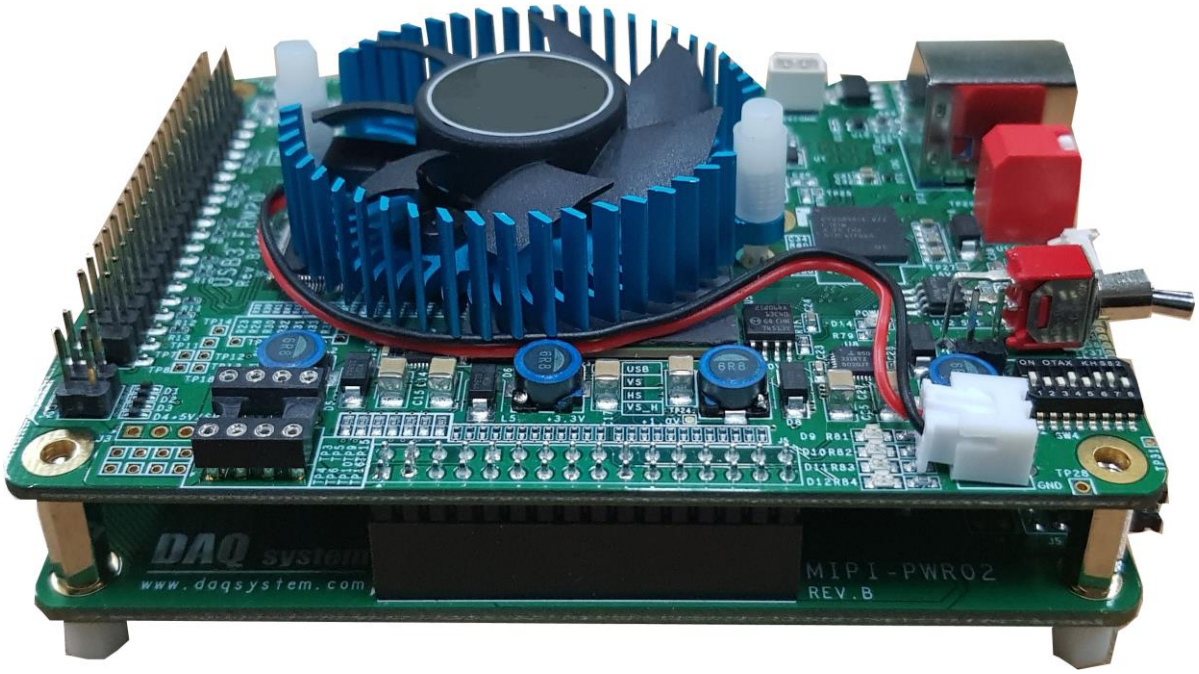
USB3-FRM20

API Manual

Version 1.1



DAQ SYSTEM
Measurement & Automation

Contents

Board Level API Functions

LVDS API Functions

Clcock API Functions

Parallel GPIO Input/Output API Functions

Sensor GPIO Input/Output API Functions

User GPIO(3.3V Fix) Input/Output API Functions

Power Board SP(Sensor Power) API Functions

I2C API Functions

Power Control API Functions

SPI(Serial Peripheral Interface) API Functions

Board Level APIs

Overview

Int **OpenDAQDevice (void)**
BOOL **ResetBoard (int nBoard)**
BOOL **CloseDAQDevice (void)**
Int **GetBoardNum (void)**
char* **GetDllVersion (void)**
BOOL **SendUserMsg (int nBoard, unsigned char* chBuf, BOOL bReply)**
BOOL **IsConnected (int nBoard)**

OpenDAQDevice

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

BOOL **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.

ResetBoard

This function 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 using 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

This function returns currently detected board number in the system.

int **GetBoardNum (void)**

Parameters:None

Return Value:

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

GetDllVersion

This function tells the DLL version.

Char* **GetDllVersion (void)**

Parameters: None.

