the position encoder connected to the subsequent electronics through a shielded cable. Below are details of each of the eight wires used for communication:

- Two wires for DATA+/DATA- transmitted in differential format
- Two wires are used for the encoder power supply and ground

1.4 C2000 tformar Master Solution

The Texas Instruments C2000 position manager tformat (PM_tformat) library is intended to provide support for implementing the tformat interface in subsequent electronics.

Features offered by the tformat library:

- Integrated MCU solution for tformat interface
- Meets interface protocol requirements
- Verified operation up to 100m cable length
- Easy interface to tformat communication commands through driver functions and data structure provided by the library.
- Efficient and optimized CRC algorithm for both position and data CRC calculations
- Unpacking the received data and reversing the position data incorporated in library functions.
- Solution tuned for position control applications, where position information is obtained from encoders every control cycle, with better control of modular functions and timing.

Key things to note while using the tformat library:

- This library supports only the basic interface drivers for commands defined in the tformat encoder specification. All the higher level application software needs to be developed by utilizing the basic interface provided by this library.
- The functionality verified using this library, and this alone, is supported, see Section 5. For any additional functionality or encoder usage not specified in this section, contact your TI support team.

1.5 tformat Master Implementation Details

This section gives a brief overview of how the tformat interface is implemented on TMS320F28379D devices. Communication over the encoder interface is achieved primarily by the following components:

- CPU
- Configurable Logic Block (CLB)
- Serial Peripheral Interface (SPI)
- Device Interconnect (XBARs)

While SPI performs the encoder data transmit and receive functions; clock generation is controlled by CLB. The following functions are implemented inside the CLB module. Note that the CLB module can only be accessed via library functions provided in the PM tfomat Library and not otherwise configurable by users.

- Ability to generate clock to the serial peripheral interface on chip: on GPIO7 and looped back to SPICLK input
- Identification of the critical delay between the clock edges sent to the encoder and the received data
- Ability to adjust the clock delay
- Monitoring the data coming from encoder, via SPI Save In Master Out (SPISIMO), and poll for start pulse
- Ability to measure the propagation delay at a specific interval as needed by the interface
- Ability to configure the block and adjust delay the via software

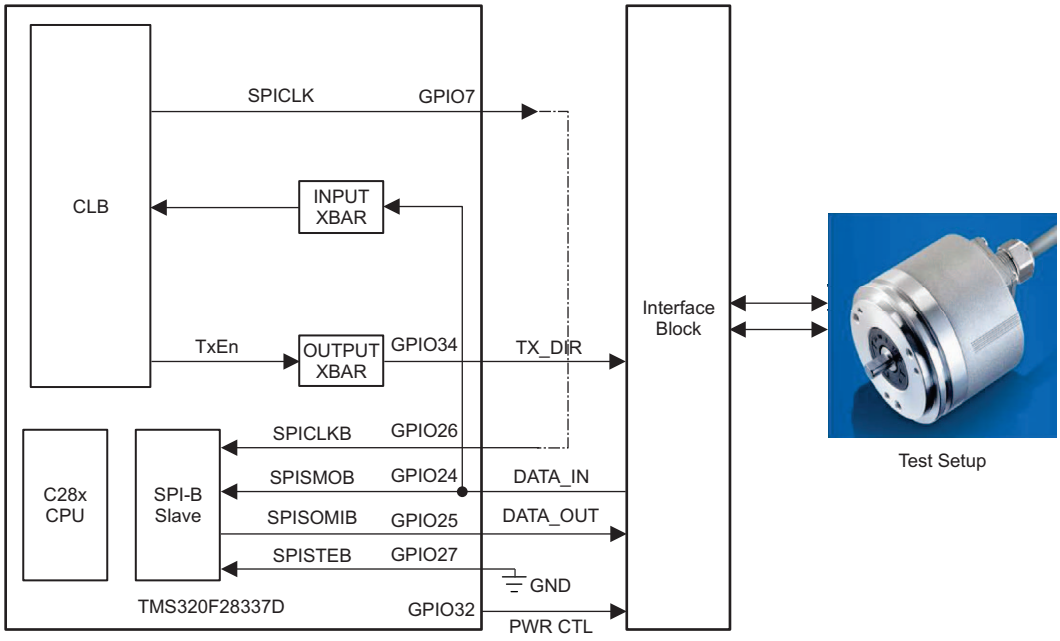

Figure 2. Implementation Diagram Inside TMS320F28379D

GPIOs indicated in Figure 2 are as implemented on TMDXIDDK379D.

