C2000 Position Manager PTO Library

Application Report

Literature Number: SPRAC77
March 2017

Contents

C2000 Position Manager PTO Library

1 Description

Pulse-train output (PTO) is a generic name for describing various forms of pulse outputs. This library assists in generating various outputs of QEP, CwCCW, pulse and direction, and more.

1.1 PTO – QepDiv

The QepDiv PTO function can be used to generate a divided pulse stream from QEP inputs. Figure 1 shows the QepDiv input and output diagram.

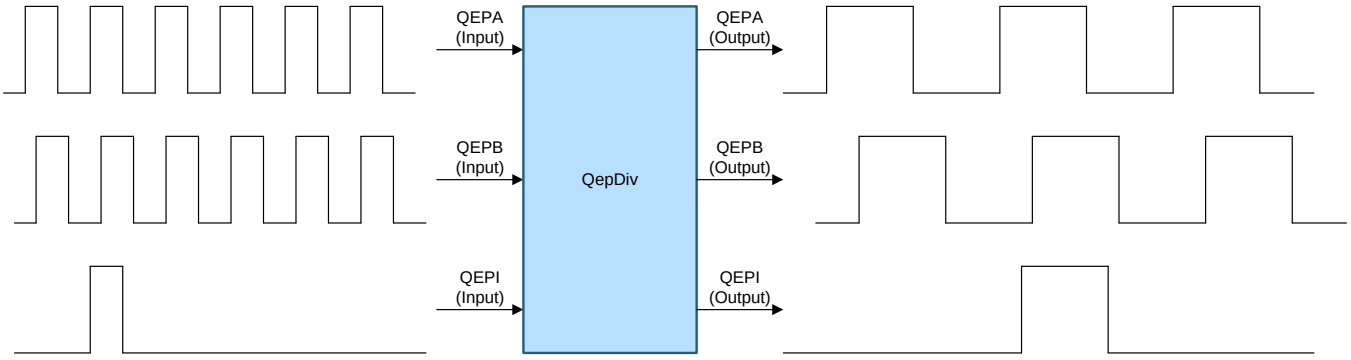

Figure 1. QepDiv Input and Output Diagram

Figure 2 shows the implementation diagram of the QepDiv interface.

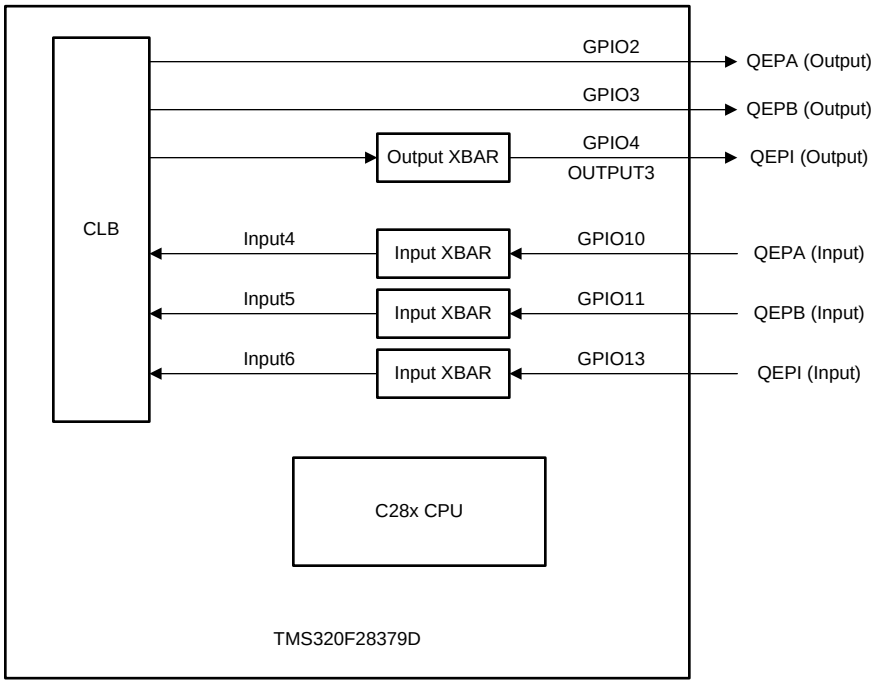

Figure 2. Implementation Diagram

Chip-level inputs to the PTO-QepDiv interface:

- QEPA input
- QEPB input
- QEPI input

Chip-level outputs to the PTO-QepDiv interface:

- QEPA output
- QEPB output
- QEPI output

1.1.1 Implementation of PTO-QepDiv Interface

This section provides an overview of how the PTO-QepDiv interface is implemented on TMS320F28379D devices. This interface is primarily achieved by the following components:

- CPU
- Configurable logic block (CLB)
- Device interconnect (XBARs)

The following functions are implemented within the CLB module.