Return Value:

Gives the date of the installed DLL.

SendUserMsg

This function starts the sending the message.

BOOL SendUserMsg(int nBoard, unsigned char *chBuf, BOOL bReply)

Parameters:

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

chBuf : Address where the message to be sent is located

bReply : "1" if message transmission is successful
"0" if message transmission fails

Return Value:

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

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

IsConnected

This function confirms that the board is connected to USB.

BOOL IsConnected (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 API Functions

Overview

BOOL LVDS_Init (int nBoard)
BOOL LVDS_Start (int nBoard)
BOOL LVDS_GetFrame (int nBoard, DWORD* nCnt, unsigned char* buf)
BOOL LVDS_GetError ((int nBoard, DWORD *dwStatus)
BOOL LVDS_Close (int nBoard)
BOOL LVDS_GetResolutuion (int nBoard, DWORD *xRes, DWORD *yRes)
BOOL LVDS_SetResolutuionDelay (int nBoard, DWORD dwDelay)
BOOL LVDS_Stop (int nBoard)
BOOL LVDS_SetDataMode (int nBoard, int nMode)
BOOL LVDS_VirtualMode (int nBoard, BOOL bSet)
BOOL LVDS_GetVersion (int nBoard, int *nFpgaVer, int nFirmVer)
BOOL LVDS_SelectInput (int nBoard, Int nInput)
BOOL LVDS_GetMipiID (int nBoard, DWORD *dwID)
BOOL LVDS_BufferFlush (int nBoard)
BOOL LVDS_CheckSum (int nBoard, BOOL bOn)
BOOL LVDS_SetRolling (int nBoard, BOOL bRolling)

LVDS_Init

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

BOOL LVDS_Init (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

When LVDS_Start function is called, it starts to transmit data from Board to DLL.
(In the case of USB Board, the data is accumulated in the buffer of the DLL through the thread and copied to the DLL-> APP using the LVDS_GetFrame function.)

BOOL LVDS_Start (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_GetFrame

It checks whether the frame data is complete, and if it is, the frame data is retrieved.
At this time, you must tell the size of the buffer to receive the data.

BOOL LVDS_GetFrame (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 : 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.
(Remark) If it is not opened or the DLL thread is not running, if the nCnt value entered by the USER is larger than the buffer size, nCnt is returned to "0" and even if the frame data is less accumulated, it returns only "FALSE".

LVDS_GetError

This function gets a Frame (Image) Error.

DWORD LVDS_GetError (DWORD *dwStatus)

Parameters:

*dwStatus : "1" : Overflow error, "2" : Read error

"4" : Size error

Return Value :

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

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

LVDS_Close

This function returns all resources used by the LVDS function. The application program calls this function at the end of the program.

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_GetResolution

This function gets currently configured camera's frame resolution.

BOOL LVDS_GetResolutuion (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_SetResolutionDelay

This function sets the resolution delay time of the video input.

BOOL LVDS_SetResolutuionDelay (int nBoard, DWORD dwDelay)

Parameters:

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

dwDelay : milisecond unit (Max. 10sec)

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 (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

This function sets the input frame (image) data mode.

BOOL LVDS_SetDataMode (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" : 8bit Mode, "1" : 16bit Mode,
"2" : 24bit Mode. "3" : 32bit Mode, "4" : YUV Mode.

Return Value:

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

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

LVDS_VirtualMode

This function sets the virtual data mode.

BOOL LVDS_VirtualMode (int nBoard, BOOL bSet)

Parameters:

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

bSet : "1" : Virtual mode.

Return Value:

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

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

LVDS_GetVersion

Get the current FPGA and Firmware version.

BOOL LVDS_GetVersion (int nBoard, int *nFpgaVer, int nFirmVer)

Parameters:

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

*nFpgaVer: The value of current FPGA version.

*nFirmVer: The value of current Firmware version.