Figure 2 depicts how the tformat transaction works in the system. For every transaction initiated using the Library command:

- CPU configures the SPITXFIFO with the command and other data required for transmission to the encoder as per the specific requirements of the current command.
- CPU sets up the configurable logic block to generate clocks for the SPI.
- CLB also generates the direction control signal for data line transceiver. This signal is needed to change the direction of the data line, after sending the mode command from subsequent electronics, to receive the data from the encoder.
- CPU configures CLB to generate a predefined number of clock pulses needed for the SPI (as per the current command requirements).

More details on various library functions provided and their usage can be found in the reminder of this document. For more details on usage and establishing basic communication with the encoder, see the examples for using the PM tformat library.

2 PM Installing the PM T-Format Library

2.1 PM tformat Library Package Contents

The tformat Library consists of the following components:

- Header files and software library for tformat interface
- Documentation - *Using Position Manager T-Format Library on IDDK HW and SW User's Guide* (SPRUI66)
- Example project showcasing tformat interface implementation on the TMDXIDDK379D hardware

2.2 tformat Library

Library contents are available at the following locations:

<base> install directory is:

```
C:\ti\controlSUITE\libs\app_libs\position_manager\tformat\vX.X
```

The following sub-directory structure is used:

```
<base>\Doc       Documentation
<base>\Float     Contains implementation of the library and corresponding include file
<base>\examples  Example using tformat library
```

3 Module Summary

This section describes the contents of PM_tformat_Include.h, the include file for using the tformat library.

3.1 PM tformat Library Commands

Details of the protocol and commands supported in different modes can be obtained from the encoder documentation.

Table 2. Commands Supported

Command	
DATAID0	Data Readout
DATAID1	Data Readout
DATAID2	Data Readout
DATAID3	Data Readout
DATAID6	Writing to EEPROM
DATAIDD	Reading from EEPROM
DATAID7	Reset
DATAID8	Reset
DATAIDC	Reset

3.2 PM T-Format Library Functions

The tformat Library consists of the following functions that enable the user to interface with tformat encoders. Table 3 lists the functions existing in the tformat library and a summary of cycles taken for execution.

More details of the functions and their usage in the following sections.

Table 3. PM T-Format Library Functions

Name	Description	CPU Cycles	Type
PM_tformat_generateCRCTable	This function generates table of 256 entries for a given CRC polynomial (polynomial) with specified number of bits (nBits). Generated tables are stored at the address specified by pTable.	30226	Initialization time
PM_tformat_getCrc	Calculate the n-bit CRC of a message buffer by using the lookup table, to get the CRC of each byte. This function should be used for calculating CRC of the receive data	220	Run time
PM_tformat_setupCommand	Setup a SPI and other modules for a given command to be transmitted. All the transactions should start with this command. This function call sets up the peripherals for upcoming transfer but does not actually perform any transfer or activity on the encoder interface. This function call in turn populates the sdata array of TFORMAT_DATA_STRUCT with the data to be transmitted to the Encoder.	1160	Run time
PM_tformat_startOperation	This function will initiate the transfer on the interface. To be called after PM_tformat_setupCommand. Performs the transaction set up by earlier. Note that the setup up and start operation are separate function calls. User can setup the transfer when needed and start the actual transfer using this function call, as needed, at a different time.	46	Run time
PM_tformat_receiveData	Function for unpacking and populating the tformat data structure with the data received from Encoder. This function will be called when the data from Encoder is available in the SPI data buffer and transferred to rdata array of TFORMAT_DATA_STRUCT. Upon the function call, received data is unpacked as per the current command and unpacked results are stored accordingly.	500	Run time
PM_tformat_setupPeriph	Setup for SPI, CLB and other interconnect XBARs is performed with this function during system initialization. This function needed to be called after every system reset. No transactions will be performed until the setup peripheral function is called.	8822	Initialization time
PM_tformat_setFreq	Function to set the clock frequency. Clock Frequency = SYSCLK/(4*TFORMAT_FREQ_DIVIDER) - used during application (2.5 MHz)	220	Initialization time

1. Implies the CPU cycle data depends on Encoder under test, and the commands/data being used along with certain function. These numbers could vary significantly depending on the command and corresponding data, and so forth.

