Application Note

AN2369/D
Rev. 0, 10/2002

Using the Discrete Input / Output TPU Function (DIO) with the MPC500 Family

Jeff Loeliger
TECD

This TPU Programming Note is intended to provide simple C interface routines to the discrete input/output TPU function (DIO). [1] The routines are targeted for the MPC500 family of devices, but they should be easy to use with any device that has a TPU. However, because the DIO function has a different function number in some TPU ROMs, using these routines with anything other than the MPC500 family will require that the function number be changed in the C interface routines.

1 Functional Overview

The discrete input/output (DIO) function allows the user to configure a time processor unit (TPU) channel as an input or output. As an input, the channel can be read at any time or sampled at a periodic rate. As an output, the channel can be driven high or low upon command by the CPU.

The function records the 16 most recent states of the TPU channel pin. The programmer may choose one of the four following conditions to update the parameter: 1) when a transition (positive, negative, or either) occurs, 2) when the CPU makes a request to read the logical value driving the pin, 3) when the CPU makes a request to drive the pin to a specified logical value or 4) at a periodic rate.

2 Detailed Description

The DIO function allows a TPU channel pin to emulate a discrete input or output pin. As an input, the pin can be read either on command, when a transition occurs or at a periodic rate. As an output, the pin can be driven high or low on command. The DIO function can be used in the following ways:

1. Output mode: the channel is initialized with *tpu_dio_init_output*. The output state is controlled using *tpu_dio_output_high*, *tpu_dio_output_low* and *tpu_dio_output*.
2. Input mode update on transition. The pin is configured for input mode using *tpu_dio_init_input_trans*. The state of the input pin is checked using *tpu_dio_pin_history*. If both edges are selected then the returned value will be

[1] The information in this Programming Note is based on TPUPN18. It is intended to compliment the information found in that Programming Note.

0xAAAA or 0x5555 after 16 transition have been detected. If rising edge is selected then the returned value will be 0xFFFF after 16 transition. If falling edge is selected then the returned value will be 0x0000 after 16 transitions.

3. Input mode update periodically. The pin is configured for this input mode using *tpu_dio_init_input_periodic*. The pin state is update at a defined rate. The pin state is returned using *tpu_dio_pin_history.*
4. Input mode immediate update. The pin is configured for this mode using tpu_dio_init_input_immed. The pin state is updated whenever requested by the host CPU. The pin state is returned using tpu_dio_pin_history.

2.1 DIO C Level API

Rather then controlling the TPU registers directly the DIO routines in this TPU Programming Note may be used to provide a simple and easy interface. There are 9 routines for controlling the DIO function in 2 files (tpu_dio.h and tpu_dio.c). The tpu_dio.h file should be included in any files that use the routines. This files contains the function prototypes and useful #defines. Each of the routines in tpu_dio.c will be looked at in detail, the routines are:

- Initialization Function:
  - void tpu_dio_init_output(struct TPU3_tag *tpu, UINT8 channel, UINT8 priority, UINT8 level);
  - void tpu_dio_init_input_trans(struct TPU3_tag *tpu, UINT8 channel, UINT8 priority, UINT8 mode);
  - void tpu_dio_init_input_periodic(struct TPU3_tag *tpu, UINT8 channel, UINT8 priority, UINT8 tcr, UINT16 rate);
  - void tpu_dio_init_input_immed(struct TPU3_tag *tpu, UINT8 channel, UINT8 priority);
- Output Functions:
  - void tpu_dio_output_high(struct TPU3_tag *tpu, UINT8 channel);
  - void tpu_dio_output_low(struct TPU3_tag *tpu, UINT8 channel);
  - void tpu_dio_output(struct TPU3_tag *tpu, UINT8 channel, UINT8 level);
- Return Value functions:
  - UINT16 tpu_dio_input_immed(struct TPU3_tag *tpu, UINT8 channel);
  - UINT16 tpu_dio_pin_history(struct TPU3_tag *tpu, UINT8 channel);
- General TPU Functions (defined in mpc500_util.h):
  - void tpu_enable(struct TPU3_tag *tpu, UINT8 channel, UINT8 priority); (Can also be used to change the priority);
  - void tpu_disable (struct TPU3_tag *tpu, UINT8 channel);

2.1.1 void tpu_dio_init_output

This function is used to initialize a channel to run the DIO function in output mode. This function has 4 parameters:

- *tpu - This is a pointer to the TPU3 module to use. It is of type TPU3_tag which is defined in m_tpu3.h
- channel - This is the channel number of the DIO channel.

