```
#define TESTAPP_GEN

/* $Id: xgpio_tapp_example.c,v 1.1.2.1 2009/11/25 07:38:15 svemula Exp $ */
/******************************************************************************
*
* (c) Copyright 2005-2009 Xilinx, Inc. All rights reserved.
*
* This file contains confidential and proprietary information of Xilinx, Inc.
* and is protected under U.S. and international copyright and other
* intellectual property laws.
*
* DISCLAIMER
* This disclaimer is not a license and does not grant any rights to the
* materials distributed herewith. Except as otherwise provided in a valid
* license issued to you by Xilinx, and to the maximum extent permitted by
* applicable law: (1) THESE MATERIALS ARE MADE AVAILABLE "AS IS" AND WITH ALL
* FAULTS, AND XILINX HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS,
* IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF
* MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE;
* and (2) Xilinx shall not be liable (whether in contract or tort, including
* negligence, or under any other theory of liability) for any loss or damage
* of any kind or nature related to, arising under or in connection with these
* materials, including for any direct, or any indirect, special, incidental,
* or consequential loss or damage (including loss of data, profits, goodwill,
* or any type of loss or damage suffered as a result of any action brought by
* a third party) even if such damage or loss was reasonably foreseeable or
* Xilinx had been advised of the possibility of the same.
*
* CRITICAL APPLICATIONS
* Xilinx products are not designed or intended to be fail-safe, or for use in
* any application requiring fail-safe performance, such as life-support or
* safety devices or systems, Class III medical devices, nuclear facilities,
* applications related to the deployment of airbags, or any other applications
* that could lead to death, personal injury, or severe property or
* environmental damage (individually and collectively, "Critical
* Applications"). Customer assumes the sole risk and liability of any use of
* Xilinx products in Critical Applications, subject only to applicable laws
* and regulations governing limitations on product liability.
*
* THIS COPYRIGHT NOTICE AND DISCLAIMER MUST BE RETAINED AS PART OF THIS FILE
* AT ALL TIMES.
*
******************************************************************************/

/*****************************************************************************/
/**
* @file xgpio_tapp_example.c
*
* This file contains a example for using GPIO hardware and driver.
* This example assumes that there is a UART Device or STDIO Device in the
* hardware system.
*
* This example can be run on the Xilinx ML300 board using the Prototype Pins &
* LEDs of the board connected to the GPIO and the Push Buttons connected.
*
* @note
*
* None
*
* <pre>
* MODIFICATION HISTORY:
*
* Ver   Who  Date         Changes
* ----- ---- -------- -----------------------------------------------
* 1.00a sv   04/15/05 Initial release for TestApp integration.
* 3.00a sv   11/21/09 Updated to use HAL Processor APIs.
* </pre>
*
******************************************************************************/

/***************************** Include Files *********************************/

#include "xparameters.h"
#include "xgpio.h"
#include "stdio.h"
#include "xstatus.h"
```

```
/************************** Constant Definitions *****************************/

/*
 * The following constant is used to wait after an LED is turned on to make
 * sure that it is visible to the human eye.  This constant might need to be
 * tuned for faster or slower processor speeds.
 */
#define LED_DELAY         1000000

/* following constant is used to determine which channel of the GPIO is
 * used if there are 2 channels supported in the GPIO.
 */
#define LED_CHANNEL 1

#define LED_MAX_BLINK   0x1     /* Number of times the LED Blinks */

#define GPIO_BITWIDTH   16      /* This is the width of the GPIO */

#define printf xil_printf       /* A smaller footprint printf */

/*
 * The following constants map to the XPAR parameters created in the
 * xparameters.h file. They are defined here such that a user can easily
 * change all the needed parameters in one place.
 */
#ifndef TESTAPP_GEN
#define GPIO_OUTPUT_DEVICE_ID   XPAR_LEDS_4BIT_DEVICE_ID
#define GPIO_INPUT_DEVICE_ID    XPAR_LEDS_4BIT_DEVICE_ID
#endif /* TESTAPP_GEN */

/**************************** Type Definitions *******************************/


/***************** Macros (Inline Functions) Definitions *********************/


/************************** Function Prototypes ******************************/

int GpioOutputExample(u16 DeviceId, u32 GpioWidth);

int GpioInputExample(u16 DeviceId, u32 *DataRead);

void GpioDriverHandler(void *CallBackRef);


/************************** Variable Definitions *****************************/

/*
 * The following are declared globally so they are zeroed and so they are
 * easily accessible from a debugger
 */
XGpio GpioOutput; /* The driver instance for GPIO Device configured as O/P */
XGpio GpioInput;  /* The driver instance for GPIO Device configured as I/P */

/*****************************************************************************/
/**
* Main function to call the example.This function is not included if the
* example is generated from the TestAppGen test tool.