3.3 Data Structures

The PM tformat library defines the tformat data structure handle as below:

Object Definition:

```
typedef struct  {               // bit descriptions
    uint16_t  controlField;
    uint16_t  statusField;
    uint16_t  dataField0;
    uint16_t  dataField1;
    uint16_t  dataField2;
    uint16_t  dataField3;
    uint16_t  dataField4;
    uint16_t  dataField5;
    uint16_t  dataField6;
    uint16_t  dataField7;
    uint16_t  crc;
    uint16_t  eepromAddress;
    uint16_t  eepromWrDtata;
    uint16_t  eepromRdDtata;
    volatile struct SPI_REGS *spi;
    uint32_t  sdata[16];     // Send data buffer
    uint32_t  rdata[16];     // Receive data buffer
    uint16_t  dataReady;
    uint16_t  fifo_level;
    uint32_t  rxPkts[3];
} TFORMAT_DATA_STRUCT;
```

Table 4. Module interface Definition

Module Element Name	Description	Type
controlField	Control Field received from encoder (CF)	16 bits
statusField	Status Field received from encoder (SF)	16 bits
dataField0	Data Field 0 received from encoder (DF0)	16 bits
dataField1	Data Field 1 received from encoder (DF1)	16 bits
dataField2	Data Field 2 received from encoder (DF2)	16 bits
dataField3	Data Field 3 received from encoder (DF3)	16 bits
dataField4	Data Field 4 received from encoder (DF4)	16 bits
dataField5	Data Field 5 received from encoder (DF5)	16 bits
dataField6	Data Field 6 received from encoder (DF6)	16 bits
dataField7	Data Field 7 received from encoder (DF7)	16 bits
eepromAddress	EEPROM address for read write accesses (ADF)	16 bits
eepromWrDtata	EEPROM Write data for write accesses (EDF)	16 bits
eepromRdDtata	EEPROM read data received in read accesses (EDF)	16 bits
crc	CRC Field 7 received from encoder (CRC)	16 bits
spi	SPI instance used for tformat implementation	Pointer to Spi*Regs
dataReady	Flag indicating dataReady – set by PM_tformat_receiveData function, cleared by PM_tformat_setupCommand function	0 or 1
sdata	Internal variables used by library – for debug purposes.	Array of 32-bit unsigned integers
rdata	Internal variables used by library – for debug purposes.	Array of 32-bit unsigned integers
fifo_level	Internal variables used by library – for debug purposes.	Max value 8
rxPkts	Received data in packed 32 bit packets	32 bits

3.4 Details of Function Usage

Detailed description of various library functions in PM tformat library and their usage can be found in the following sections.

Table 5. Summary of Function Details

Title **Page**

PM_tformat_generateCRCTable

Directions

This function generates table of 256 entries for a given CRC polynomial (polynomial) with specified number of bits (nBits). Generated tables are stored at the address specified by pTable.

Definition

```
void PM_tformat_generateCRCTable(uint16_t nBits, uint16_t polynomial, uint16_t
*pTable)
```

Parameters

Input	
nBits	Number of bits of the given polynomial
polynomial	Polynomial used for CRC calculations
pTable	Pointer to the table where the CRC table values are stored
Return	
None	

Usage

```
#define NBITS_POLY1  8
#define POLY1        0x01 SIZEOFTABLE  256
uint16_t tformatCRCtable [SIZEOFTABLE];

    // Generate table for poly 1
    PM_tformat_generateCRCTable(
        NBITS_POLY1,
        POLY1,
        tformatCRCtable);
```

PM_tformat_getCrc

Description Calculate CRC of a message buffer by using the lookup table, to get the CRC of each byte

Definition

```
uint32_t PM_tformat_getCrc (uint16_t input_crc_accum, uint16_t nBitsData,
uint16_t nBitsPoly, uint16_t * msg, uint16_t *crc_table, uint16_t rxLen)
```