- priority - This is the priority to the channel. This parameter should be assigned a value of: TPU_PRIORITY_HIGH, TPU_PRIORITY_MIDDLE or TPU_PRIORITY_LOW. The TPU priorities are defined in mpc500_utils.h.
- level- This is the initial level of the output pin.

Care should be taken when initializing TPU channels. The TPU's behavior may be unpredictable if a channel is reconfigured while it is running. The channel should be stopped before it is configured. Setting the channel's priority to disabled does this. If the channel is currently being serviced when the priority is set to disable it will continue to service the channel until the state ends. To make sure the channel is not being service you need to wait for the longest state execution time after disabling the channel. All channels are disabled out of reset so the channels can be configured immediately from reset.

The *tpu_dio_init_output* function attempts to wait between the disabling of the channels before it starts configuring them, however the actual execution speed of the code will be depend on the specific system. If you are not configuring the channel from reset, then ideally it is best to have the function disabled before calling this function. Using the tpu_disable function in the mpc500_utils.c file can disable TPU channels. For example, disabling channel 5 is done like this:

```
tpu_disable(tpu, 5);
```

2.1.2 void tpu_dio_init_input_trans

This function is used to initialize a channel to run the DIO function in input transition mode. This function has 4 parameters:

- *tpu - This is a pointer to the TPU3 module to use. It is of type TPU3_tag which is defined in m_tpu3.h
- channel - This is the channel number of the DIO channel.
- priority - This is the priority to the channel. This parameter should be assigned a value of: TPU_PRIORITY_HIGH, TPU_PRIORITY_MIDDLE or TPU_PRIORITY_LOW. The TPU priorities are defined in mpc500_utils.h.
- mode- This defines the edge and TCR to use. The TCR should be defined as TPU_DIO_TCR1 or TPU_DIO_TCR2 and the edge should be defined as TPU_DIO_RISING_EDGE, TPU_DIO_FALLING_EDGE or TPU_DIO_BOTH_EDGES. The edge and TCR values are defined in tpu_dio.h

As described in *tpu_dio_init_output* it is best if the channel is disable and not running before this initialization routine is called.

2.1.3 void tpu_dio_init_input_periodic

This function is used to initialize a channel to run the DIO function in periodic match mode. This function has 5 parameters:

- *tpu - This is a pointer to the TPU3 module to use. It is of type TPU3_tag which is defined in m_tpu3.h
- channel - This is the channel number of the DIO channel.
- priority - This is the priority to the channel. This parameter should be assigned a value of: TPU_PRIORITY_HIGH, TPU_PRIORITY_MIDDLE or TPU_PRIORITY_LOW. The TPU priorities are defined in mpc500_utils.h.

- tcr – This is the timebase to use for the periodic rate. This should be defined as TPU_DIO_TCR1 or TPU_DIO_TCR2, the values are defined in tpu_dio.h
- rate- This defines the periodic update rate. This is in counts of the selected timebase. This value must be less than $8000.

As described in *tpu_dio_init_output* it is best if the channel is disable and not running before this initialization routine is called.

2.1.4 void tpu_dio_init_input_immed

This function is used to initialize a channel to run the DIO function in immediate update input mode. This function has 3 parameters:

- *tpu - This is a pointer to the TPU3 module to use. It is of type TPU3_tag which is defined in m_tpu3.h
- channel - This is the channel number of the DIO channel.
- priority - This is the priority to the channel. This parameter should be assigned a value of: TPU_PRIORITY_HIGH, TPU_PRIORITY_MIDDLE or TPU_PRIORITY_LOW. The TPU priorities are defined in mpc500_utils.h.

As described in *tpu_dio_init_output* it is best if the channel is disable and not running before this initialization routine is called.

2.1.5 void tpu_dio_output_high

This function is used to drive a TPU pin high. This function has 2 parameters:

- *tpu - This is a pointer to the TPU3 module to use. It is of type TPU3_tag which is defined in m_tpu3.h
- channel - This is the channel number of the DIO channel.

The channel must be initialized with *tpu_dio_init_output* before using this function.

2.1.6 void tpu_dio_output_low

This function is used to drive a TPU pin low. This function has 2 parameters:

- *tpu - This is a pointer to the TPU3 module to use. It is of type TPU3_tag which is defined in m_tpu3.h
- channel - This is the channel number of the DIO channel.

The channel must be initialized with *tpu_dio_init_output* before using this function.