- Monitors to QEPA, QEPB, and index inputs connected to the GPIO.
- Detects the direction of the motion and any changes in direction.
- Detects the edges of the input signals.
- Implements the division function and generates the QEPA, QEPB, and index outputs.

NOTE: The CLB module are only accessible through library functions provided in Texas Instruments™ library functions and are not otherwise configurable by users.

Input and output XBARs are used as input- and output-signal routing to and from the CLB as applicable. The CPU initializes the function, configuration of the CLB, XBARs, and GPIOs as applicable.

The QepDiv interface implements division factors of /1, /2, /4, /8, and continues this pattern up to /1024 and /2048.

The PTO-QepDiv operation has the following usage limitations:

- The maximum frequency of the input signals (QEPA and QEPB) is limited to 5 MHz.
- The index pulse is generated on the index output when a rising edge is detected on the index input signal.
- The width of the index pulse can be user defined. See the PM_pto_qepdiv_config function and the corresponding example provided in controlSuite.
- The divider values work as follows:
  - The frequency of the output QEPA or QEPB = frequency of input QEPA or QEPB / (2 × divider)

1.2 PTO – PulseGen

The PTO-PulseGen function can be used to generate pulse and direction outputs as required by the application. Figure 3 shows the PulseGen output diagram.

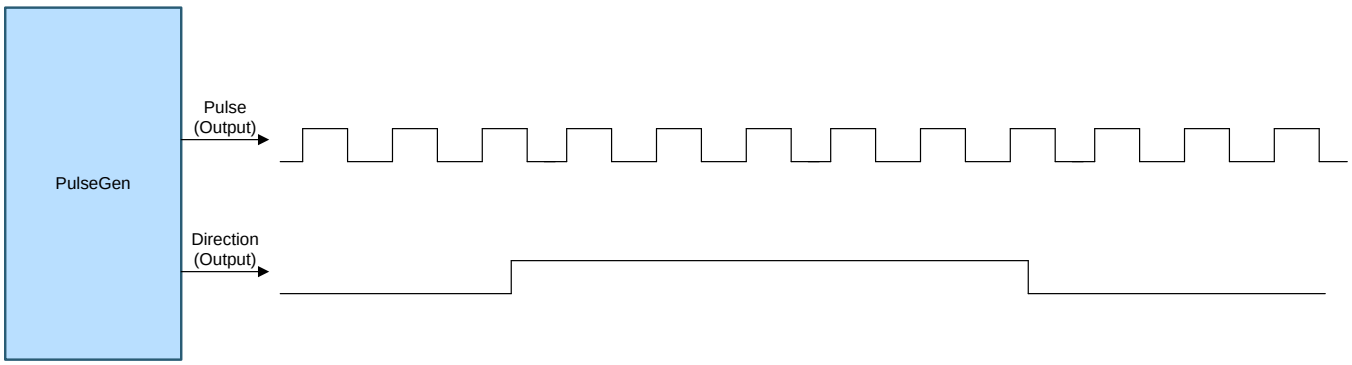

Figure 3. PulseGen Output Diagram

Figure 4 shows the implementation diagram of the PulseGen interface.

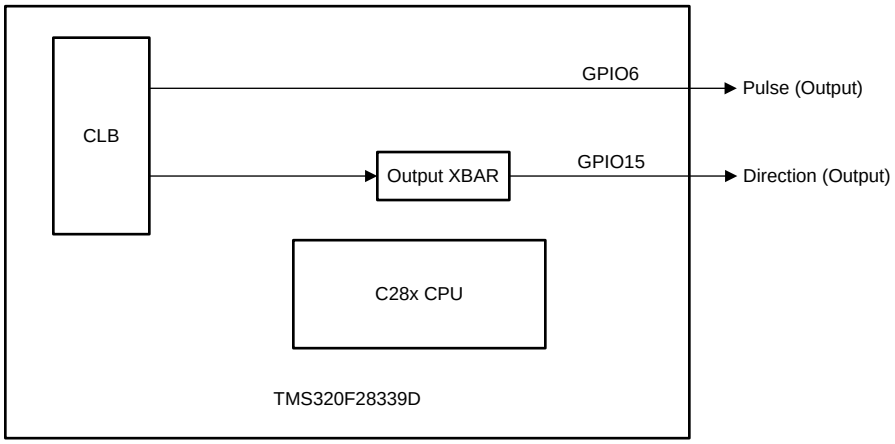

Figure 4. Implementation Diagram

Chip-level inputs to the PTO-PulseGen interface: none.