Parameters

Input	
input_crc_accum	Initial CRC value (seed) for CRC calculation
nBitsData	Number of bits of data for which the CRC needs will be calculated
nBitsPoly	Number of bits of polynomial used for CRC computations
msg	Pointer to the data on which CRC will be computed
crc_table	Pointer to the table where the CRC table values are stored
rxLen	Number of bytes of the data for CRC calculation
crc	n-bit CRC value calculated

Usage

```
Define tformat Data structure during initialization.
TFORMAT_DATA_STRUCT tformatData;

Example Code:
crcResult =  PM_tformat_getCRC(
                         0,                                    // Initial seed
                         nBits,                                // number of bits
of data
                         NBITS_POLY1,                          // polynomial bits
                        (uint16_t *)& tformatData.rxPkts,      // Pointer to data
array
                         tformatCRCtable,                      // CRC table with
polynomial
                         4);                                   // number of bytes
```

PM_tformat_setupCommand

Description

Setup a SPI and other modules for a given command to be transmitted. All the transactions should start with this command. This function call sets up the peripherals for the upcoming transfer but does not actually perform any transfer or activity on the interface. This function call populates the sdata array of TFORMAT_DATA_STRUCT with the data to be transmitted to the encoder.

Definition

```
void Val = PM_tformat_setupCommand (uint16_t dataID, uint16_t eepromAddr,
uint16_t eepromData, uint16_t crc);
```

Parameters

Input	
dataID	Mode command for the transfer to be done.
eepromAddr	EEPROM Address depending on the mode command for read or write access
eepromData	EEPROM Data depending on the mode command for write access
crc	CRC value to be sent to the encoder in case of EEPROM accesses
Return	
Val	If incorrect, command value is passed to this function, which would return zero. For all other cases function returns a value of one.

Usage

Example Code:

```
Val = PM_tformat_setupCommand (DATAIDD, address, 0, crcResult);
PM_tformat_startOperation();
while (tformatData.dataReady != 1) {}
retval1 = PM_tformat_receiveData(DATAIDD);
crcResult = PM_tformat_getCRC(0, 32, 8, (uint16_t *)&tformatData.rxPkts,
tformatCRCtable, 4);
```

PM_tformat_receiveData

Description

Function for unpacking and populating the tformat data structure with the data received from encoder. This function will be called when the data from encoder is available in the SPI data buffer and transferred to rdata array of TFORMAT_DATA_STRUCT. Upon the function call, received data is unpacked as per the current command and unpacked results are stored accordingly.

NOTE: The format for transfer of position values varies in length depending on the encoder model. Encoder transmits the position value with LSB first. For further details, see the Encoder documentation provided by encoder manufacturer.

Definition

```
uint16_t PM_tformat_receiveData (uint16_t dataID);
```

Parameters

Input	
dataID	Mode command for the tformat transfer done. This function should be called with the same mode command that was used to initiate the transfer.
Return	
val	If the incorrect command value is passed to this function it will return zero. For all other cases, function returns a value of one.

Usage

Example Code:

```
retval1 = PM_tformat_setupCommand (DATAID3, 0, 0, 0);
PM_tformat_startOperation();
while (tformatData.dataReady != 1) {}
retval1 = PM_tformat_receiveData(DATAID3);
```

PM_tformat_startOperation

Description

This function initiates the transfer to the encoder. This function should only be called after PM_tformat_setupCommand. Hence, the PM_tformat_startOperation function kick starts the transaction that was set up earlier by PM_tformat_setupCommand. Note that the setup up and start operation are separate function calls. User can setup the transfer when needed and start the actual transfer using this function call, as needed, at a different time.

Definition

```
void PM_tformat_startOperation(void);
```

Parameters

Input	
	None
Return	
	None

Usage

Example Code:

```
retval1 = PM_tformat_setupCommand (DATAIDD, address, 0, crcResult);
PM_tformat_startOperation();
while (tformatData.dataReady != 1) {}
retval1 = PM_tformat_receiveData(DATAIDD);
```

This function clears the tformatData.dataReady flag zero when called. This flag should subsequently be set by the SPI Interrupt service routine when the data is received from the encoder. You can poll for this flag to know if the data from the encoder is successfully received after the PM_tformat_startOperation function call.

PM_tformat_setupPeriph

Description

Setup for SPI, CLB and other interconnect XBARs for tformat are performed with this function during system initialization. This function needed to be called after every system reset. No tformat transactions will be performed until the setup peripheral function is called.