Return Value:

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

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

LVDS_SelectInput

This function sets the LVDS Input mode.

BOOL LVDS_SelectInput (int nBoard, int nInput)

Parameters:

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

nInput : "0" : C-PHY Mode, "1" : D_PHY Mode,

Return Value:

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

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

LVDS_GetMipiID

This function reads the MIPI image ID.

BOOL LVDS_GetMipiID (int nBoard, DWORD *dwID)

Parameters:

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

*dwID: MIPI Image ID Address.

Return Value:

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

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

LVDS_BufferFlush

This function initializes the buffer.

BOOL LVDS_BufferFlush (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_CheckSum

This function determines whether CRC Check Sum is executed.

BOOL LVDS_SelectInput (int nBoard, BOOL bOn)

Parameters:

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

bOn : "0" : Do not add CheckSum Data
"1" : Add 2 bytes of CheckSum Data to each frame line.

Return Value :

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

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

LVDS_SetRolling

This function updates the DDR memory without using the LVDS frame data and the GetFrame function.

BOOL LVDS_SetRolling (int nBoard, BOOL bRolling)

Parameters:

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

bRolling : If "1", DDR memory is updated regardless of Getframe function.

Return Value :

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

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

Clock API Functions

Overview

BOOL CLK_Select (int nBoard, int nSelect)

BOOL CLK_Set (int nBoard, DWORD val)

BOOL CLK_Off (int nBoard, BOOL bOff)

CLK_Select

Select the fixed clock or programmed clock.

BOOL CLK_Select (int nBoard, int nSelect)

Parameters:

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

nSelect : If the value is "0", it is a Fixed Clock.
If the value is "Others", it is a Program Clock.

Return Value :

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

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

CLK_Set

It sets the MCLK(Master Clock) frequency. (1039Hz ~ 68Mhz)

BOOL CLK_Set (int nBoard, DWORD val)

Parameters:

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

val : 1039Hz ~ 68Mhz.

Return Value :

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

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

CLK_Off

This function turns on/off MCLK (Master Clock).

BOOL CLK_Off (int nBoard, BOOL bOff)

Parameters:

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

bOff : "0" : Clock On , '1" : Clock Off

Return Value :

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

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

Parallel GPIO Input/Output API Functions

Overview

BOOL DIO_SetDirection (int nBoard, DWORD dwVal)
BOOL DIO_GetDirection (int nBoard, DWORD *dwVal)
DWORD DIO_Read (int nBoard)
BOOL DIO_Write (int nBoard, DWORD dwVal)
BOOL DIO_GetWrite (int nBoard, DWORD *dwVal)

DIO_SetDirection

This function sets whether to use each port of the General Purpose I/O J5 connector (25..32) as input or output.

BOOL DIO_SetDirection (int nBoard, DWORD dwVal)

Parameters:

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

dwVal : Input/Output direction setting value. Each port is '1' : Output / '0' : Input

Return Value:

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

DIO_GetDirection

This function reads the currently set J5 connector (25..32) direction value.

BOOL DIO_GetDirection (int nBoard, DWORD *dwVal)

Parameters:

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

*dwVal : Input/Output direction to read in variables.

Return Value:

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

DIO_Read

This function reads the input value of the General Purpose I/O J5 connector (25..32).

DWORD DIO_Read (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

This function outputs the desired J5 connector (25..32) value to the output port.

BOOL DIO_Write (int nBoard, DWORD dwVal)

Parameters:

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

DIO_GetWrite

This function writes the output current J5 connector (25..32) value.

BOOL DIO_GetWrite (int nBoard, DWORD *dwVal)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.
*dwVal : Variable of the output port's current value.

Return Value:

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

Sensor GPIO Input/Output API Functions

Overview

BOOL SDIO_SetDirection (int nBoard, DWORD dwVal)
BOOL SDIO_GetDirection (int nBoard, DWORD *dwVal)
DWORD SDIO_Read (int nBoard)
BOOL SDIO_Write (int nBoard, DWORD dwVal)
BOOL SDIO_GetWrite (int nBoard, DWORD *dwVal)

SDIO_SetDirection

This function sets whether to use each port of the sensor board's GPIO(J1 CNT0..3) as an input or output.

BOOL SDIO_SetDirection (int nBoard, DWORD dwVal)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.
dwVal : Input/Output direction value
"1" : Output, "0" : Input for each port

Return Value:

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

SDIO_GetDirection

This function reads the current direction value of GPIO of sensor board.

BOOL SDIO_GetDirection (int nBoard, DWORD *dwVal)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.
*dwVal : Variable to read the input / output direction.
"1" : Output, "0" : Input for each port

Return Value:

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

SDIO_Read

This function reads the GPIO input value of the sensor board.

DWORD SDIO_Read (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".

SDIO_Write

This function outputs the desired value to the sensor board's GPIO to the output port.

BOOL SDIO_Write (int nBoard, DWORD dwVal)

Parameters:

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.
dwVal : Value to write to the output port.

Return Value:

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

SDIO_GetWrite

This function writes the current value output to the GPIO of the sensor board.

BOOL SDIO_GetWrite (int nBoard, DWORD *dwVal)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.
*dwVal : Variable to write current value of output port.

Return Value:

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

User GPIO(3.3V Fix) Input/Output API Functions

Overview

BOOL DIO33_SetDirection (int nBoard, DWORD dwVal)
BOOL DIO33_GetDirection (int nBoard, DWORD *dwVal)
DWORD DIO33_Read (int nBoard)
BOOL DIO33_Write (int nBoard, DWORD dwVal)
BOOL DIO33_GetWrite (int nBoard, DWORD *dwVal)

DIO33_SetDirection

This function sets whether to use each port of 3.3V GPIO (J5 connector (21..24)) as input or output.

BOOL DIO33_SetDirection (int nBoard, DWORD dwVal)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.
dwVal : Input/Output direction value
"1" : Output, "0" : Input for each port

Return Value:

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

DIO33_GetDirection

This function reads the currently set user 3.3V GPIO (J5 connector(21..24)) direction value.

BOOL DIO33_GetDirection (int nBoard, DWORD *dwVal)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.
*dwVal : Variable to read the input / output direction.
"1" : Output, "0" : Input for each port

Return Value:

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

DIO33_Read

This function reads user 3.3V GPIO (J5 connector (21..24)) input value.

DWORD DIO33_Read (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".

DIO33_Write

This function outputs the desired value of the user 3.3V GPIO to the output port.

BOOL DIO33_Write (int nBoard, DWORD dwVal)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.
dwVal : Value to write to the output port.

Return Value:

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

DIO33_GetWrite

This function writes the current value output to the user 3.3V GPIO(J5 connector (21..24)).

BOOL DIO33_GetWrite (int nBoard, DWORD *dwVal)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.
*dwVal : Variable to write current value of output port.

Return Value:

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

Power Board SP(Sensor Power) API Functions

Overview

BOOL SP_SetDirection (int nBoard, DWORD dwVal)

BOOL SP_GetDirection (int nBoard, DWORD *dwVal)

BOOL SP_Write (int nBoard, int nCh, float fVolt)

SP_SetDirection

This function sets which channel to use among Sensor Power. This function is only used on boards using PowerBoard.

BOOL SP_SetDirection (int nBoard, DWORD dwVal)

Parameters:

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

dwVal : Channel Values.
0x00 => ALL Off,
0x01 => Ch1(SP0) Off, 0x02 => Ch2(SP1) Off
0x04 => Ch3(SP2) Off, 0x08 => Ch4(SP3) Off
0x10 => Ch5(SP4) Off, 0x20 => Ch6(SP5) Off
0x40 => Reserve, 0x80 => Reserve

Return Value:

If the function call fails, it returns "FALSE".
If the function call succeeds, it returns "TRUE".
(In logic that does not use PowerBoard, -1 is returned.)