2.1.7 void tpu_dio_output

This function is used to drive a TPU pin to the state defined by level. This function has 3 parameters:

- *tpu - This is a pointer to the TPU3 module to use. It is of type TPU3_tag which is defined in m_tpu3.h
- channel - This is the channel number of the DIO channel.

- level - This defines the to drive on the pin. The level should be defined as TPU_DIO_FORCE_HIGH or TPU_DIO_FORCE_LOW. The level values are defined in tpu_dio.h

The channel must be initialized with *tpu_dio_init_output* before using this function.

2.1.8 UINT16 tpu_dio_input_immed

This function is used to get the current state of a TPU pin. This function has 2 parameters:

- *tpu - This is a pointer to the TPU3 module to use. It is of type TPU3_tag which is defined in m_tpu3.h
- channel - This is the channel number of the DIO channel.

The value returned is the last 16 states of the pin. The most recent value is in the most significant bit. The channel must be initialized with *tpu_dio_init_input_immed* before using this function.

2.1.9 UINT16 tpu_dio_pin_history

This function is used to return a history of the states on a TPU pin. This function has 2 parameters:

- *tpu - This is a pointer to the TPU3 module to use. It is of type TPU3_tag which is defined in m_tpu3.h
- channel - This is the channel number of the DIO channel.

The value returned is the last 16 states of the pin. The most recent value is in the most significant bit. This function can be used with all DIO modes.

3 Discrete I/O Examples

The following examples show configuration of the discrete I/O function for each of the operating modes. Each example is a C program that shows how to configure and use the DIO interface routines.

3.1 Example 1

3.1.1 Description

This is a simple program to use a TPU pin as a discrete output. The pin is configured as an output pin with an initial low level. The pin is then toggled using once using *tpu_dio_output* and then continuously using *tpu_dio_output_high* and *tpu_dio_ouput_low.*

3.1.2 Program

```
/*************************************************************************/
/* FILE NAME: tpu_dio_example1.c              COPYRIGHT (c)  2002 */
/* VERSION: 1.0                                   All Rights Reserved     */
/*                                                                        */
/* DESCRIPTION: This sample program shows a simple example of a program   */
/* that uses the DIO API to control an output pin.                        */
```

Example 1

```
/* The program is targeted for the MPC555 but should work on any MPC500   */
/* device with a TPU. For other devices the setup routines will also need */
/* to be changed.                                                         */
/*========================================================================*/
/* HISTORY           ORIGINAL AUTHOR: Jeff Loeliger                       */
/* REV      AUTHOR      DATE       DESCRIPTION OF CHANGE                 */
/* ---   -----------  ---------    --------------------                  */
/* 1.0   J. Loeliger  30/Aug/02    Initial version of function.          */
/*************************************************************************/

#include "mpc555.h"     /* Define all of the MPC555 registers, this needs to */
                        /* changed if other MPC500 devices are used.         */
#include "mpc500.c"     /* Configuration routines for MPC555 EVB, will need */
                        /* to be changed if other hardware is used.         */
#include "mpc500_util.h"    /* Utility routines for using MPC500 devices */
#include "tpu_dio.h"    /* TPU DIO functions */

void main ()
{
     struct TPU3_tag *tpua = &TPU_A;   /* pointer for TPU routines */

     setup_mpc500(40);       /*Setup device and program PLL to 40MHz*/

    /* Initialize channel:                          */
    /*    -Use channel 3                            */
    /*    -Initial state is low                      */
    /*    -Schedule as high priority in the TPU     */
    tpu_dio_init_output(tpua, 3, TPU_PRIORITY_HIGH, TPU_DIO_PIN_LOW);

    /* toggle output pin once */
    tpu_dio_output(tpua, 3, TPU_DIO_PIN_HIGH);
    tpu_dio_output(tpua, 3, TPU_DIO_PIN_LOW);

    /* toggle output pin */
    while (1){
        tpu_dio_output_high(tpua, 3);
        tpu_dio_output_low(tpua, 3);
    }
}
```

3.2 Example 2

3.2.1 Description

This is a simple program to use a TPU pin as a discrete input. The pin is configured as an input and the pin value is updated every time there is a transition on the pin. The current pin state is stored in the MSB with the previous 15 stored in the other bits. The program loops continuously monitoring the pin state using *tpu_dio_pin_history.*