*
* @param        None
*
* @return       XST_SUCCESS if successful, XST_FAILURE if unsuccessful
*
* @note         None
*
******************************************************************************/
#ifndef TESTAPP_GEN
int main(void)
{
        int Status;
        u32 InputData;

        Status = GpioOutputExample(GPIO_OUTPUT_DEVICE_ID, GPIO_BITWIDTH);
        if (Status != XST_SUCCESS) {
                  return XST_FAILURE;
        }
```

```
        Status = GpioInputExample(GPIO_INPUT_DEVICE_ID, &InputData);
        if (Status != XST_SUCCESS) {
                return XST_FAILURE;
        }

        printf("Data read from GPIO Input is  0x%x \n", (int)InputData);

        return XST_SUCCESS;
}
#endif


/*****************************************************************************/
/**
*
* This function does a minimal test on the GPIO device configured as OUTPUT
* and driver as a  example.
*
*
* @param        DeviceId is the XPAR_<GPIO_instance>_DEVICE_ID value from
*               xparameters.h
* @param        GpioWidth is the width of the GPIO
*
* @return       XST_SUCCESS if successful, XST_FAILURE if unsuccessful
*
* @note         None
*
******************************************************************************/
int GpioOutputExample(u16 DeviceId, u32 GpioWidth)
{
        u32 Data;
        volatile int Delay;
        u32 LedBit;
        u32 LedLoop;
        int Status;

        /*
         * Initialize the GPIO driver so that it's ready to use,
         * specify the device ID that is generated in xparameters.h
         */
         Status = XGpio_Initialize(&GpioOutput, DeviceId);
         if (Status != XST_SUCCESS)  {
                 return XST_FAILURE;
         }


         /*
          * Set the direction for all signals to be outputs
          */
         XGpio_SetDataDirection(&GpioOutput, LED_CHANNEL, 0x0);

         /*
          * Set the GPIO outputs to low
          */
         XGpio_DiscreteWrite(&GpioOutput, LED_CHANNEL, 0x0);

         for (LedBit = 0x0; LedBit < GpioWidth; LedBit++)  {

                for (LedLoop = 0; LedLoop < LED_MAX_BLINK; LedLoop++) {

                        /*
                         * Set the GPIO Output to High
                         */
                        XGpio_DiscreteWrite(&GpioOutput, LED_CHANNEL,
                                                1 << LedBit);

#ifndef __SIM__
                        /*
                         * Wait a small amount of time so the LED is visible
                         */
                        for (Delay = 0; Delay < LED_DELAY; Delay++);

#endif
                        /*
                         * Clear the GPIO Output
                         */
                        XGpio_DiscreteClear(&GpioOutput, LED_CHANNEL,
```

```
                                        1 << LedBit);


#ifndef __SIM__
                        /*
                         * Wait a small amount of time so the LED is visible
                         */
                        for (Delay = 0; Delay < LED_DELAY; Delay++);
#endif

                }

        }

        return XST_SUCCESS;

}


/*****************************************************************************/
/**
*
* This function  performs a test on the GPIO driver/device with the GPIO
* configured as INPUT
*
* @param        DeviceId is the XPAR_<GPIO_instance>_DEVICE_ID value from
*                       xparameters.h
* @param        DataRead is the pointer where the data read from GPIO Input is
*                       returned
*
* @return       XST_SUCCESS if the Test is successful, otherwise XST_FAILURE
*
* @note   None.
*
******************************************************************************/
int GpioInputExample(u16 DeviceId, u32 *DataRead)
{
        int Status;

        /*
         * Initialize the GPIO driver so that it's ready to use,
         * specify the device ID that is generated in xparameters.h
         */
        Status = XGpio_Initialize(&GpioInput, DeviceId);
        if (Status != XST_SUCCESS) {
                return XST_FAILURE;
        }

        /*
         * Set the direction for all signals to be inputs
         */
        XGpio_SetDataDirection(&GpioInput, LED_CHANNEL, 0xFFFFFFFF);

        /*
         * Read the state of the data so that it can be  verified
         */
        *DataRead = XGpio_DiscreteRead(&GpioInput, LED_CHANNEL);

        return XST_SUCCESS;

}
```