Definition

```
void PM_tformat_setupPeriph (void);
```

Parameters

Input	
	None
Return	
	None

Usage

Example Code:

```
tformatData.spi = &SpibRegs;
PM_tformat_setupPeriph();
```

The PM tformat library used an instance of SPI for communication. For proper initialization, SPI instance needs to be set in tformatData.spi before calling this function, as shown above

PM_tformat_setFreq

Description

Function to set the tformat clock frequency. tformat transactions occur at fixed data rate of 2.5 MHz.

Definition

```
void PM_tformat_setFreq(uint32_t Freq_us);
Data Rate = SYSCLK/(4* Freq_us)
```

Parameters

Input	
	Freq_us: A clock divider for the system clock sets tformat Clock Frequency = SYSCLK/(4* Freq_us)
Return	
	None

Usage

Example Code:

```
PM_tformat_setFreq(TFORMAT_FREQ_DIVIDER);
```

4 Using PM_tformat Library

4.1 Adding tformat Lib to the Project

1. Include the library in *{ProjectName}-Includes.h*.

   ```
   #include "PM_tformat_Include.h"
   ```

   Add the PM_tformat library path in the include paths under Project Properties → Build → C2000 Compiler → Include Options.

   Path for the library: C:\ti\controlSUITE\libs\app_libs\position_manager\v1_00_00_00\tformat\Float\lib

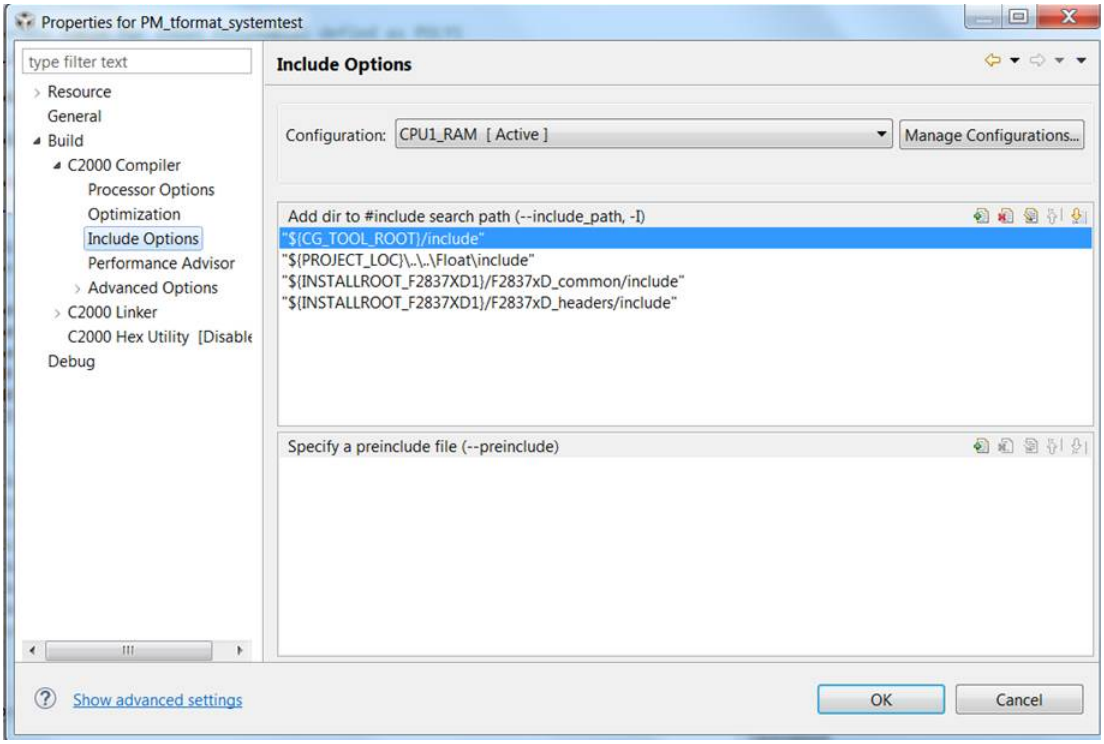

Figure 3. Compiler Options for a Project Using PM tformat Library

NOTE: Note that the exact location may vary depending on where controlSUITE is installed and which other libraries the project is using.