3.2.2 Program

```
/*************************************************************************/
/* FILE NAME: tpu_dio_example2.c              COPYRIGHT (c)  2002 */
/* VERSION: 1.0                                   All Rights Reserved     */
/*                                                                        */
/* DESCRIPTION: This sample program shows a simple example of a program   */
/* that uses the DIO API to record the pin state on every transition.     */
/* The program is targeted for the MPC555 but should work on any MPC500   */
/* device with a TPU. For other devices the setup routines will also need */
/* to be changed.                                                         */
/*========================================================================*/
/* HISTORY           ORIGINAL AUTHOR: Jeff Loeliger                       */
/* REV      AUTHOR      DATE       DESCRIPTION OF CHANGE                  */
/* ---   -----------  ---------    --------------------                   */
/* 1.0   J. Loeliger  30/Aug/02    Initial version of function.           */
/*************************************************************************/

#include "mpc555.h"     /* Define all of the MPC555 registers, this needs to */
                        /* changed if other MPC500 devices are used.         */
#include "mpc500.c"     /* Configuration routines for MPC555 EVB, will need */
                        /* to be changed if other hardware is used.         */
#include "mpc500_util.h"    /* Utility routines for using MPC500 devices */
#include "tpu_dio.h"    /* TPU DIO functions */

UINT16 pin;     /* current pin state in MSB */

void main ()
{
     struct TPU3_tag *tpua = &TPU_A;   /* pointer for TPU routines */

     setup_mpc500(40);       /*Setup device and program PLL to 40MHz*/
```

```
    /* Initialize channel:                          */
    /*    -Use channel 5                            */
    /*    -Update on both edges                     */
    /*    -Schedule as high priority in the TPU     */
    tpu_dio_init_input_trans(tpua, 5, TPU_PRIORITY_HIGH, TPU_DIO_BOTH_EDGES);

    /* get current pin state */
    /* when using both edges the pin value should be $5555 or $AAAA */
    while (1){
        pin = tpu_dio_pin_history(tpua, 5);
    }
}
```

3.3 Example 3

3.3.1 Description

This is a simple program to use a TPU pin as a discrete input pin that is sampled at a periodic rate. The program loops continuously monitoring the pin state using *tpu_dio_pin_history.*

3.3.2 Program

```
/*************************************************************************/
/* FILE NAME: tpu_dio_example3.c              COPYRIGHT (c)  2002 */
/* VERSION: 1.0                                   All Rights Reserved     */
/*                                                                        */
/* DESCRIPTION: This sample program shows a simple example of a program   */
/* that uses the DIO API to record the pin state at a periodic rate.      */
/* The program is targeted for the MPC555 but should work on any MPC500   */
/* device with a TPU. For other devices the setup routines will also need */
/* to be changed.                                                         */
/*========================================================================*/
/* HISTORY           ORIGINAL AUTHOR: Jeff Loeliger                       */
/* REV      AUTHOR      DATE       DESCRIPTION OF CHANGE                  */
/* ---   -----------  ---------    --------------------                   */
/* 1.0   J. Loeliger  30/Aug/02    Initial version of function.           */
/*************************************************************************/

#include "mpc555.h"     /* Define all of the MPC555 registers, this needs to */
                        /* changed if other MPC500 devices are used.         */
#include "mpc500.c"     /* Configuration routines for MPC555 EVB, will need */
                        /* to be changed if other hardware is used.         */
```

```
#include "mpc500_util.h"    /* Utility routines for using MPC500 devices */
#include "tpu_dio.h"    /* TPU DIO functions */

UINT16 pin;     /* current pin state in MSB */

void main ()
{
     struct TPU3_tag *tpua = &TPU_A;   /* pointer for TPU routines */

     setup_mpc500(40);       /*Setup device and program PLL to 40MHz*/

    /* Initialize channel:                          */
    /*    -Use channel 7                            */
    /*    -Use TCR1                                 */
    /*    -Update every $1000 counts of TCR1        */
    /*    -Schedule as high priority in the TPU     */
    tpu_dio_init_input_periodic(tpua,7,TPU_PRIORITY_HIGH,TPU_DIO_TCR1,0x1000);

    /* get pin history pin state */
    while (1){
        pin = tpu_dio_pin_history(tpua, 7);
    }
}
```

3.4 Example 4

3.4.1 Description

This is a simple program to use a TPU pin as a discrete input pin. The current value of the pin is return when the CPU requests it. The program loops continuously monitoring the pin state using *tpu_dio_pin_history.*

3.4.2 Program

```
/*************************************************************************/
/* FILE NAME: tpu_dio_example4.c              COPYRIGHT (c)  2002 */
/* VERSION: 1.0                                   All Rights Reserved     */
/*                                                                        */
/* DESCRIPTION: This sample program shows a simple example of a program   */
/* that uses the DIO API to record the pin state when requested by the    */
/* CPU.                                                                   */
/* The program is targeted for the MPC555 but should work on any MPC500   */
/* device with a TPU. For other devices the setup routines will also need */
/* to be changed.                                                         */
```