Chip-level outputs to the PTO-PulseGen interface:

- Pulse output
- Direction output

The PTO-PulseGen operation has the following usage limitations:

- The minimum number of cycles must be 1000 cycles for the PTO period (at 200-MHz system clock, this corresponds to 10 µs [for example, 100 KHz]).
- The number of cycles must be between 40% to 60% of the PTO period for the interrupt time to avoid conflicts with PTO updates.
- The maximum frequency of the PTO-PulseGen output is 5 MHz at a 200-MHz CPU CLk. See the example provided in controlSuite.

1.2.1 Implementation of PTO-PulseGen Interface

This section provides an overview of how the PTO-PulseGen interface is implemented on TMS320F28379D devices. This interface is achieved primarily by the following components:

- CPU
- CLB
- XBARs

The following functions are implemented within the CLB module.

- Software to provide the number of pulses and the duration of each pulse.
- The CLB generates the pulses and is defined by the software interface function.
- Direction input is applied as defined by the software interface function.

NOTE: The CLB module are only accessible through library functions provided in TI library functions and are not otherwise configurable by users.

Output XBARs are used for output-signal routing to and from the CLB, as applicable. The CPI is used at the start to initialize the PulseGen interface initialization function, configuration of the CLB, XBARs, and GPIOs as required.

2 PM PTO Library Installation

2.1 Library Package Contents

The PTO library consists of the following components:

- Header files and a software library for the PTO interface
- Documentation (*PTO Library User's Guide*)
- Project showcasing the PTO interface implementation on F28379D hardware (example: ControlCard)

2.2 PTO Library

The PTO library is located at the following base directory:
C:\ti\controlSUITE\libs\app_libs\position_manager\pto\vx.x.

The following subdirectory structure is used:

- <base directory>\Doc – Documentation
- <base directory>\Float – Contains implementation of the library and corresponding include file
- <base directory>\examples – Example using PTO library

3 Module Summary

This section details the contents of the PTO library functions.

- PM_pto_pulsegen_Include.h – Include file for using the PTO PulseGen function library.
- PM_pto_qepdiv_Include.h – include file for using the PTO QepDiv function library.

3.1 PM PTO Library Functions

Table 1 lists the functions existing in the PTO library and a summary of cycles taken for execution. See Section 3.2 and Section 3.3 for more details of the functions and their uses.

Table 1. PTO Library Functions

Name	Description	CPU Cycles	Type
PulseGen			
PM_pto_pulsegen_reset	Used to reset the pulsegen parameters set by earlier configuration and start a fresh setup. This function needs to be called in case the pulse generation needs to be reset and started again at a later stage.	238	Initialization time
PM_pto_pulsegen _startOperation	This function will initiate the pulse generation on the interface. To be called after PM_pto_pulsegen_setupPeriph. Performs the transaction set up by earlier function. Note that the setup up and start operation are separate function calls. User can setup the peripherals when needed and start the actual pulse generation using this function call, as needed, at a different time.	28	Run time
PM_pto_pulsegen _runPulseGen	A runtime function to be called periodically for dynamically configuring and changing the pulse generation requirements as needed by the application. This function needs to be called periodically with appropriate parameters like the number of pulses, period, duration etc. Details in the later section.	267	Run time
PM_pto_pulsegen _setupPeriph	Setup for 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.	2479	Initialization time
QepDiv			
PM_pto_qepdiv_reset	Used to reset the qepdiv parameters set by earlier configuration and start a fresh setup. This function needs to be called in case the pulse generation needs to be reset and started again at a later stage.	258	Initialization time

Table 1. PTO Library Functions (continued)

