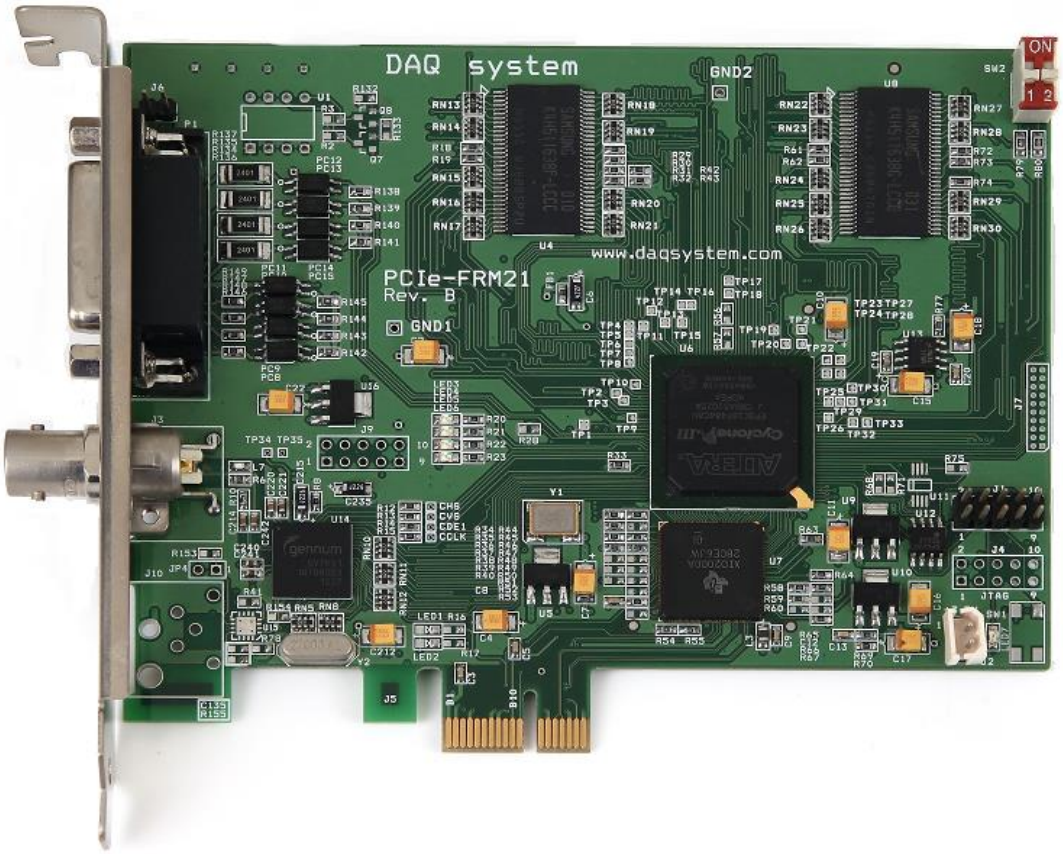
PCIe-FRM21

API Manual

Version 1.1



DAQ SYSTEM
Measurement & Automation

Contents

Board Level API Functions

LVDS(Camera Link) API Functions

UART API Functions

DIO(Digital Input Output) API Functions

Multi-Board LVDS(Camera Link) API Functions

Multi-Board UART API Functions

Multi-Board DIO(Digital Input Output)API Functions

Board Level API Functions

Overview

int **OpenDAQDevice (void)**
BOOL **ResetBoard (int nBoard)**
BOOL **CloseDAQDevice (void)**
int **GetBoardNum (void)**

OpenDAQDevice

It opens a device. You may call this function at the very first time you run the program and some suspicious operation.

int **OpenDAQDevice (void)**

Parameters: None .

Return Value:

If the function succeeds, it returns the number of boards which were detected.
If the function fails, the return value is -1, it means there is no device in the system.
(In case of multi-board, up to 4 is possible)

ResetBoard

It initializes a device at currently equipped system (PC).

BOOL **ResetBoard (int nBoard)**

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

It returns TRUE in case of the success of reset and initialization.
If you get FALSE you should not call any API functions with the board and call the **CloseDAQDevice()** instead.

CloseDAQDevice

The CloseDAQDevice function closes all opened devices (boards). If use of device is finished, it can certainly close a device for making it other programs so as usable.