Example 4

```
/*=========================================================================*/
/* HISTORY           ORIGINAL AUTHOR: Jeff Loeliger                        */
/* REV      AUTHOR      DATE       DESCRIPTION OF CHANGE                  */
/* ---   -----------  ---------    --------------------                   */
/* 1.0   J. Loeliger  30/Aug/02    Initial version of function.           */
/*************************************************************************/

#include "mpc555.h"     /* Define all of the MPC555 registers, this needs to */
                        /* changed if other MPC500 devices are used.         */
#include "mpc500.c"     /* Configuration routines for MPC555 EVB, will need */
                        /* to be changed if other hardware is used.         */
#include "mpc500_util.h"    /* Utility routines for using MPC500 devices */
#include "tpu_dio.h"    /* TPU DIO functions */

UINT16 pin;     /* current pin state in MSB */

void main ()
{
     struct TPU3_tag *tpua = &TPU_A;   /* pointer for TPU routines */

     setup_mpc500(40);       /*Setup device and program PLL to 40MHz*/

    /* Initialize channel:                          */
    /*    -Use channel 15                            */
    /*    -Schedule as high priority in the TPU     */
    tpu_dio_init_input_immed(tpua, 15, TPU_PRIORITY_HIGH);

    /* get current pin state */
    while (1){
        pin = tpu_dio_input_immed(tpua, 5);
    }
}
```

4 Performance

Like all TPU functions, DIO function performance in an application is to some extent dependent upon the service time (latency) of other active TPU channels. This is due to the operational nature of the scheduler. The more TPU channels are active, the more performance decreases. Worst-case latency in any TPU application can be closely estimated. To analyze the performance of an application that appears to approach the limits of the TPU, use the guidelines given in the TPU reference manual and the information in the DIO state timing table below.

4.1 Function State Timing

When calculating the worst-case latency for the TPU the execution time of each state of the TPU is need. The state timings for the DIO function are shown in Table 1. The states used by the C interface functions are shown in Table 2. States S3 and S4 are entered when there is a transition on the input pins and S2 is entered when a match occurs.

Table 1. DIO Function—State Timing

State Number & Name	Max CPU Clock Cycles	RAM accesses by TPU
S1 INIT_DIO	18	4
S2 MATCH_DIO	10	3
S3 TRANS_PS_LOW_DIO	4	2
S4 TRANS_PS_HIGH_DIO	4	2
S5 LOW_PIN_REQUEST_DIO	8	2
S6 HIGH_PIN_REQUEST_DIO	8	2

Table 2. DIO Function State Usage

DIO API Function	State Uses
tpu_dio_init_output	S1
tpu_dio_init_input_trans	S1
tpu_dio_init_input_period	S1
tpu_dio_init_immed	S1
tpu_dio_output_high	S6
tpu_dio_output_low	S5
tpu_dio_output	S5 or S6
tpu_dio_input	S1
tpu_dio_pin_history	None

4.2 Function Code Size

Total TPU function code size determines what combination of functions can fit into a given ROM or emulation DPTRAM memory microcode space. The DIO function code size is:

15 µ instructions + 8 entries = 23 long words

5 Listing 1

```
/*****************************************************************************/
/* FILE NAME: tpu_dio.h                        COPYRIGHT (c)  2002 */
/* VERSION: 1.0                                    All Rights Reserved     */
/*                                                                         */
/* DESCRIPTION: This file defines the interface to the TPU DIO functions  */
/* and provides useful #defines.                                           */
```

```
/*                                                                          */
/*==========================================================================*/
/* HISTORY           ORIGINAL AUTHOR: Jeff Loeliger                         */
/* REV      AUTHOR      DATE       DESCRIPTION OF CHANGE                   */
/* ---   -----------  ---------    --------------------                    */
/* 1.0   J. Loeliger  19/Aug/02    Initial version of function.            */
/****************************************************************************/
#ifndef _TPU_DIO_H
#define _TPU_DIO_H

#include "m_common.h"
#include "m_tpu3.h"

/* Define pin state */
#define TPU_DIO_PIN_HIGH 1
#define TPU_DIO_PIN_LOW  0

/* Configuration values */
#define TPU_DIO_RISING_EDGE  0x07
#define TPU_DIO_FALLING_EDGE 0x0B
#define TPU_DIO_BOTH_EDGES   0x0F
#define TPU_DIO_TCR1         0x03
#define TPU_DIO_TCR2         0x23

/* TPU DIO function prototypes */
void tpu_dio_init_output(struct TPU3_tag *tpu, UINT8 channel, UINT8 priority,\
            UINT8 level);
void tpu_dio_init_input_trans(struct TPU3_tag *tpu, UINT8 channel, \
            UINT8 priority, UINT8 mode);
void tpu_dio_init_input_periodic(struct TPU3_tag *tpu, UINT8 channel, \
            UINT8 priority, UINT8 tcr, UINT16 rate);
void tpu_dio_init_input_immed(struct TPU3_tag *tpu, UINT8 channel, UINT8 priority);

void tpu_dio_output_high(struct TPU3_tag *tpu, UINT8 channel);
void tpu_dio_output_low(struct TPU3_tag *tpu, UINT8 channel);
void tpu_dio_output(struct TPU3_tag *tpu, UINT8 channel, UINT8 level);
UINT16 tpu_dio_input_immed(struct TPU3_tag *tpu, UINT8 channel);
UINT16 tpu_dio_pin_history(struct TPU3_tag *tpu, UINT8 channel);

/* The #defines below here are normally only used internally by the */
/* DIO API functions and should not be needed in user programs.     */
```

```
/* Define HSR values */
#define TPU_DIO_INIT       0x3
#define TPU_DIO_FORCE_LOW  0x2
#define TPU_DIO_FORCE_HIGH 0x3

/* Define HSQ values */
#define TPU_DIO_TRANS_MODE 0x0
#define TPU_DIO_MATCH_MODE 0x1
#define TPU_DIO_IMMED_MODE 0x2

/* Define parameter RAM locations */
#define TPU_DIO_CHANNEL_CONTROL 0
#define TPU_DIO_PIN_LEVEL       1
#define TPU_DIO_MATCH_RATE      2

#endif /* ifndef _TPU_DIO_H  */
```