Link tformat Library: (PM_tformat.lib) to the project, it is located at:
controlSUITE\libs\app_libs\position_manager\v01_00_00_00\tformat\Float\lib

2. Figure 4 shows the changes to the linker options that are required to include the tformat library.

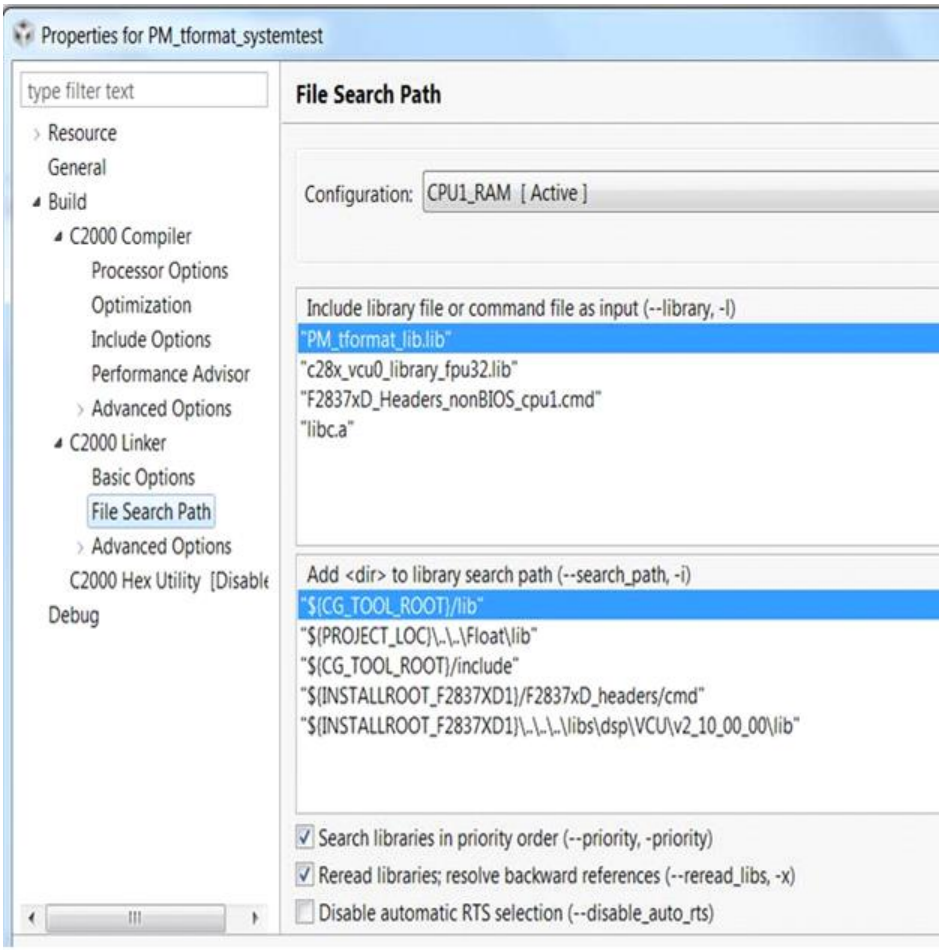

Figure 4. Adding PM tformat Library to the Linker Options in CCS Project

NOTE: Note that the exact location may vary depending on where controlSUITE is installed and which other libraries the project is using.

4.2 Steps for Initialization

The following steps are needed for initialization and proper functioning of the tformat library functions. For more details, see the examples provided along with the library.

1. Create and add module structure to {ProjectName}-Main.c.

```
TFORMAT_DATA_STRUCT tformatData;              //PM tformat data structure
```

2. Define the frequency of the clock divider.

```
#define TFORMAT_FREQ_DIVIDER        20
```

3. Set the SPI instance to be used for communication. For usage on TMDXIDDK379D, the SPI instance has to be set to SpiB and the SpiB receive FIFO interrupt has to be enabled.

```
tformatData.spi = &SpibRegs;
PieVectTable.SPIB_RX_INT = &spiRxFifoIsr;
PieCtrlRegs.PIECTRL.bit.ENPIE = 1;     // Enable the PIE block
PieCtrlRegs.PIEIER6.bit.INTx3 = 1;     // Enable PIE Group 6, INT 9
IER = 0x20;                            // Enable CPU INT6
EINT;
```

