```
/* ctifpgalib.h - CTI FPGA Application Programming Interface */

/*
Copyright (c) 2011, CTI, Connect Tech Inc. All Rights Reserved.

THIS IS THE UNPUBLISHED PROPRIETARY SOURCE CODE OF CONNECT TECH INC.
The copyright notice above does not evidence any actual or intended
publication of such source code.

This module contains Proprietary Information of Connect Tech Inc
and should be treated as Confidential.
*/


#ifndef _ctifpgalib_H
#define _ctifpgalib_H


/*****************************************************************************
 *  Error Status                                                             *
 *****************************************************************************/
#define TRUE                            1
#define FALSE                           0
#define CTI_STATUS_OK                   0x0000
#define CTI_STATUS_ERROR                0x0001
#define CTI_STATUS_INVALIDPARM          0x0002
#define CTI_STATUS_TIMEOUT              0x0003
#define CTI_STATUS_IOCTL_FAILED         0x0004
#define CTI_STATUS_INVALID_ARG1         0x10001
#define CTI_STATUS_INVALID_ARG2         0x10002
#define CTI_STATUS_INVALID_ARG3         0x10003
#define CTI_STATUS_INVALID_ARG4         0x10004
#define CTI_STATUS_INVALID_ARG5         0x10005



/*****************************************************************************
 *  Data types
 *****************************************************************************/

#if defined(_WIN32) || defined(_WIN64)
    #define CTI_MSWINDOWS
#else
       #define CTI_LINUX
#endif

#ifdef CTI_MSWINDOWS
typedef signed char           S8;
typedef unsigned char         U8;
typedef signed short          S16;
typedef unsigned short        U16;
typedef signed int           S32;
typedef unsigned int         U32;
typedef signed _int64         S64;
typedef unsigned _int64       U64;
#endif

#ifdef CTI_LINUX
```

```
typedef signed char           S8;
typedef unsigned char         U8;
typedef signed short          S16;
typedef unsigned short        U16;
typedef signed int           S32;
typedef unsigned int         U32;
typedef signed long long    S64;
typedef unsigned long long  U64;
#endif

#ifdef __cplusplus
#define DEFCPP extern "C"
#else
#define DEFCPP
#endif
/************************************************************************
 *  Structures
 ************************************************************************/
typedef union _CtiUfpgaBoard
{
       HANDLE hBoard; // handle to board for Windows
       int fd; // file descriptor for non Windows
}CtiUfpgaBoard;

/************************************************************************
 *  Functions
 ************************************************************************/
/************************************************************************
* CTIConnectBoard
*
* This routine is used to connect to CTI FPGA baord.
* Arguments:
*                    pBoardInfo: pointer to CtiUfpgaBoard
*                    Index: index of the board
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
DEFCPP int _stdcall CTIConnectBoard (CtiUfpgaBoard *pBoardInfo, UCHAR Index);

/************************************************************************
* CTIDisconnectBoard
*
* This routine is used to disconnect from CTI FPGA baord.
* Arguments
*                    pBoardInfo: pointer to CtiUfpgaBoard
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
DEFCPP int _stdcall CTIDisconnectBoard (CtiUfpgaBoard *pBoardInfo);

/************************************************************************
* CTIFindBoard
*
* This routine returns all boards information defined by UFPGA_ALL_BOARDINFO
* Arguments:
*                    pAllBoardInfo: pointer to UFPGA_ALL_BOARDINFO
```

```
*
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
DEFCPP int _stdcall CTIFindBoard (UFPGA_ALL_BOARDINFO *pAllBoardInfo);

/***
**/
DEFCPP int _stdcall CTIGetBoardIndex(unsigned int VID, unsigned int DID, int instance,
int* pIndex);

/*****************************************************************************
* CTIFPGARdByte
*
* This routine reads a byte from CTI FPGA baord's IO location.
* Arguments
*                    pBoardInfo: pointer to CtiUfpgaBoard
*                    bar: PCI BAR index (0 to 5)
*                    offset: offset to BAR memory
*                    data: pointer to data byte
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
DEFCPP int _stdcall CTIFPGARdByte(CtiUfpgaBoard *pBoardInfo, int bar, int offset,
unsigned char* data);

/*****************************************************************************
* CTIFPGARdWord
*
* This routine reads a WORD from CTI FPGA baord's IO location.
* Arguments
*                    pBoardInfo: pointer to CtiUfpgaBoard
*                    bar: PCI BAR index (0 to 5)
*                    offset: offset to BAR memory
*                    data: pointer to data WORD
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
DEFCPP int _stdcall CTIFPGARdWord(CtiUfpgaBoard *pBoardInfo, int bar, int offset,
unsigned short* data);

/*****************************************************************************
* CTIFPGARdDword
*
* This routine reads a DWORD from CTI FPGA baord's IO location.
* Arguments
*                    pBoardInfo: pointer to CtiUfpgaBoard
*                    bar: PCI BAR index (0 to 5)
*                    offset: offset to BAR memory
*                    data: pointer to data DWORD
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
```

```
DEFCPP int _stdcall CTIFPGARdDword(CtiUfpgaBoard *pBoardInfo, int bar, int offset,
unsigned int* data);

/*****************************************************************************
* CTIFPGARdCfgDword
*
* This routine reads a DWORD from CTI FPGA baord's PCI Configuration register location.
* Arguments
*                    pBoardInfo: pointer to CtiUfpgaBoard
*                    offset: offset to PCI Configuration space
*                    data: pointer to data DWORD
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
DEFCPP int _stdcall CTIFPGARdCfgDword(CtiUfpgaBoard *pBoardInfo, int offset, unsigned int
*data);

/*****************************************************************************
* CTIFPGARdMem
*
* This routine reads a block of data from CTI FPGA baord's IO location.
* Arguments
*                    pBoardInfo: pointer to CtiUfpgaBoard
*                    bar: PCI BAR index (0 to 5)
*                    offset: offset to BAR memory
*                    szbytes: number of bytes pointed to by 'buffer'
*                    buffer: pointer to data buffer
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
DEFCPP int _stdcall CTIFPGARdMem(CtiUfpgaBoard *pBoardInfo, int bar, int offset, int
szbytes, char *buffer);

/*****************************************************************************
* CTIFPGARdCfgMem
*
* This routine reads a byte from CTI FPGA baord's PCI Configuration register location.
* Arguments
*                    pBoardInfo: pointer to CtiUfpgaBoard
*                    offset: offset to PCI Configuration space
*                    szbytes: number of bytes pointed to by 'buffer'
*                    buffer: pointer to data buffer
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
DEFCPP int _stdcall CTIFPGARdCfgMem(CtiUfpgaBoard *pBoardInfo, int offset, int szbytes,
char *buffer);

/*****************************************************************************
* CTIFPGARdFifo
*
* This routine reads a block fifo from CTI FPGA baord's IO location.
* Arguments
*                    pBoardInfo: pointer to CtiUfpgaBoard
*                    bar: PCI BAR index (0 to 5)
```

```
*                   offset: offset to BAR FIFO location
*                   szbytes: number of bytes pointed to by 'buffer'
*                   buffer: pointer to data buffer
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
DEFCPP int _stdcall CTIFPGARdFifo(CtiUfpgaBoard *pBoardInfo, int bar, int offset, int
szbytes, char *buffer);

/*****************************************************************************
* CTIFPGAWrByte
*
* This routine writes a byte to CTI FPGA baord's IO location.
* Arguments
*                   pBoardInfo: pointer to CtiUfpgaBoard
*                   bar: PCI BAR index (0 to 5)
*                   offset: offset to BAR memory
*                   data: data to write
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/

DEFCPP int _stdcall CTIFPGAWrByte(CtiUfpgaBoard *pBoardInfo, int bar, int offset,
unsigned char data);

/*****************************************************************************
* CTIFPGAWrWord
*
* This routine writes a WORD to CTI FPGA baord's IO location.