SP_GetDirection

This function sets which channel to bring among Sensor Power. This function is only used on boards using PowerBoard.

BOOL SP_GetDirection (int nBoard, DWORD *dwVal)

Parameters:

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

*dwVal : Channel Values.

0x00 => ALL Off,

0x01 => Ch1(SP0) On, 0x02 => Ch2(SP1) On

0x04 => Ch3(SP2) On, 0x08 => Ch4(SP3) On

0x10 => Ch5(SP4) On, 0x20 => Ch6(SP5) On

0x40 => Reserve, 0x80 => Reserve

Return Value:

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

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

(In logic that does not use PowerBoard, -1 is returned.)

SP_Write

This function sets the voltage value for each channel of Sensor Power. This function is only used on boards using PowerBoard.

BOOL SP_Write (int nBoard, int nCh, float fVolt)

Parameters:

nBoard : It informs a board number at currently equipped system.

The board number set up by DIP switch.

nCh : Channel Values.

0x01 : SP0

0x02 : SP1

0x04 : SP2

0x08 : SP3

0x10 : SP4

0x20 : SP5

fVolt : Voltage Value (1.0 ~ 4.0V)

Return Value:

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

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

If the fVolt value is less than 0.89 or greater than 4.1, -2 is returned.

(In logic that does not use PowerBoard, -1 is returned.)

I2C API Functions

Overview

BOOL I2C_PWR_Init (int nBoard, DWORD nKHz)

BOOL I2C_PWR_Read (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)

BOOL I2C_PWR_Write (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)

BOOL I2C_PWR_Close (int nBoard)

BOOL I2C_ADC_Init (int nBoard, DWORD nKHz)

BOOL I2C_ADC_Read (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)

BOOL I2C_ADC_Write (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)

BOOL I2C_ADC_Close (int nBoard)

BOOL I2C_OS_Init (int nBoard, DWORD nKHz)

BOOL I2C_OS_Read (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)

BOOL I2C_OS_Write (int nBoard, BYTE slAddr, DWORD nCnt, unsigned char* buf)

BOOL I2C_OS_Closel (int nBoard)

BOOL I2C_SYS_Reset (int nBoard)

BOOL I2C_SYS_Set_Clock (int nBoard, int nClock)

BOOL I2C_SYS_Read (int nBoard, BYTE slAddr, DWORD nAddrLen, DWORD nAddr, DWORD nCnt, unsigned char* buf)

BOOL I2C_SYS_Write (int nBoard, BYTE slAddr, DWORD nAddrLen, DWORD nAddr, DWORD nCnt, unsigned char* buf)

BOOL I2C_SYS_Read2 (int nBoard, Byte slAddr, DWORD nAddrLen, DWORD nAddr, DWORD nCnt, unsigned char* buf)

BOOL I2C_SYS_Write2 (int nBoard, Byte slAddr, DWORD nAddrLen, DWORD nAddr, DWORD nCnt, unsigned char* buf)

BOOL I2C_SYS_Read_Ex (int nBoard, BYTE slAddr, DWORD nAddrLen, unsigned char* Addrbuf, DWORD nCnt, unsigned char* buf)

BOOL I2C_SYS_Read2_Ex (int nBoard, BYTE slAddr, DWORD nAddrLen, unsigned char* Addrbuf, DWORD nCnt, unsigned char* buf)

BOOL I2C_SYS_Write_Ex (int nBoard, BYTE slAddr, DWORD nAddrLen, unsigned char* Addrbuf, DWORD nCnt, unsigned char* buf)

BOOL I2C_SYS_Write2_Ex (int nBoard, BYTE slAddr, DWORD nAddrLen, unsigned char* Addrbuf, DWORD nCnt, unsigned char* buf)

BOOL SEN_Reset (int nBoard, BOOL bReset)

BOOL SEN_Enable (int nBoard, BOOL bEnable)

BOOL SEN_Power (int nBoard, BOOL bOn)

I2C_PWR_Init

This functions initializes the I2C function of Power module depending on clock speed. When the program run, call once. Others I2C PWR functions use after calling this function.

BOOL I2C_PWR_Init (int nBoard, DWORD nKHz)

Parameters:

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

nKHz : I2C communication speed (set the Clock speed.)
1: 100KHz, 2: 200KHz, 3: 300KHz, 4: 400KHz, Others: 100KHz

Return Value:

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

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

I2C_PWR_Read

This functions read a fixed value of specific address through I2C.

BOOL I2C_PWR_Read (int nBoard, Byte slAddr, DWORD nCnt, unsigned char* buf)

Parameters:

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

slAddr : I2C address

nCnt : The number of bytes of data read.

buf : Buffer pointer to read data.

Return Value:

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

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

I2C_PWR_Write

This function writes a fixed value of specific address through I2C.

BOOL I2C_PWR_Write (Byte slAddr, DWORD nCnt, unsigned char* buf)

Parameters:

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.
slAddr : I2C address
nCnt : The number of bytes of data read.
buf : Buffer pointer to read data.

Return Value:

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

I2C_PWR_Close

The I2C function completely is finished. When the program finishes, call once.

BOOL I2C_PWR_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".

I2C_ADC_Init

This function initializes I2C function of AD Converter depending on the specified clock speed. When the program run, call once. Others I2C ADC functions use after calling this function.

BOOL I2C_ADC_Init (int nBoard, DWORD nKHz)

Parameters:

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

nKHz : I2C communication speed (set the Clock speed.)
1: 100KHz, 2: 200KHz, 3: 300KHz, 4: 400KHz, Others: 100KHz

Return Value:

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

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

I2C_ADC_Read

This function reads a fixed value of specific address through I2C.

BOOL I2C_ADC_Read (int nBoard, Byte slAddr, DWORD nCnt, unsigned char* buf)

Parameters:

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

slAddr : I2C address

nCnt : The number of bytes of data read.

buf : Buffer pointer to read data.

Return Value:

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

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

I2C_ADC_Write

This function writes a fixed value of specific address through I2C.

BOOL I2C_ADC_Write (int nBoard, Byte slAddr, DWORD nCnt, unsigned char* buf)

Parameters:

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

slAddr : I2C address

nCnt : The number of bytes of data write.

buf : Buffer pointer to write data.

Return Value:

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

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

I2C_ADC_Close

The I2C function completely is finished. When the program finishes, call once.

BOOL I2C_ADC_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".

I2C_OS_Init

This function initializes I2C function of Open/Short module depending on the specified clock speed. When the program run, call once. Others I2C OS functions use after calling this function.

BOOL I2C_OS_Init (int nBoard, DWORD nKHz)

Parameters:

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

nKHz : I2C communication speed (set the Clock speed.)
1: 100KHz, 2: 200KHz, 3: 300KHz, 4: 400KHz, Others: 100KHz

Return Value:

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

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

I2C_OS_Read

This function reads a fixed value of specific address through I2C.

BOOL I2C_OS_Read (int nBoard, Byte slAddr, DWORD nCnt, unsigned char* buf)

Parameters:

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

slAddr : I2C address

nCnt : The number of bytes of data read.

buf : Buffer pointer to read data.

Return Value:

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

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

I2C_OS_Write

This function writes a fixed value of specific address through I2C.

BOOL I2C_OS_Write (int nBoard, Byte slAddr, DWORD nCnt, unsigned char* buf)

Parameters:

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

slAddr : I2C address

nCnt : The number of bytes of data write.

buf : Buffer pointer to write data.

Return Value:

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

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

I2C_OS_Close

The I2C function completely is finished. When the program finishes, call once.