Name	Description	CPU Cycles	Type
PM_pto_qepdiv _startOperation	This function will initiate the pulse generation on the interface. To be called after PM_pto_qepdiv_setupPeriph. Performs the transaction set up by earlier function. Note that the setup up and start operation are separate function calls. User can setup the peripherals when needed and start the actual pulse generation using this function call, as needed, at a different time.	65	Run time
PM_pto_qepdiv_config	This function configures the divider, the divider value cannot be changed dynamically. User needs to reset the module using PM_pto_qepdiv_reset before reconfiguring the functionality.	149	Run time
PM_pto_qepdiv _setupPeriph	Setup for 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.	4780	Initialization time

3.2 Details of Function Usage (PulseGen)

3.2.1 PM_pto_pulsegen_runPulseGen

Description

A runtime function to be called periodically for dynamically configuring and changing the pulse generation requirements as required by the application. This function must be called periodically with appropriate parameters like the number of pulses, period, duration, and more.

Definition

```
uint16_t PM_pto_pulsegen_runPulseGen(
        uint32_t PulseLo,
        uint32_t PulseHi,
        uint32_t ptoActivePeriod,
        uint32_t ptoFullPeriod,
        uint32_t ptoInterruptTime,
        uint16_t ptoDirection,
        uint16_t run
        );
```

Parameters

Input:

- PulseLo – Low pulse width
- PulseHi – High pulse width
- ptoActivePeriod – Period the pulses are sent out; less than ptoFullPeriod
- ptoFullPeriod – Full PTO period
- ptoInterruptTime – Time when that the interrupt is generated to the CPU
- ptoDirection – Direction output; latched as it is on direction output at the beginning of new period
- run – Value indicting 1-run and 0-stop. Sampled at the beginning of the new period to determine to continue or halt the pulse generation

Return:

- Val – If the function is executed successfully, the function will return ptoFullPeriod as the return value

Usage

In pto_pulsegen.c, a sample configuration function called pto_setOptions is provided as an example to assist with the PM_pto_pulsegen_runPulseGen function and to perform the intermediate calculations. See the following code sample calculation that illustrates how various parameters for this function can be generated. See the pto_pulsegen.c and pto_pulsegen.h files for more details.

```
uint32_t pto_setOptions(
```

```
            uint32_t numPulses, //number of pulses needed to be generated in next period
            uint32_t Period,        // PTO period in clock cycles
            uint32_t ptoInterruptTime, // Interrupt generation time
            uint16_t ptoDirection,    // Direction output
            uint16_t run)        //run-stop condition.
{
    uint32_t pulseFreq, reminder;
    uint32_t PulseLo;
    uint32_t PulseHi;
    uint32_t ptoActivePeriod;
    uint32_t ptoFullPeriod;


    pulseFreq = Period / numPulses;
    reminder = Period - (pulseFreq * numPulses);
    PulseLo = (pulseFreq/2 );
    PulseHi = pulseFreq;
    ptoActivePeriod = (pulseFreq * numPulses);
    ptoFullPeriod = Period;

    PM_pto_pulsegen_runPulseGen(
            PulseLo,
            PulseHi,
            ptoActivePeriod,
            ptoFullPeriod,
            ptoInterruptTime,
            ptoDirection,
            run);

    return(reminder);
}
```

3.2.2 PM_pto_startOperation

Description

This function initiates the pulse generation. This function must be called after PM_pto_pulsegen_setupPeriph. Hence, the PM_pto_pulsegen_startOperation kick starts the pulse generation that was set up earlier.

NOTE: The setup and start operations are separate function calls. Users can set up the transfer and start the pulse generation by using this function call, as required, at a different time.

Definition

```
void PM_pto_pulsegen_startOperation(void);
```

Parameters

Input: none

Return: none

Usage

Example code:

```
pto_pulsegen_Init();
    DELAY_US(800L);
    PM_pto_pulsegen_startOperation();
    retval1 = PM_pto_pulsegen_runPulseGen(7, 15, 960, 990, 500, 1, 1);
```

3.2.3 PM_pto_pulsegen_setupPeriph

Description

This function performs the setup for the CLB and other interconnect XBARs during system initialization. This function must be called after every system reset. No transactions will be performed until the setup peripheral function is called.

Definition

```
void PM_pto_pulsegen_setupPeriph (void);
```

Parameters

Input: none

Return: none

Usage

Example code:

```
PM_pto_pulsegen_setupPeriph();
```

3.2.4 PM_pto_pulsegen_reset

Description

This function resets the pulsegen parameters set by the earlier configuration (PulseGen function calls) and starts a new setup. This function must be called in case the pulse generation must be reset and started again at a later stage.