6 Listing 2

```
/*****************************************************************************/
/* FILE NAME: tpu_dio.c                        COPYRIGHT (c)  2002 */
/* VERSION: 1.0                                    All Rights Reserved     */
/*                                                                           */
/* DESCRIPTION: This file contains the TPU DIO functions. These functions */
/* allow you to completely control TPU channels running the DIO function. */
/* They provide a simple interface requiring the minimum amount of        */
/* configuration by the user.                                              */
/*===========================================================================*/
/* HISTORY           ORIGINAL AUTHOR: Jeff Loeliger                        */
/* REV      AUTHOR      DATE       DESCRIPTION OF CHANGE                  */
/* ---   -----------  ---------    --------------------                   */
/* 1.0   J. Loeliger  19/Aug/02    Initial version of function.           */
/*****************************************************************************/
#include "tpu_dio.h"
#include "mpc500_util.h"

/*****************************************************************************
FUNCTION      : tpu_dio_init_output
```

```
PURPOSE       : To initialize a channel to run the DIO function in output mode.
INPUTS NOTES  : This function has 4 parameters:
                  *tpu - This is a pointer to the TPU3 module to use. It is of
                         type TPU3_tag which is defined in m_tpu3.h
                  channel - This is the channel number of the primary QDEC
                            channel.
                  priority - This is the priority to assign to both channels.
                             This parameter should be assigned a value of:
                             TPU_PRIORITY_HIGH, TPU_PRIORITY_MIDDLE or
                             TPU_PRIORITY_LOW.
                  level - This is the initial level of the pin.
RETURNS NOTES : none
WARNING       : The channels must be stopped before it is reconfigured. The
                function disables the channels but if they were currently
                being serviced it would continue. The delay for assigning the
                pram pointer may to enough but depends on system loading.
*************************************************************************/
void tpu_dio_init_output(struct TPU3_tag *tpu, UINT8 channel, UINT8 priority,\
            UINT8 level)
{
    /* disable channel so it can be configured safely */
    tpu_disable( tpu, channel);

    /* select DIO function for channel */
    tpu_func( tpu, channel, TPU_FUNCTION_DIO);

    /* Initialize channel */
    if(level == TPU_DIO_PIN_HIGH)
        tpu_hsr(tpu, channel, TPU_DIO_FORCE_HIGH);
    else
        tpu_hsr(tpu, channel, TPU_DIO_FORCE_LOW);

    /* Enable channel by assigning a priority. */
    tpu_enable(tpu, channel, priority);
}

/*************************************************************************
FUNCTION      : tpu_dio_output_high
PURPOSE       : To drive a channel running DIO to a high level.
INPUTS NOTES  : This function has 2 parameters:
                  *tpu - This is a pointer to the TPU3 module to use. It is of
```

```
                        type TPU3_tag which is defined in m_tpu3.h
                channel - This is the channel number of the primary QDEC
                          channel.
RETURNS NOTES : none
*****************************************************************************/
void tpu_dio_output_high(struct TPU3_tag *tpu, UINT8 channel)
{
    tpu_hsr(tpu, channel, TPU_DIO_FORCE_HIGH);
}

/*****************************************************************************
FUNCTION      : tpu_dio_output_low
PURPOSE       : To drive a channel running DIO to a low level.
INPUTS NOTES  : This function has 2 parameters:
                *tpu - This is a pointer to the TPU3 module to use. It is of
                        type TPU3_tag which is defined in m_tpu3.h
                channel - This is the channel number of the primary QDEC
                          channel.
RETURNS NOTES : none
*****************************************************************************/
void tpu_dio_output_low(struct TPU3_tag *tpu, UINT8 channel)
{
    tpu_hsr(tpu, channel, TPU_DIO_FORCE_LOW);
}

/*****************************************************************************
FUNCTION      : tpu_dio_output
PURPOSE       : To drive a channel running DIO to a requested level.
INPUTS NOTES  : This function has 3 parameters:
                *tpu - This is a pointer to the TPU3 module to use. It is of
                        type TPU3_tag which is defined in m_tpu3.h
                channel - This is the channel number of the primary QDEC
                          channel.
                level - This is the requested level of the pin.
RETURNS NOTES : none
*****************************************************************************/
void tpu_dio_output(struct TPU3_tag *tpu, UINT8 channel, UINT8 level)
{
    if(level == TPU_DIO_PIN_HIGH)
        tpu_hsr(tpu, channel, TPU_DIO_FORCE_HIGH);
    else
```

```
        tpu_hsr(tpu, channel, TPU_DIO_FORCE_LOW);
}

/*****************************************************************************
FUNCTION      : tpu_dio_init_input_trans
PURPOSE       : To initialize one channel for capture on input transisiton.
INPUTS NOTES  : This function has 4 parameters:
                 *tpu - This is a pointer to the TPU3 module to use. It is of
                        type TPU3_tag which is defined in m_tpu3.h
                 channel - This is the channel number of the DIO channel.
                 priority - This is the priority to assign to both channels
                            This parameter should be assigned a value of:
                            TPU_PRIORITY_HIGH, TPU_PRIORITY_MIDDLE or
                            TPU_PRIORITY_LOW.
                 mode - This defines the edge and TCR to use.
RETURNS NOTES : none
WARNING       : The channel must be stopped before it is reconfigured. The
                function disables the channel but if it is currently
                being serviced it would continue. The delay for assigning the
                pram pointer may to enough but depends on system loading.
*****************************************************************************/
void tpu_dio_init_input_trans(struct TPU3_tag *tpu, UINT8 channel, \
