```
/*****************************************************************************
* Filename:
C:\Users\superman\Desktop\byu-ee-427-labs\PIT\MyProcessorIPLib/drivers/pit_v1_00_a/src/pit
.h
* Version:           1.00.a
* Description:       pit Driver Header File
* Date:              Wed Nov 02 16:29:39 2016 (by Create and Import Peripheral Wizard)
*
*
* Taylor Cowley and Andrew Okazaki.
* Note: This file was auto-generated by the EDK and unchanged by us
*
*
*****************************************************************************/

#ifndef PIT_H
#define PIT_H

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

#include "xbasic_types.h"
#include "xstatus.h"
#include "xil_io.h"

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


/**
 * User Logic Slave Space Offsets
 * -- SLV_REG0 : user logic slave module register 0
 * -- SLV_REG1 : user logic slave module register 1
 * -- SLV_REG2 : user logic slave module register 2
 * -- SLV_REG3 : user logic slave module register 3
 */
#define PIT_USER_SLV_SPACE_OFFSET (0x00000000)
#define PIT_SLV_REG0_OFFSET (PIT_USER_SLV_SPACE_OFFSET + 0x00000000)
#define PIT_SLV_REG1_OFFSET (PIT_USER_SLV_SPACE_OFFSET + 0x00000004)
#define PIT_SLV_REG2_OFFSET (PIT_USER_SLV_SPACE_OFFSET + 0x00000008)
#define PIT_SLV_REG3_OFFSET (PIT_USER_SLV_SPACE_OFFSET + 0x0000000C)

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


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

/**
 *
 * Write a value to a PIT register. A 32 bit write is performed.
 * If the component is implemented in a smaller width, only the least
 * significant data is written.
 *
 * @param   BaseAddress is the base address of the PIT device.
 * @param   RegOffset is the register offset from the base to write to.
 * @param   Data is the data written to the register.
 *
 * @return  None.
 *
 * @note
```

```
 * C-style signature:
 *  void PIT_mWriteReg(Xuint32 BaseAddress, unsigned RegOffset, Xuint32 Data)
 *
 */
#define PIT_mWriteReg(BaseAddress, RegOffset, Data) \
    Xil_Out32((BaseAddress) + (RegOffset), (Xuint32)(Data))

/**
 *
 * Read a value from a PIT register. A 32 bit read is performed.
 * If the component is implemented in a smaller width, only the least
 * significant data is read from the register. The most significant data
 * will be read as 0.
 *
 * @param   BaseAddress is the base address of the PIT device.
 * @param   RegOffset is the register offset from the base to write to.
 *
 * @return  Data is the data from the register.
 *
 * @note
 * C-style signature:
 *  Xuint32 PIT_mReadReg(Xuint32 BaseAddress, unsigned RegOffset)
 *
 */
#define PIT_mReadReg(BaseAddress, RegOffset) \
    Xil_In32((BaseAddress) + (RegOffset))


/**
 *
 * Write/Read 32 bit value to/from PIT user logic slave registers.
 *
 * @param   BaseAddress is the base address of the PIT device.
 * @param   RegOffset is the offset from the slave register to write to or read from.
 * @param   Value is the data written to the register.
 *
 * @return  Data is the data from the user logic slave register.
 *
 * @note
 * C-style signature:
 *  void PIT_mWriteSlaveRegn(Xuint32 BaseAddress, unsigned RegOffset, Xuint32 Value)
 *  Xuint32 PIT_mReadSlaveRegn(Xuint32 BaseAddress, unsigned RegOffset)
 *
 */
#define PIT_mWriteSlaveReg0(BaseAddress, RegOffset, Value) \
    Xil_Out32((BaseAddress) + (PIT_SLV_REG0_OFFSET) + (RegOffset), (Xuint32)(Value))
#define PIT_mWriteSlaveReg1(BaseAddress, RegOffset, Value) \
    Xil_Out32((BaseAddress) + (PIT_SLV_REG1_OFFSET) + (RegOffset), (Xuint32)(Value))
#define PIT_mWriteSlaveReg2(BaseAddress, RegOffset, Value) \
    Xil_Out32((BaseAddress) + (PIT_SLV_REG2_OFFSET) + (RegOffset), (Xuint32)(Value))
#define PIT_mWriteSlaveReg3(BaseAddress, RegOffset, Value) \
    Xil_Out32((BaseAddress) + (PIT_SLV_REG3_OFFSET) + (RegOffset), (Xuint32)(Value))

#define PIT_mReadSlaveReg0(BaseAddress, RegOffset) \
    Xil_In32((BaseAddress) + (PIT_SLV_REG0_OFFSET) + (RegOffset))
#define PIT_mReadSlaveReg1(BaseAddress, RegOffset) \
    Xil_In32((BaseAddress) + (PIT_SLV_REG1_OFFSET) + (RegOffset))
#define PIT_mReadSlaveReg2(BaseAddress, RegOffset) \
```

```
    Xil_In32((BaseAddress) + (PIT_SLV_REG2_OFFSET) + (RegOffset))
#define PIT_mReadSlaveReg3(BaseAddress, RegOffset) \
    Xil_In32((BaseAddress) + (PIT_SLV_REG3_OFFSET) + (RegOffset))

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


/**
 *
 * Run a self-test on the driver/device. Note this may be a destructive test if
 * resets of the device are performed.
 *
 * If the hardware system is not built correctly, this function may never
 * return to the caller.
 *
 * @param   baseaddr_p is the base address of the PIT instance to be worked on.
 *
 * @return
 *
 *    - XST_SUCCESS   if all self-test code passed
 *    - XST_FAILURE   if any self-test code failed
 *
 * @note    Caching must be turned off for this function to work.
 * @note    Self test may fail if data memory and device are not on the same bus.
 *
 */
XStatus PIT_SelfTest(void * baseaddr_p);
/**
*  Defines the number of registers available for read and write*/
#define TEST_AXI_LITE_USER_NUM_REG 4


#endif /** PIT_H */
```