* Arguments
*                   pBoardInfo: pointer to CtiUfpgaBoard
*                   bar: PCI BAR index (0 to 5)
*                   offset: offset to BAR memory
*                   data: data to write
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
DEFCPP int _stdcall CTIFPGAWrWord(CtiUfpgaBoard *pBoardInfo, int bar, int offset,
unsigned short data);

/*****************************************************************************
* CTIFPGAWrDword
*
* This routine writes a DWORD to CTI FPGA baord's IO location.
* Arguments
*                   pBoardInfo: pointer to CtiUfpgaBoard
*                   bar: PCI BAR index (0 to 5)
*                   offset: offset to BAR memory
*                   data: data to write
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
DEFCPP int _stdcall CTIFPGAWrDword(CtiUfpgaBoard *pBoardInfo, int bar, int offset,
unsigned int data);
```

```
/*****************************************************************************
* CTIFPGAWrCfgDword
*
* This routine writes a DWORD to CTI FPGA baord's PCI Configuration register location.
* Arguments
*                    pBoardInfo: pointer to CtiUfpgaBoard
*                    offset: offset to PCI Configuration space
*                    szbytes: number of bytes pointed to by 'data'
*                    data: data to write
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
DEFCPP int _stdcall CTIFPGAWrCfgDword(CtiUfpgaBoard *pBoardInfo, int offset, unsigned int
data);

/*****************************************************************************
* CTIFPGAWrMem
*
* This routine writes a block of data to CTI FPGA baord's IO location.
* Arguments
*                    pBoardInfo: pointer to CtiUfpgaBoard
*                    bar: PCI BAR index (0 to 5)
*                    offset: offset to BAR memory
*                    szbytes: number of bytes pointed to by 'buffer'
*                    buffer: pointer to data buffer to write
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
DEFCPP int _stdcall CTIFPGAWrMem(CtiUfpgaBoard *pBoardInfo, int bar, int offset, int
szbytes, char *buffer);

/*****************************************************************************
* CTIFPGAWrCfgMem
*
* This routine writes a block of data to CTI FPGA baord's PCI Configuration register
location.
* Arguments
*                    pBoardInfo: pointer to CtiUfpgaBoard
*                    offset: offset to PCI Configuration space
*                    szbytes: number of bytes pointed to by 'buffer'
*                    buffer: pointer to data buffer to write
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
DEFCPP int _stdcall CTIFPGAWrCfgMem(CtiUfpgaBoard *pBoardInfo, int offset, int szbytes,
char *buffer);

/*****************************************************************************
* CTIFPGAWrFifo
*
* This routine writes a block of data to CTI FPGA baord's FIFO location.
* Arguments
*                    pBoardInfo: pointer to CtiUfpgaBoard
*                    bar: PCI BAR index (0 to 5)
```

```
*                   offset: offset to BAR FIFO location
*                   szbytes: number of bytes pointed to by 'buffer'
*                   buffer: pointer to data buffer to write
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
DEFCPP int _stdcall CTIFPGAWrFifo(CtiUfpgaBoard *pBoardInfo, int bar, int offset, int
szbytes, char *buffer);

/****************************************************************************
* CTIWaitForIntr
*
* This routine triggers interrupt and wait for interrupt to happen.
* Arguments
*                   pBoardInfo: pointer to CtiUfpgaBoard
*                   bar: PCI BAR index (0 to 5)
*                   offset: offset to BAR FIFO location
*                   bitNumber: bit number to toggle
*                   polarity: 1 or 0 that disables interrupt
*                   timeout: timeout in MS
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/

/****************************************************************************
* CTIWaitForIntr
*
* This routine triggers interrupt and wait for interrupt to happen.
* Arguments
*                   pBoardInfo: pointer to CtiUfpgaBoard
*                   bar: PCI BAR index (0 to 5)
*                   offset: offset to BAR FIFO location
*                   bitNumber: bit number to toggle
*                   polarity: 1 or 0 that disables interrupt
*                   timeout: timeout in MS
*
* Returns:
*   If it succeeds then it returns CTI_STATUS_OK otherwise returns error.
*/
DEFCPP int _stdcall CTIWaitForIntr(CtiUfpgaBoard *pBoardInfo, int bar, int offset,
unsigned char bitNumber, BOOL polarity, unsigned int timeout);
#endif
```