BOOL I2C_OS_Close (int nBiard)

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".

I2C_SYS_Reset

This function initializes the I2C system resources of the system module.

BOOL I2C_SYS_Reset (void)

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".

I2C_SYS_Set_Clock

This function sets the clock of I2C system on the system module. (100KHz ~ 1MHz)

BOOL I2C_SYS_Set_Clcok (int nClock)

Parameters:

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

nClock : 1MHz : 1000000, 400KHz : 400000, 200KHz : 200000, 100KHz : 100000

Return Value:

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

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

I2C_SYS_Read

This function receives the data via I2C.

BOOL I2C_SYS_Read (Byte slAddr, DWORD nAddrLen, DWORD nAddr, DWORD nCnt, unsigned char* buf)

Parameters:

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

slAddr : Slave address

nAddrLen : Address Lengh

nAddr : Register Address

nCnt : Byte size of letter address
Maximum number of character that can be read is limited to the 256byte.

buf : Buffer Address.

Return Value:

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

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

I2C_SYS_Write

This function sends the data via I2C.

BOOL I2C_SYS_Write (Byte slAddr, DWORD nAddrLen, DWORD nAddr, DWORD nCnt, unsigned char* buf)

Parameters:

slAddr : Slave address

nAddrLen : Address Lengh

nAddr : Register Address

nCnt : Byte size of letter address.
Maximum number of character that can be sent is limited to the 1Kbyte(1024).

buf : Buffer Address

Return Value:

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

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

I2C_SYS_Read2

This function receives the data via I2C. Repeated start possible.

BOOL I2C_SYS_Read2 (int nBoard, Byte slAddr, DWORD nAddrLen, DWORD nAddr, DWORD nCnt, unsigned char* buf)

Parameters:

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

slAddr : Slave address

nAddrLen : Address Lengh

nAddr : Register Address

nCnt : Byte size of letter address
Maximum number of character that can be read is limited to the 256byte.

buf : Buffer Address.

Return Value:

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

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

I2C_SYS_Write2

This function sends the data via I2C. Read Back Mark possible.

BOOL I2C_SYS_Write2 (int nBoard, Byte slAddr, DWORD nAddrLen, DWORD nAddr, DWORD nCnt, unsigned char* buf)

Parameters:

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

slAddr : Slave address

nAddrLen : Address Lengh

nAddr : Register Address

nCnt : Byte size of letter address. Maximum number of character that can be sent is limited to the 1Kbyte(1024).

buf : Buffer Address

Return Value:

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

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

I2C_SYS_Read_Ex

This function reads a fixed value of specific address through I2C on the system module.

BOOL I2C_SYS_Read_Ex (int nBoard, Byte slAddr, DWORD nAddrLen, unsigned char* AddrBuf, DWORD nCnt, unsigned char* buf)

Parameters:

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

slAddr : Slave address

nAddrLen : Address Lengh

Addrbuf : Address Buffer

nCnt : Byte size of letter address
Maximum number of character that can be read is limited to the 256byte.

buf : Buffer Address.

Return Value:

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

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

I2C_SYS_Read2_Ex

This function reads a fixed value of specific address through I2C on the system module. Repeated start possible.

BOOL I2C_SYS_Read_Ex (int nBoard, Byte slAddr, DWORD nAddrLen, unsigned char* AddrBuf, DWORD nCnt, unsigned char* buf)

Parameters:

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

slAddr : Slave address

nAddrLen : Address Lengh

Addrbuf : Address Buffer

nCnt : Byte size of letter address
Maximum number of character that can be read is limited to the 256byte.

buf : Buffer Address.

Return Value:

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

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

I2C_SYS_Write_Ex

This function writes a fixed value of specific address through I2C on the system module.