BOOL CloseDAQDevice (void)

Parameters: None.

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

GetBoardNum

Returns currently detected board number in the system.

int GetBoardNum (void)

Parameters: None

Return Value:

The number of detected boards, The Board number is set by dip switch.

LVDS(Camera Link) API Functions

Overview

BOOL	**LVDS_Init (void)**
BOOL	**LVDS_Start (void)**
BOOL	**LVDS_Check (void)**
BOOL	**LVDS_GetFrame (DWORD* nCnt, unsigned char* buf)**
BOOL	**LVDS_Close (void)**
BOOL	**LVDS_SetModel (int Model)**
BOOL	**LVDS_GetModel (int *Model)**
BOOL	**LVDS_SetResolution (DWORD xRes, DWORD yRes)**
BOOL	**LVDS_GetResolution (DWORD *xRes, DWORD *yRes)**
BOOL	**LVDS_Stop (void)**
BOOL	**LVDS_SetDataMode (int nMode)**
BOOL	**LVDS_SetFilter (DWORD dwValue)**
BOOL	**LVDS_SetHsFilter (DWORD dwValue)**
BOOL	**LVDS_GetFilter (DWORD *dwValue)**
BOOL	**LVDS_SetHsCount (int nCount)**
BOOL	**LVDS_GetHsCount (int *nCount)**
BOOL	**LVDS_GetVersion (int *nVersion)**
BOOL	**LVDS_CameraMode (int nMode)**
BOOL	**LVDS_SetTriggerOutput (BOOL bBypass, int nCCNumber)**
BOOL	**LVDS_SetTriggerPulse (int nPulseCount, int nPulseWidth)**
BOOL	**SDI_SetMode (int nMode)**

LVDS_Init

This function Initializes resources used for the LVDS sub-system, for example interrupt and LVDS control register.

BOOL **LVDS_Init (void)**

Parameters: None.

Return Value:

If the function call fails, it returns “FALSE”.

If the function call succeeds, it returns “TRUE”.

LVDS_Start

This function starts receiving frame data. After calling this function, you can check whether the data is complete by calling the LVDS_GetFrame function.

BOOL LVDS_Start (void)

Parameters: None.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_Check

This function checks out the frame data acquisition has been completed.

BOOL LVDS_Check (void)

Parameters: none.

Return Value:

If the frame data has been completed, It returns "True".

If he frame data has not been completed. It returns "FALSE"

LVDS_GetFrame

This function checks whether the frame data is complete, and if it is, retrieves the frame data. At this time, the size of the buffer to receive data must be informed.

BOOL LVDS_GetFrame (DWORD* nCnt, unsigned char* buf)

Parameters:

*nCnt : It is the address which contains the number of data to be received in byte size. Specifies the size buffer when the function is called, and read the values of the variables after a call to find out how many actually read. The data size is in bytes.

*buf : Frame buffer pointer.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, check the values of the size that you want to read nCnt.

(Note) If the frame data is not completed, FALSE is returned immediately and the return occurs with the nCnt value set to 0.

LVDS_Close

This function releases all resource were used for LVDS function. The application program calls this function when the program ends.

BOOL LVDS_Close (void)

Parameters: None.

Return Value :

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_SetModel

This function sets up the Camera model for LVDS interface.

BOOL LVDS_SetModel (int Model)

Parameters:

Model : Model number '0' selects 2048 x 1560 resolution Camera, otherwise selects 3160 x 2560 resolution Camera.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_GetModel

This function gets a setup value of the Camera model for LVDS interface.

BOOL LVDS_GetModel (int *Model)

Parameters:

*Model: Address value for camera model number

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_SetResolution

This function sets camera resolution for the specific camera Model.

BOOL LVDS_SetResolution (DWORD xRes, DWORD yRes)

Parameters:

xRes : Value of the horizontal Camera resolution

yRes : Value of the vertical Camera resolution

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_GetResolution

This function gets currently configured camera's frame resolution

BOOL LVDS_GetResolution (DWORD *xRes, DWORD *yRes)

Parameters:

*xRes : Address pointer to receive horizontal Camera resolution

*yRes : Address pointer to receive vertical Camera resolution

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_Stop

This function stops the frame data capture.

BOOL LVDS_Stop (void)

Parameters: None.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_SetDataMode

This function sets image pixel data mode.

BOOL LVDS_SetDataMode (int nMode)

Parameters:

nMode : If it is "2", it is 24bit Mode, and if it is "Others", it is 16bit Mode.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_SetFilter

This function can put a strain on the system when an abnormal Hsync comes in.

In this case, a filter is set in hardware to filter out abnormal signals.

BOOL LVDS_SetFilter (DWORD dwValue)

Parameters:

dwValue : The valid range of values is from 0 to 65535, the unit is about 15nSEC. The default value is 160. If you set up a filter, you should keep in mind a Front porch and Back porch.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_SetHsFilter

This function does not occur in a normal case, but if a cable is connected, or if Hsync is not generated normally from a damaged camera and it enters quickly, it may give a strain to the system. In this case, you can filter out abnormal signals by setting a hardware filter in Hsync.

BOOL LVDS_SetHsFilter (DWORD dwValue)

Parameters:

dwValue : The valid range of values is from 0 to 65535, the unit is about 15nSEC. The default value is 160. If you set up a filter, you should keep in mind a Front porch and Back porch.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_GetFilter

This function gets the Hsync filter current value.

BOOL LVDS_GetFilter (DWORD *dwValue)

Parameters:

*dwValue: Valuable address of set filter values.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_SetHsCount

This function sets the value of a Horizontal sync count for image checking.

BOOL LVDS_SetHsCount (int nCount)

Parameters:

nCount: The minimum value of the frame's Horizontal sync.
The frame is useless smaller than this value.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_GetHsCount

This function gets the value of a Horizontal sync count.

BOOL LVDS_GetHsCount (int *nCount)

Parameters:

*nCount: Valuable of Horizontal count.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_GetVersion

This function gets FPGA version.

BOOL LVDS_GetVersion (int *nVersion)

Parameters:

*nVersion : The pointer of the FPGA version. Some API are only supported by boards which have the FPGA version number 2 or more.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_CameraMode

This function sets a Camera Operation Mode.

BOOL LVDS_CameraMode (int nMode)

Parameters:

nMode : "0" : Area Mode (default),
"1" : Line Mode (Free Run)
"2" : Line Mode (Trigger from external port)
"3" : Line Mode. (Trigger with internal (33MHz) clock)

Return Value :

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_SetTriggerOutput

This function sets a configure value of the corresponding bits(CC1 ~ CC4).

BOOL LVDS_SetTriggerOutput (BOOL bBypass, int nCCNumber)

Parameters:

bBypass : "True" if bypassing the assigned cc used for pulse setting of external trigger (DIO Board), otherwise "False"

nCCNumber : "0" : CC0, "1" : CC1, "2" : CC2, "3" : CC3, "Others" : null

Return Value :

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_SetTriggerPulse

This function configures a delay and width of Trigger.

BOOL LVDS_SetTriggerOutput (int nPulseCount, int nPulseWidth)

Parameters:

nPulseCount : How many pulses are used to trigger out (default : 2200)

nPulseWidth : How many pulses are used for the trigger width (default : 4)

Return Value :

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

SDI_SetMode

This function selects the frame (image) data mode.

BOOL SDI_SetMode (int nMode)

Parameters:

nMode : "0" : Progressive Mode, "1" : Interlace Mode.

Return Value :

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

UART API Functions

Overview

BOOL **UART_Init (void)**
BOOL **UART_GetData (DWORD* nCnt, unsigned char* buf)**
BOOL **UART_SendData (DWORD* nCnt, unsigned char* buf)**
BOOL **UART_Close (void)**
BOOL **UART_SetBaud (DWORD nBaud)**
BOOL **UART_BufferFlush (void)**

UART_Init

This function initializes resources used for the UART sub-system, for example interrupt and UART control register.

BOOL **UART_Init (void)**

Parameters: None.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

UART_GetData

This function receives characters through the differential UART.

BOOL **UART_GetData (DWORD* nCnt, unsigned char* buf)**

Parameters:

*nCnt : The address which contains the number of characters to be received.
The maximum number of characters to be received is limited to 4Kbyte(4096).

*buf : The buffer address.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

UART_SendData

This function sends characters through the differential UART.

BOOL UART_SendData (DWORD* nCnt, unsigned char* buf)

Parameters:

*nCnt : The address which contains the number of characters to be sent.
The maximum number of characters to be sent is limited to 4K byte(4096).

*buf : The buffer address.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

UART_Close

This function releases all resource were used for UART function.

BOOL UART_Close (void)

Parameters: None.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

UART_SetBaud

This function sets the UART Baud.

BOOL UART_SetBaud (DWORD nBaud)

Parameters:

nBaud : 0 : 9600, 1 : 19200, 2 : 38400, 3 :57600, 4 : 115200

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

UART_BufferFlush

This function clears the UART Rx buffer.

BOOL **UART_BufferFlush (void)**

Parameters: None.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

DIO(Digital Input Output) API Functions

Overview

BOOL **DIO_Read (void)**

BOOL **DIO_Write (DWORD val)**

DIO_Read

This function reads the Digital Input state.

BOOL **DIO_Read (void)**

Parameters: None.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

DIO_Write

This function sets up the Digital Output pin state.

BOOL **DIO_Write (DWORD val)**

Parameters:

val : The value to be written to the port.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

Multi Board support APIs

In case of single board API, only one board is used in the installed system. However, in a system with two or more boards installed (up to 4 supported), multiple APIs must be used. Multi board API is only available for FPGA version #2 or higher boards.

Multi Board LVDS(Camera Link) APIs

Overview

BOOL	**LVDS_Init_Mul (int nBoard)**
BOOL	**LVDS_Start_Mul (int nBoard)**
BOOL	**LVDS_GetFrame_Mul (int nBoard, DWORD* nCnt, unsigned char* buf)**
BOOL	**LVDS_Close_Mul (int nBoard)**
BOOL	**LVDS_SetModel_Mul (int nBoard, int Model)**
BOOL	**LVDS_GetModel_Mul (int nBoard, int *Model)**
BOOL	**LVDS_SetResolution_Mul (int nBoard, DWORD xRes, DWORD yRes)**
BOOL	**LVDS_GetResolution_Mul (int nBoard, DWORD *xRes, DWORD *yRes)**
BOOL	**LVDS_Stop_Mul (int nBoard)**
BOOL	**LVDS_SetDataMode_Mul (int nBoard, int nMode)**
BOOL	**LVDS_SetFilter_Mul (int nBoard, DWORD dwValue)**
BOOL	**LVDS_SetHsFilter_Mul (int nBoard, DWORD dwValue)**
BOOL	**LVDS_GetFilter_Mul (int nBoard, DWORD *dwValue)**
BOOL	**LVDS_SetHsCount_Mul (int nBoard, int nCount)**
BOOL	**LVDS_GetHsCount_Mul (int nBoard, int *nCount)**
BOOL	**LVDS_GetVersion_Mul (int nBoard, int *nVersion)**
BOOL	**LVDS_CameraMode_Mul (int nBoard, int nMode)**
BOOL	**LVDS_SetTriggerOutput_Mul (int nBoard, BOOL bBypass, int nCCNumber)**
BOOL	**LVDS_SetTriggerPulse_Mul (int nBoard, int nPulseCount, int nPulseWidth)**
BOOL	**SDI_SetMode_Mul (int nBoard, int nMode)**

LVDS_Init_Mul

This function initializes resources used for the LVDS sub-system, for example interrupt and LVDS control register.

BOOL LVDS_Init_Mul (int nBoard)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function call fails, it returns "FALSE".
If the function call succeeds, it returns "TRUE".

LVDS_Start_Mul

This function starts receiving frame data. After calling this function, you can check whether the data is complete by calling the LVDS_GetFrame function.

BOOL LVDS_Start_Mul (int nBoard)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function call fails, it returns "FALSE".
If the function call succeeds, it returns "TRUE".

LVDS_Check_Mul

This function checks out the frame data acquisition has been completed.

BOOL LVDS_Check_Mul (int nBoard)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the frame data has been completed, It returns "True".
If he frame data has not been completed. It returns "FALSE"

LVDS_GetFrame_Mul

This function checks whether the frame data is complete, and if it is, retrieves the frame data. At this time, the size of the buffer to receive data must be informed.

BOOL LVDS_GetFrame_Mul (int nBoard, DWORD* nCnt, unsigned char* buf)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

*nCnt : It is the address which contains the number of data to be received in byte size. Specifies the size buffer when the function is called, and read the values of the variables after a call to find out how many actually read. The data size is in bytes.

*buf : The buffer address.

Return Value:

If the function call fails, it returns "FALSE".
If the function call succeeds, it returns "TRUE".

LVDS_Close_Mul

This function returns all the resources used by the LVDS function. The application program calls this function when the program ends.

BOOL LVDS_Close (int nBoard)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value :

If the function call fails, it returns "FALSE".
If the function call succeeds, it returns "TRUE".

LVDS_SetModel_Mul

This function sets up the Camera model for LVDS interface.

BOOL LVDS_SetModel_Mul (int nBoard, int Model)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Model : Model number '0' selects 2048 x 1560 resolution Camera,
otherwise selects 3160 x 2560 resolution Camera.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_GetModel_Mul

This function gets a setup value of the Camera model for LVDS interface.

BOOL LVDS_GetModel_Mul (int nBoard, int *Model)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

*Model: Address value for camera model number

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_SetResolution_Mul

This function sets camera resolution for the specific camera Model.

BOOL LVDS_SetResolution_Mul (int nBoard, DWORD xRes, DWORD yRes)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

xRes : Value of the horizontal Camera resolution

yRes : Value of the vertical Camera resolution

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_GetResolution_Mul

This function gets currently configured camera's frame resolution

BOOL LVDS_GetResolution_Mul (int nBoard, DWORD *xRes, DWORD *yRes)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

*xRes : Address pointer to receive horizontal Camera resolution

*yRes : Address pointer to receive vertical Camera resolution

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_Stop_Mul

This function stops the frame data capture.

BOOL LVDS_Stop_Mul (int nBoard)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function call fails, it returns "FALSE".
If the function call succeeds, it returns "TRUE".

LVDS_SetDataMode_Mul

This function sets image pixel data mode.

BOOL LVDS_SetDataMode_Mul (int nBoard, int nMode)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.
nMode : If the value is 2, the pixel data be expressed by 24bits, others be 16bits.

Return Value:

If the function call fails, it returns "FALSE".
If the function call succeeds, it returns "TRUE".

LVDS_SetFilter_Mul

This function can put a strain on the system when an abnormal Hsync comes in. In this case, a filter is set in hardware to filter out abnormal signals.

BOOL LVDS_SetFilter_Mul (int nBoard, DWORD dwValue)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

dwValue : The valid range of values is from 0 to 65535, the unit is about 15nSEC. The default value is 160. If you set up a filter, you should keep in mind a Front porch and Back porch.

Return Value:

If the function call fails, it returns "FALSE".
If the function call succeeds, it returns "TRUE".

LVDS_SetHsFilter_Mul

This function does not occur in a normal case, but if a cable is connected, or if Hsync is not generated normally from a damaged camera and it enters quickly, it may give a strain to the system. In this case, you can filter out abnormal signals by setting a hardware filter in Hsync.

BOOL LVDS_SetHsFilter_Mul (int nBoard, DWORD dwValue)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

dwValue : The valid range of values is from 0 to 65535, the unit is about 15nSEC. The default value is 160. If you set up a filter, you should keep in mind a Front porch and Back porch.

Return Value:

If the function call fails, it returns "FALSE".
If the function call succeeds, it returns "TRUE".

LVDS_GetFilter_Mul

This function gets the Vsync filter current value.

BOOL LVDS_GetFilter_Mul (int nBoard, DWORD *dwValue)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

*dwValue: Valuable address of set filter values.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_SetHsCount_Mul

This function sets the value of a Horizontal sync count for image checking.

BOOL LVDS_SetHsCount_Mul (int nBoard, int nCount)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

nCount: The minimum value of the frame's Horizontal sync.
The frame is useless smaller than this value.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

LVDS_GetHsCount_Mul

This function gets the value of a Horizontal sync count.

BOOL LVDS_GetHsCount_Mul (int nBoard, int *nCount)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

*nCount: Valuable address of Horizontal count.

Return Value:

If the function call fails, it returns "FALSE".
If the function call succeeds, it returns "TRUE".

LVDS_GetVersion_Mul

This function gets a FPGA version.

BOOL LVDS_GetVersion_Mul (int nBoard, int *nVersion)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

*nVersion : The pointer of the FPGA version.
Some API are only supported by boards which have the FPGA version number 2 or more.

Return Value:

If the function call fails, it returns "FALSE".
If the function call succeeds, it returns "TRUE".

LVDS_CameraMode_Mul

This function sets a Camera Operation Mode.

BOOL LVDS_CameraMode_Mul (int nBoard, int nMode)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

nMode : "0" : Area Mode (default),
"1" : Line Mode (Free Run)
"2" : Line Mode (Trigger from external port)
"3" : Line Mode. (Trigger with internal (33MHz) clock)

Return Value :

If the function call fails, it returns "FALSE".
If the function call succeeds, it returns "TRUE".

LVDS_SetTriggerOutput_Mul

This function sets a configure value of the corresponding bits(CC1 ~ CC4).

BOOL LVDS_SetTriggerOutput_Mul (int nBoard, BOOL bBypass, int nCCNumber)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

bBypass: "True" if bypassing the assigned cc used for pulse setting of external trigger (DIO Board), otherwise "False"

nCCNumber : "0" : CC0, "1" : CC1, "2" : CC2, "3" : CC3, "Others" : null

Return Value :

If the function call fails, it returns "FALSE".
If the function call succeeds, it returns "TRUE".

LVDS_SetTriggerPulse_Mul

This function configures a delay and width of Trigger.

BOOL LVDS_SetTriggerOutput_Mul (int nBoard, int nPulseCount, int nPulseWidth)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

nPulseCount : How many pulses are used to trigger out (default : 2200)

nPulseWidth : How many pulses are used for the trigger width (default : 4)

Return Value :

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

SDI_SetMode_Mul

This function selects the frame (image) data mode.

BOOL SDI_SetMode_Mul (int nBoard, int nMode)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

nMode : "0" : Progressive Mode,
"1" : Interlace Mode.

Return Value :

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

Multi-Board UART API Functions

Overview

BOOL UART_Init_Mul (int nBoard)
BOOL UART_GetData_Mul (int nBoard, DWORD* nCnt, unsigned char* buf)
BOOL UART_SendData_Mul (int nBoard, DWORD* nCnt, unsigned char* buf)
BOOL UART_Close_Mul (int nBoard)
BOOL UART_SetBaud_Mul (int nBoard, DWORD nBaud)
BOOL UART_BufferFlush_Mul (int nBoard)

UART_Init_Mul

This function initialize resources used for the UART sub-system, for example interrupt and UART control register.

BOOL UART_Init_Mul (int nBoard)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function call fails, it returns "FALSE".
If the function call succeeds, it returns "TRUE".

UART_GetData_Mul

This function receives characters through the differential UART.

BOOL UART_GetData_Mul (int nBoard, DWORD* nCnt, unsigned char* buf)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.
*nCnt : The address which contains the number of characters to be received.
The maximum number of characters to be received is limited to 4Kbyte(4096).
*buf : The buffer address.

Return Value:

If the function call fails, it returns "FALSE".
If the function call succeeds, it returns "TRUE".

UART_SendData_Mul

This function sends characters through the differential UART.

BOOL UART_SendData_Mul (int nBoard, DWORD* nCnt, unsigned char* buf)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

*nCnt : The address which contains the number of characters to be sent.
The maximum number of characters to be sent is limited to 4K byte(4096).

*buf : The buffer address.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

UART_Close_Mul

This function releases all resource were used for UART function.

BOOL UART_Close_Mul (int nBoard)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

UART_SetBaud_Mul

This function sets UART Baud rates.

BOOL UART_SetBaud_Mul (int nBoard, DWORD nBaud)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

nBaud : 0 : 9600, 1 : 19200, 2 : 38400, 3 :57600, 4 : 115200

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

UART_BufferFlush_Mul

This function flushes UART RX Buffer

BOOL UART_BufferFlush_Mul (int nBoard)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

Multi-board DIO(Digital Input Output) API Functions

Overview

BOOL **DIO_Read_Mul (int nBoard)**

BOOL **DIO_Write_Mul (int nBoard, DWORD val)**

DIO_Read_Mul

This function reads from input port.

BOOL **DIO_Read_Mul (int nBoard)**

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

DIO_Write_Mul

This function writes to output port.

BOOL **DIO_Write_Mul (int nBoard, DWORD val)**

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

val : The value to be written to the port.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

Memo

Contact Point

Web sit : https://www.daqsystem.com

Email : postmaster@daqsystem.com