            UINT8 priority, UINT8 mode)
{
    struct TPU_param_tag *pram;

    /* disable channel so it can be configured safely */
    tpu_disable( tpu, channel);

    /* select DIO function for channel */
    tpu_func( tpu, channel, TPU_FUNCTION_DIO);

    /* Initialize parameter RAM channel control */
    tpu->PARM.R[channel][TPU_DIO_CHANNEL_CONTROL] = mode;

    /* Configure the channel for transition mode */
    tpu_hsq(tpu, channel, TPU_DIO_TRANS_MODE);

    /* Initialize the channel */
    tpu_hsr(tpu, channel, TPU_DIO_INIT);
```

```
    /* Enable the channel by assigning a priority to it. */
    tpu_enable(tpu, channel, priority);
}

/*****************************************************************************
FUNCTION      : tpu_dio_init_input_periodic
PURPOSE       : To initialize one channel as periodically updated input
                counter.
INPUTS NOTES  : This function has 5 parameters:
                 *tpu - This is a pointer to the TPU3 module to use. It is of
                        type TPU3_tag which is defined in m_tpu3.h
                 channel - This is the channel number of the DIO channel.
                 priority - This is the priority to assign to both channels
                            This parameter should be assigned a value of:
                            TPU_PRIORITY_HIGH, TPU_PRIORITY_MIDDLE or
                            TPU_PRIORITY_LOW.
                 tcr - This is the timebase to reference.
                 rate - This is the periodic update rate.
RETURNS NOTES : none
WARNING       : The channel must be stopped before it is reconfigured. The
                function disables the channel but if it is currently
                being serviced it would continue. The delay for assigning the
                pram pointer may to enough but depends on system loading.
*****************************************************************************/
void tpu_dio_init_input_periodic(struct TPU3_tag *tpu, UINT8 channel, \
            UINT8 priority, UINT8 tcr, UINT16 rate)
{
    struct TPU_param_tag *pram;

    /* disable channel so it can be configured safely */
    tpu_disable( tpu, channel);

    /* select DIO function for channel */
    tpu_func( tpu, channel, TPU_FUNCTION_DIO);

    /* Initialize parameter RAM channel control */
    tpu->PARM.R[channel][TPU_DIO_CHANNEL_CONTROL] = tcr;

    /* Initialize parameter RAM with update rate */
    tpu->PARM.R[channel][TPU_DIO_MATCH_RATE] = rate;
```

```
    /* Configure the channel for match mode */
    tpu_hsq(tpu, channel, TPU_DIO_MATCH_MODE);

    /* Initialize the channel */
    tpu_hsr(tpu, channel, TPU_DIO_INIT);

    /* Enable the channel by assigning a priority to it. */
    tpu_enable(tpu, channel, priority);
}

/*****************************************************************************
FUNCTION      : tpu_dio_init_input_immed
PURPOSE       : To initialize one channel as periodicly updated input
                counter.
INPUTS NOTES  : This function has 3 parameters:
                 *tpu - This is a pointer to the TPU3 module to use. It is of
                        type TPU3_tag which is defined in m_tpu3.h
                channel - This is the channel number of the DIO channel.
                priority - This is the priority to assign to both channels
                           This parameter should be assigned a value of:
                           TPU_PRIORITY_HIGH, TPU_PRIORITY_MIDDLE or
                           TPU_PRIORITY_LOW.
RETURNS NOTES : none
WARNING       : The channel must be stopped before it is reconfigured. The
                function disables the channel but if it is currently
                being serviced it would continue. The delay for assigning the
                pram pointer may to enough but depends on system loading.
*****************************************************************************/
void tpu_dio_init_input_immed(struct TPU3_tag *tpu, UINT8 channel, UINT8 priority)
{
    struct TPU_param_tag *pram;

    /* disable channel so it can be configured safely */
    tpu_disable( tpu, channel);

    /* select DIO function for channel */
    tpu_func( tpu, channel, TPU_FUNCTION_DIO);

    /* Initialize parameter RAM channel control */
    tpu->PARM.R[channel][TPU_DIO_CHANNEL_CONTROL] = 0x3;
```

```
    /* Configure the channel for transition mode and use HSR later*/
    tpu_hsq(tpu, channel, TPU_DIO_TRANS_MODE);

    /* Initialize the channel */
    tpu_hsr(tpu, channel, TPU_DIO_INIT);

    /* Enable the channel by assigning a priority to it. */
    tpu_enable(tpu, channel, priority);
}

/*****************************************************************************
FUNCTION      : tpu_dio_input_immed
PURPOSE       : This function returns the current pin state.
INPUTS NOTES  : This function has 2 parameters:
                *tpu - This is a pointer to the TPU3 module to use. It is of
                       type TPU3_tag which is defined in m_tpu3.h
                channel - This is the channel number of the primary QDEC
                          channel. The next channel is used as the secondary.
RETURNS NOTES : The pin state.
*****************************************************************************/
UINT16 tpu_dio_input_immed(struct TPU3_tag *tpu, UINT8 channel)
{
    /* Configure the channel for transition mode and use HSR later*/
    tpu_hsq(tpu, channel, TPU_DIO_IMMED_MODE);

    /* Initialize the channel */
    tpu_hsr(tpu, channel, TPU_DIO_INIT);

    /* wait for value to be updated */
    tpu_ready(tpu, channel);

    return (tpu->PARM.R[channel][TPU_DIO_PIN_LEVEL]);
}

/*****************************************************************************
FUNCTION      : tpu_dio_pin_history
PURPOSE       : This function returns the pin history.
INPUTS NOTES  : This function has 2 parameters:
                *tpu - This is a pointer to the TPU3 module to use. It is of
                       type TPU3_tag which is defined in m_tpu3.h
                channel - This is the channel number of the primary QDEC
```

```
                          channel. The next channel is used as the secondary.
RETURNS NOTES : The current pin history.
*******************************************************************************/
UINT16 tpu_dio_pin_history(struct TPU3_tag *tpu, UINT8 channel)
{
     return (tpu->PARM.R[channel][TPU_DIO_PIN_LEVEL]);
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor