```c
#define TESTAPP_GEN


/* $Id: xintc_tapp_example.c,v 1.1.2.1 2010/09/17 05:26:04 svemula Exp $ */
/******************************************************************************
*
* (c) Copyright 2002-2009 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 xintc_tapp_example.c
*
* This file contains a self test example using the Interrupt Controller driver
* (XIntc) and hardware device. Please reference other device driver examples to
* see more examples of how the Intc and interrupts can be used by a software
* application.
*
* This example shows the use of the Interrupt Controller both with a PowerPC405
* and MicroBlaze processor.
*
* The TestApp Gen utility uses this file to perform the self test and setup
* of Intc for interrupts.
*
* @note
*
* None
*
* <pre>
*
* MODIFICATION HISTORY:
*
* Ver   Who  Date        Changes
* ----- ---- -------- -------------------------------------------------------
* 1.00a sv   06/29/05  Created for Test App Integration
* 1.00c sn   05/09/06  Added Interrupt Setup Function
* 2.00a ktn  10/20/09  Updated to use HAL Processor APIs and minor changes as
*                      per coding guidelines.
* </pre>
******************************************************************************/
```

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

#include "xparameters.h"
#include "xstatus.h"
#include "xintc.h"
#include "xil_exception.h"


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

/*
 * 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. This definition is not
 * included if the example is generated from the TestAppGen test tool.
 */
#ifndef TESTAPP_GEN
#define INTC_DEVICE_ID            XPAR_INTC_0_DEVICE_ID
#endif

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


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


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

int IntcSelfTestExample(u16 DeviceId);
int IntcInterruptSetup(XIntc *IntcInstancePtr, u16 DeviceId);

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

static XIntc InterruptController; /* Instance of the Interrupt Controller */


/*****************************************************************************/
/**
*
* This is the main function for the Interrupt Controller example. This
* function is not included if the example is generated from the TestAppGen test
* tool.
*
* @param        None.
*
* @return       XST_SUCCESS to indicate success, otherwise XST_FAILURE.
*
* @note         None.
*
******************************************************************************/
#ifndef TESTAPP_GEN
int main(void)
   {
   int Status;

/* Run the Intc example , specify the Device ID generated in xparameters.h.  */
   Status = IntcSelfTestExample(INTC_DEVICE_ID);
   if (Status != XST_SUCCESS)
      { return XST_FAILURE; }

   return XST_SUCCESS;
   }
#endif

/*****************************************************************************/
/**
* This function runs a self-test on the driver/device. This is a destructive
* test. This function is an example of how to use the interrupt controller
* driver component (XIntc) and the hardware device.  This function is designed
* to work without any hardware devices to cause interrupts.  It may not return
* if the interrupt controller is not properly connected to the processor in
* either software or hardware.
* This function relies on the fact that the interrupt controller hardware
* has come out of the reset state such that it will allow interrupts to be
* simulated by the software.
* @param        DeviceId is device ID of the Interrupt Controller Device,
*               typically XPAR_<INTC_instance>_DEVICE_ID value from
*               xparameters.h.
* @return       XST_SUCCESS to indicate success, otherwise XST_FAILURE.
* @note         None.
******************************************************************************/
int IntcSelfTestExample(u16 DeviceId)
   {
   int Status;

/* Initialize the interrupt controller driver so that it is ready to use. */
   Status = XIntc_Initialize(&InterruptController, DeviceId);
   if (Status != XST_SUCCESS)
      { return XST_FAILURE; }

/* Perform a self-test to ensure that the hardware was built correctly.  */
   Status = XIntc_SelfTest(&InterruptController);
   if (Status != XST_SUCCESS)
      { return XST_FAILURE; }

   return XST_SUCCESS;
}

/******************************************************************************/
/**
* This function is used by the TestAppGen generated application to setup
* the interrupt controller.
* @param        IntcInstancePtr is the reference to the Interrupt Controller
*               instance.
* @param        DeviceId is device ID of the Interrupt Controller Device,
*               typically XPAR_<INTC_instance>_DEVICE_ID value from
*               xparameters.h.
* @return       XST_SUCCESS to indicate success, otherwise XST_FAILURE.
* @note         None.
******************************************************************************/
int IntcInterruptSetup(XIntc *IntcInstancePtr, u16 DeviceId)
   {
   int Status;

/* Initialize the interrupt controller driver so that it is ready to use.  */
   Status = XIntc_Initialize(IntcInstancePtr, DeviceId);
   if (Status != XST_SUCCESS)
      { return XST_FAILURE; }

/* Perform a self-test to ensure that the hardware was built  correctly.  */
   Status = XIntc_SelfTest(IntcInstancePtr);
   if (Status != XST_SUCCESS)
      { return XST_FAILURE; }

/* Initialize the exception table. */
   Xil_ExceptionInit();

/* Register the interrupt controller handler with the exception table. */
   Xil_ExceptionRegisterHandler(XIL_EXCEPTION_ID_INT, (Xil_ExceptionHandler)XIntc_DeviceInterruptHandler, (vo
id*) 0);

/* Enable exceptions.  */
   Xil_ExceptionEnable();

/* Start the interrupt controller such that interrupts are enabled for all devices that cause interrupts.  */
   Status = XIntc_Start(IntcInstancePtr, XIN_REAL_MODE);
   if (Status != XST_SUCCESS)
      { return XST_FAILURE; }

   return XST_SUCCESS;
   }
```