Definition

```
void PM_pto_pulsegen_reset (void);
```

Parameters

Input: none

Return: none

Usage

Example code:

```
PM_pto_pulsegen_reset();
```

3.3 Details of Function Usage (QepDiv)

3.3.1 PM_pto_qepdiv_config

Description

This function configures the divider. The divider value cannot be changed dynamically. Users must reset the module using PM_pto_qepdiv_reset before reconfiguring the functionality.

Definition

```
PM_pto_qepdiv_config(uint16_t Divider, uint16_t IndexWidth);
```

Parameters

Input:

- Divider – Value of the divider
- Index width – Number of cycles for which the index pulse output is kept on

Return:

- Val – If the function is executed successfully, it will return ptoFullPeriod as the return value

Usage

Example code:

```
retval1 = PM_pto_qepdiv_config(4, 10);
    PM_pto_qepdiv_startOperation(1);
```

3.3.2 PM_pto_startOperation

Description

This function initiates the pulse generation. This function must only be called after PM_pto_qepdiv_setupPeriph. Hence, the PM_pto_qepdiv_startOperation function kick starts the pulse generation that was set up earlier.

NOTE: The setup and start operations are separate function calls. Users can set up the transfer and start the pulse generation by using this function call, as required, at a different time.

Definition

```
void PM_pto_qepdiv_startOperation(uint16_t run);
```

Parameters

Input (parameters to be passed to start or stop the function):

- Start = 1
- Stop = 0

Return: none

Usage

Example code:

```
retval1 = PM_pto_qepdiv_config(4, 10);
    PM_pto_qepdiv_startOperation(1);
```

3.3.3 PM_pto_qepdiv_setupPeriph

Description

Setup for the CLB and other interconnect XBARs is performed with the PM_pto_qepdiv_setupPeriph function during system initialization. This function must be called after every system reset. No transactions will be performed until the setup peripheral function is called.

Definition

```
void PM_pto_qepdiv_setupPeriph (void);
```

Parameters

Input: none

Return: none

Usage

Example code:

```
PM_pto_qepdiv_setupPeriph();
```

3.3.4 PM_pto_qepdiv_reset

Description

Used to reset the qepdiv parameters set by earlier configurations and to begin a fresh setup. This function must be called in case the pulse generation must be reset and started again at a later stage.

Definition

```
void PM_pto_qepdiv_reset (void);
```

Parameters

Input: none

Return: none

Usage

Example code:

```
PM_pto_qepdiv_reset();
```

4 Using PM_pto Library

4.1 Adding PTO Lib to Projects

Use the following instructions to add the PTO library to a project.

1. Include the library path in ProjectName-Include.h.

```
#include "PM_pto_pulsegen_Include.h"
#include "PM_pto_qepdiv_Include.h"
```

2. Navigate to Project Properties → Build → C2000 Compiler → Include Options, then add the PM_pto library path in the include paths (see Figure 5).

   The path for the library is C:\ti\controlSUITE\libs\app_libs\position_manager\v01_00_00_00\pto\Float\lib.

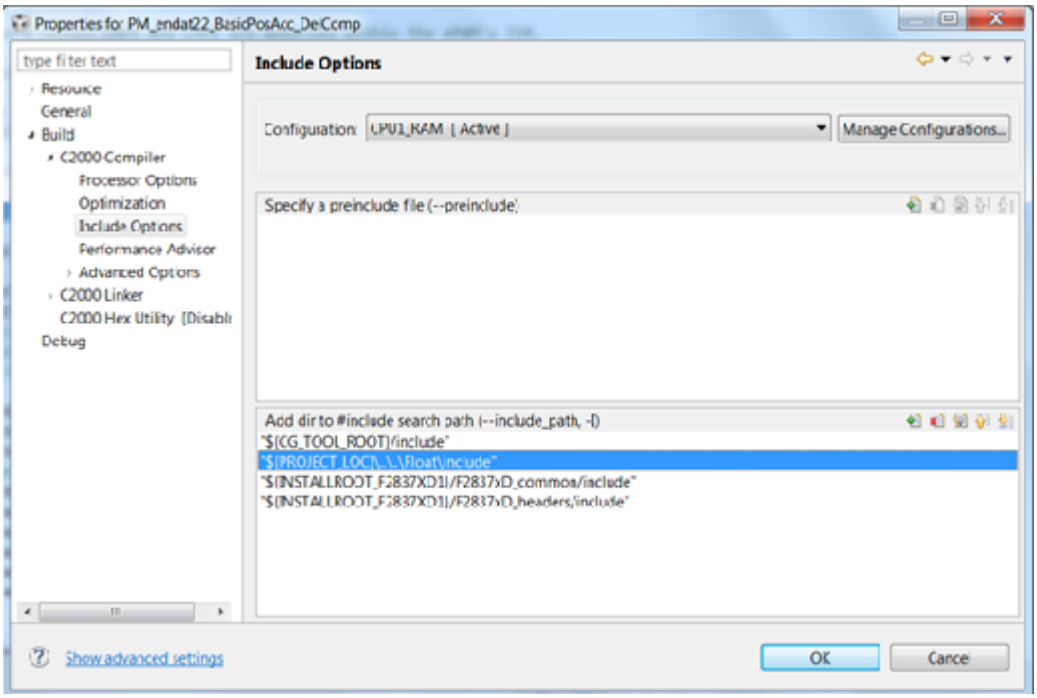

Figure 5. Compiler Options for Projects Using PM PTO Library

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

3. Link the PTO library (PM_pto_pulsegen_lib.lib and PM_pto_qepdiv_lib.lib) to the project (located at C:\ti\controlSUITE\libs\app_libs\position_manager\v01_00_00_00\pto\Float\lib)

Figure 6 and Figure 7 show the changes to the linker options that are required to include the PTO library.

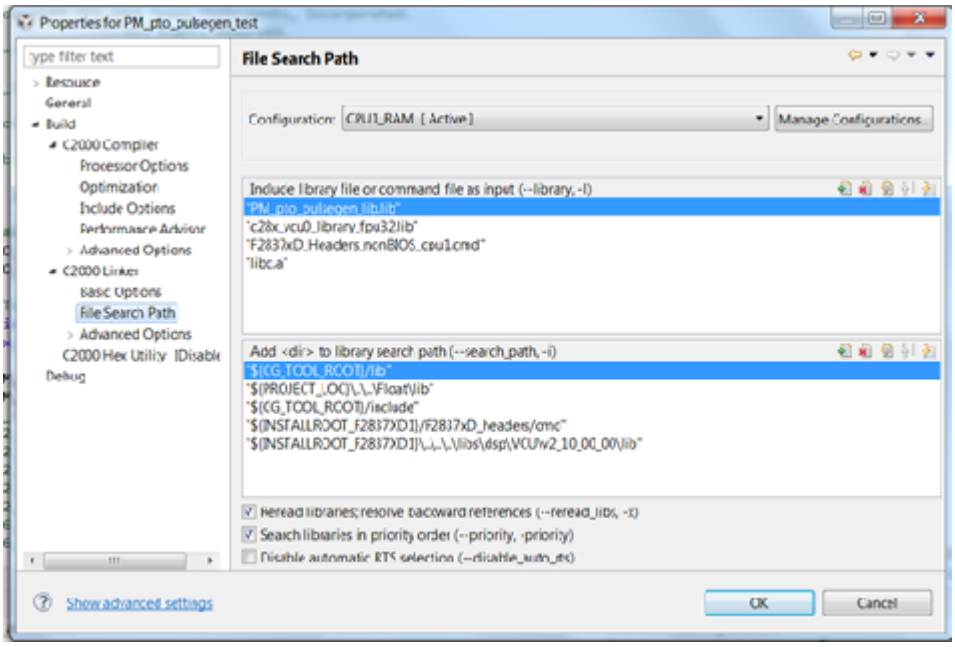

Figure 6. C2000 Linker Options – PulseGen

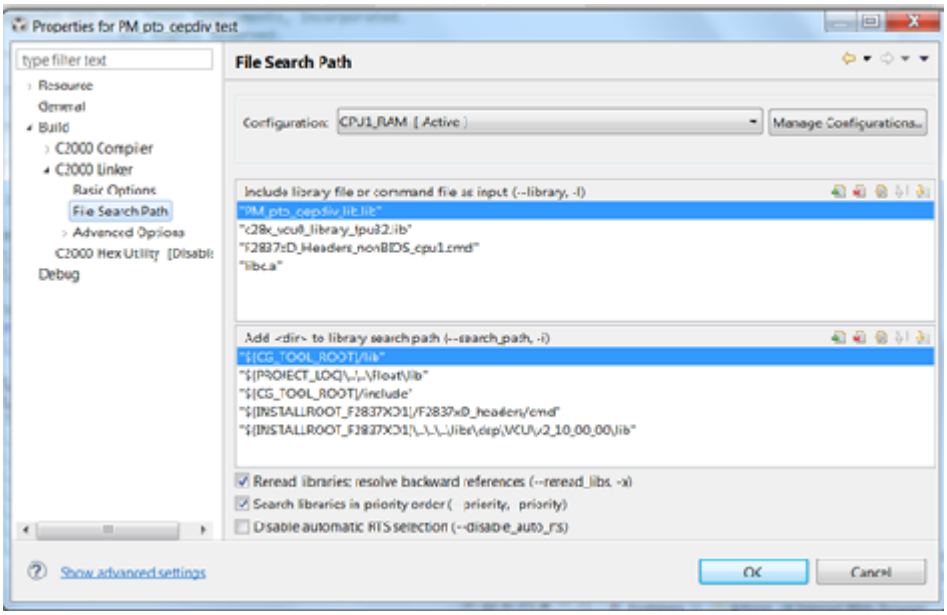

Figure 7. C2000 Linker Options – QepDiv

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

4.2 Initialization Steps

4.2.1 PTO-PulseGen Library

The following steps are required for initialization and proper function of the PTO PulseGen library. Refer to the examples provided with the library for more details.

1. Enable the clocks to EPWM4.

```
CpuSysRegs.PCLKCR2.bit.EPWM4 = 1;
```

2. Initialize and set up the peripheral configuration by calling the PM_pto_pulsegen_setupPeriph function.

```
PM_pto_pulsegen_setupPeriph();
```

3. Set up the GPIOs required for configuration. This step is for the F28379D device.

```
GpioCtrlRegs.GPAMUX1.bit.GPIO6 = 1; // Configure GPIO6 (Pulse out)

    GpioCtrlRegs.GPAGMUX1.bit.GPIO15 = 1; // Configure GPIO15 as direction output - outxbar4
    GpioCtrlRegs.GPAMUX1.bit.GPIO15 = 2;
```

4. Refer to the pto_setOptions function to set the pulse generation configuration.

   ptolsr is used as the primary ISR to update the PTO configuration in the example.

4.2.2 PTO-QepDiv Library

The following steps are required for initialization and proper function of the PTO QepDiv library. Refer to the examples provided with the library for more details.

1. Enable the clocks to EPWM2 and EPWM1.

```
CpuSysRegs.PCLKCR2.bit.EPWM2 = 1;
    CpuSysRegs.PCLKCR2.bit.EPWM1 = 1;
```

2. Initialize and set up the peripheral configuration by calling the PM_pto_qepdiv_setupPeriph function.

```
PM_pto_qepdiv_setupPeriph();
```

3. Set up the GPIOs required for configuration. This step is for the F28379D device.

```
//QEP inputs to be tapped from GPIO10/11/13 - via InputXBar Input4/5/6
    InputXbarRegs.INPUT4SELECT = 10;    //  QEPA
    InputXbarRegs.INPUT5SELECT = 11;    //  QEPB
    InputXbarRegs.INPUT6SELECT = 13;    //  QEPI

//QEP outputs available on GPIO2/3 - QEPA/B (over EPWM2A/B)
    GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 1; // Configure GPIO2 (Pulse out A)
    GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 1; // Configure GPIO3 (Pulse out B)
//QEP outputs Index on OUTPUTXBAR3 - on GPIO4 (users can choose any GPIO with OUTPUTXBAR3
    GpioCtrlRegs.GPAGMUX1.bit.GPIO4 = 1; // Index output - outxbar3
    GpioCtrlRegs.GPAMUX1.bit.GPIO4 = 1;
```

4. Refer to the PM_pto_qepdiv_config function to set the configuration.
5. Run PM_pto_qepdiv_startOperation to kick start the qepdiv configuration in the example.
6. Test the setup used in the example.

   In the example provided for the pto_qepdiv library, spare EPWMs are used to provide QEP inputs. Customers must connect these EPWM outputs to the QEP inputs or feed external signals directly.

```
EPWM4A (GPIO6)-> EQEPA (GPIO10)
EPWM5A (GPIO8)-> EQEPB (GPIO11)
EPWM4B (GPIO7)-> Index (GPIO13)
// These are just for test purposes and do not correspond to real time usage
```

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Date	Revision	Notes
March 2017	SPRAC77*	Initial release

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources 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.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

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