BOOL I2C_SYS_Write_Ex (int nBoard, Byte slAddr, DWORD nAddrLen, unsigned char* AddrBuf, DWORD nCnt, unsigned char* buf)

Parameters:

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

slAddr : Slave address

nAddrLen : Address Lengh

AddrBuf : Address Buffer

nCnt : Byte size of letter address. Maximum number of character that can be sent is limited to the 1Kbyte(1024).

buf : Buffer Address

Return Value:

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

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

I2C_SYS_Write2_Ex

This function writes a fixed value of specific address through I2C on the system module. Read Back Mark possible.

BOOL I2C_SYS_Write_Ex (int nBoard, Byte slAddr, DWORD nAddrLen, unsigned char* AddrBuf, DWORD nCnt, unsigned char* buf)

Parameters:

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

slAddr : Slave address

nAddrLen : Address Lengh

AddrBuf : Address Buffer

nCnt : Byte size of letter address.
Maximum number of character that can be sent is limited to the 1Kbyte(1024).

buf : Buffer Address

Return Value:

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

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

SEN_Reset

This function controls the MIPI sensor reset signal.

BOOL SEN_Reset (int nBoard, BOOL bReset)

Parameters:

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

bReset : "0" : Disable, "1" : Enable

Return Value:

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

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

SEN_Enable

This function controls the MIPI sensor enable signal.

BOOL SEN_Enable (int nBoard, BOOL bEnable)

Parameters:

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

bEnable : "0" : Disable, "1" : Enable

Return Value:

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

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

SEN_Power

This function turns the 5V power On/Off.

BOOL SEN_Power (BOOL bOn)

Parameters:

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

addr : "0" : +5V Power Off, "1" : +5V Power On

Return Value:

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

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

Power Control API Functions

Overview

BOOL PWR_IO_Voltage (int nBoard, float fVoltage)
BOOL PWR_IO_On (int nBoard)
BOOL PWR_IO_Off (int nBoard)
BOOL I2C_SEN_On (int nBoard)
BOOL I2C_SEN_Off (int nBoard)
BOOL I2C_BOOT_Addr_Sel (int nBoard, BOOL addr)

PWR_IO_Voltage

After check the USB device exist, send voltage value using SendUserMsg function. (Minimum value is 1.25V)

BOOL PWR_IO_Voltage (int nBoard, float fVoltage)

Parameters:

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

Return Value:

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

PWR_IO_On

After check the USB device exist, enable the sensor power.

BOOL PWR_IO_On (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".

PWR_IO_Off

After check the USB device exist, disable the sensor power.

BOOL PWR_IO_Off (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".

I2C_SEN_On

After check the USB device exist, enable the sensor I2C power.

BOOL I2C_SEN_On (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".

I2C_SEN_Off

After check the USB device exist, disable the sensor I2C power.

BOOL I2C_SEN_Off (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".

I2C_BOOT_Addr_Sel

After check the USB device exist, control the EPROM power.

BOOL I2C_BOOT_Addr_Sel (BOOL addr)

Parameters:

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

addr : "0" : 5V power Off , "1" : keep the power.

Return Value:

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

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

SPI(Serial Peripheral Interface) API Functions

Overview

BOOL SPI_Send (int nBoard, DWORD dwLen, DWORD dwData0, DWORD dwData1)

BOOL FAN_Control (int nBoard, BOOL bOn, BYTE bySpeed)

SPI_Send

Transfer the data on the Serial Peripheral Interface bus.

BOOL SPI_Send (int nBoard, DWORD dWlen, DWORD dwData0, DWORD dwData1)

Parameters:

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

dWlen : Length of data

dwData0 : Transfer data 0

dwData1 : Transfer data 1

Return Value:

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

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

FAN_Control

This function turns the FAN on and off and controls the speed.

BOOL FAN_Control (int nBoard, BOOL bOn, BYTE bySpeed)

Parameters:

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

bOn : "1" : Fan On , "0" : Fan Off

bySpeed : FAN Speed

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

DAQ SYSTEM
Measurement & Automation