NOTE: Alternatively, users can also poll for the Interrupt Flag and not necessarily use interrupt. Make sure the SPIRXFIFO contents are copied into tformatData.rdata after the flag is set. This is required to be executed before calling PM_tformat_receiveData function. Also, interrupt flag needs to be cleared to get the SPI ready for next tformat transaction.

```
for (i=0;i<=tformatData.fifo_level;i++){tformatData.rdata[i]= tformatData.spi->SPIRXBUF;}
tformatData.spi->SPIFFRX.bit.RXFFINTCLR=1;  // Clear Interrupt flag
```

4. Enable clocks to EPWM Instances 1, 2, 3 and 4.

```
tformatData.spi = &amp;SpibRegs;
    PM_tformat_setupPeriph();
```

5. Initialize and setup peripheral configuration by calling PM_tformat_setupPeriph function.

```
tformatData.spi = &amp;SpibRegs;
PM_tformat_setupPeriph();
```

6. Setup GPIOs needed for configuration, which is required for TMDXIDDK379D. The GPIOs used for SPI have to be changed based on the chosen SPI instance. GPIO6, GPIO7 and GPIO34 have to be configured always.

```
GpioCtrlRegs.GPAMUX1.bit.GPIO6 = 1;   // Configure GPIO6 as SPI Clk master

GpioCtrlRegs.GPAGMUX2.bit.GPIO24 = 1;
GpioCtrlRegs.GPAGMUX2.bit.GPIO25 = 1;
GpioCtrlRegs.GPAGMUX2.bit.GPIO26 = 1;
GpioCtrlRegs.GPAGMUX2.bit.GPIO27 = 1;

GpioCtrlRegs.GPAMUX2.bit.GPIO24 = 2;  // Configure GPIO24 as SPISIMOB
GpioCtrlRegs.GPAMUX2.bit.GPIO25 = 2;  // Configure GPIO25 as SPISOMIB
GpioCtrlRegs.GPAMUX2.bit.GPIO26 = 2;  // Configure GPIO26 as SPICLKB
GpioCtrlRegs.GPAMUX2.bit.GPIO27 = 2;  // Configure GPIO27 as SPISTEB

GpioCtrlRegs.GPAQSEL2.bit.GPIO24 = 3; // Asynch input GPIO24 (SPISIMOB)
GpioCtrlRegs.GPAQSEL2.bit.GPIO25 = 3; // Asynch input GPIO25 (SPISOMIB)
GpioCtrlRegs.GPAQSEL2.bit.GPIO26 = 3; // Asynch input GPIO26 (SPICLKB)
GpioCtrlRegs.GPAQSEL2.bit.GPIO27 = 3; // Asynch input GPIO27 (SPISTEB)

GpioCtrlRegs.GPBMUX1.bit.GPIO34 = 1;  // Configure GPIO34 as TxEN

GpioCtrlRegs.GPBDIR.bit.GPIO32 = 1;   // Configure GPIO32 as Pwr Ctl
```

7. Setup XBAR as shown below. The GPIOs used for SPI has to be changed based on the chosen SPI instance.

```
//SPISIMOB on GPIO24 connected to CLB4 i/p 1 via XTRIP and CLB X-Bars
// XTRIP1 is used inside the CLB for monitoring received data from Encoder.
InputXbarRegs.INPUT1SELECT = 24;         // GPTRIP XBAR TRIP1 -> GPIO24
```

8. Initialization for CRC related object and table generation.

```
//CRC tables to be used for CRC calculations
uint16_t table1[SIZEOFTABLE];

// Generate table for poly 1 and generated tables are placed in
tformatCRCtable(NBITS_POLY1, POLY1, tformatCRCtable);

-- Constants are defined in PM_tformat_Include.h as below.
#define NBITS_POLY1  8
#define POLY1        0x01
#define SIZEOFTABLE  256
```

9. Turn the power ON. GPIO used for Power control can be changed based on the hardware. GPIO32 is used for power control on TMDXIDDK379D.

```
// Power up 5v supply through GPIO32
    GpioDataRegs.GPBDAT.bit.GPIO32 = 1;
```

5 Resource Requirements

The resources of the MCU (see Table 6) are consumed by the PM tformat Library for implementing the interface.

Table 6. Resource Requirements

Resource Name	Type	Purpose	Usage Restrictions
Dedicated Resources			
GPIO7	IO	SPI clock generated by MCU	IO dedicated for tformat implementation
EPWM4	IO	Internally for Clock generation	EPWM4 dedicated for tformat implementation
Input XBAR (NPUTXBAR1)	Module/IO	To be connected to SPISIMO of the corresponding SPI instance	INPUTXBAR1 is used for implementation. Remaining inputs are available for application use.
Output XBAR (OUTPUTXBAR1)	Module/IO	Bringing out tformat TxEn (Direction Control) signal on GPIO32 via OUTPUT1 of Output XBAR	OUTPUT1 is used for implementation. Remaining outputs are available for application use.
Configurable Resources			
SPI	Module and IOs	One SPI instance to emulate tformat interface (SPIB on IDDK)	Any instance of SPI can be chosen – Module and corresponding IOs will be dedicated for tformat
PwrCtl	IO	For encoder Power control	Can choose any IO power control. GPIO32 is used on IDDK and example projects.
TxEn	IO	Direction control on data	GPIO34 is used for this purpose on IDDK and example projects. User can choose any GPIO with OUTPUTXBAR1 mux option.
Shared Resources			
CPU and Memory	Module	Check CPU and Memory utilization for various functions	Application to ensure enough CPU cycles and Memory are allocated

6 Test Report

Table 7 lists tests with various types of encoders; cable lengths are performed. Tests include basic command set exercising and reading position values, with additional data if applicable.

Table 7. Test Report

Encoder Name	Type	Resolution	Cable Length in Meters 1	Max tformat Clock in MHz	Test Results
TS5643N100	Rotary	17 bits	70m	2.5 MBPS	Pass
TS5700N8501	Rotary	24 bits	70m	2.5 MBPS	Pass

1. Cable lengths up to 100m have also been tested with some of the encoders.

7 FAQs

1. **Question:** Why is the position information not received or received incorrectly?

   Answer: Perform or cross check the following:

   - Make sure the Power-On procedure, specific to the encoder, is used properly.
   - Check for encoder error messages and warnings
   - Clear the Errors and Warnings if any, and perform encoder Reset
   - Check CRC of the received position data

   For further help, see the encoder documentation provided by encoder manufacturer.
2. **Question:** What are the limitations of the tformat implementation with this library?

   Answer: While using the tformat library, see Section 1.4.
3. **Question:** I have encountered a CRC failure. What could be the reason for this?

   Answer: CRC errors can occur due to noise in transmission path and several other reasons detailed in encoder documentation provided by encoder manufacturer.
4. **Question:** How is the tformat interface implemented on the TMS320F28379D devices?

   Answer: For details, see Section 1.4.
5. **Question:** Does TI share the source for the PM tformat library to customers?

   Answer: TI does not share the source code with customers. For specific requests, contact the TI sales team.
6. **Question:** Does TI provide application level interface functions for tformat?

   Answer: Basic usage examples are provided along with the library. The example has high-level application layer functions for initialization, running full command set, setting and reading parameters, performing cable propagation delay compensation, enabling additional data, checking CRC for various types of received data, and so forth, which are sufficient for most of the applications. Any additional application layer functionality should be developed by using the basic driver interface commands provided in the PM tformat Library.

8 References

- Power Supply Reference design for tformat 2.2 encoder interfaces can be obtained from Texas Instruments (TIDA-00172): http://www.ti.com/tool/TIDA-00172
- Documentation from Heidenhain:
  - Bidirectional Synchronous-Serial Interface for Position Encoders
  - tformat Application notes
- C2000 DesignDRIVE Development Kit for Industrial Motor Control - TMDXIDDK379D:
  - *Using Position Manager T-Format Library on IDDK Hardware and Software User's Guide* (SPRUI66)
  - *DesignDRIVE Development Kit IDDK Hardware Reference Guide* (SPRUI23)
  - *DesignDRIVE Development Kit IDDK User's Guide* (SPRUI24)
  - *Using Position Manager tformat22 Library on IDDK Hardware User's Guide* (SPRUI34)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

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