LIB USER Manual

Version：4.0

Release：24/10/2024

Contents

Introduction

This library provides a simple API to configure and operate USB2MPort devices. The library provides interface abstraction so that users can develop their application without any knowing about the usb. C libraries implementing the USB2MPort Interface Specification are provided for Windows XP or later and Ubuntu14.04 or later. Similarly, various demo project are provided to import library functions into VB,VC,labivew,QT and C++builder2009.

Approach to call API functions

There are two approaches to call API functions. One is for a process opening an USB2MPort device, the other is for one process opening more than one USB2MPort devices. One PC can open ten USB2MPort devices simultaneously . Steps show as below table.

step	Approach One	Approach Two
	(apply to a process open an USB2MPort)	(apply to a process open mutil-USB2MPort)
1	Call USBIO_SetUSBNotify to monitor the pull-out and plug-in of USB2MPort device	Call USBIO_SetUSBNotify to monitor pull-out and plug-in of USB2MPort device
2	Call USBIO_OpenDeviceto get device No.	Call USBIO_GetMaxNumofDev to get max number of USB2MPort that can be opened simultaneously.

3	If USB2MPort support trgiger,call USBIO_SetTrigNotify to monitor trigger event according to device No.	Call USBIO_GetSerialNo enum all the serial Number of USB2MPort device attched to PC.
4	Call other api function for read/write according to device No.	Select a serial number to as the parameter of USBIO_OpenDeviceByNumber to get device No.
5	Call USBIO_CloseDevice according to device No.	If USB2MPort support trgiger,call USBIO_SetTrigNotify to monitor trigger event according to device No.
6		Call other api function for read/write according to device No.
7		Call USBIO_CloseDevice to according to device No.
	Refer to VC demo project(SPI_RW)	Refer to VC demo project(I2c_RW)

API Functions

➢ USBIO_SetUSBNotify

Prototype	bool USBIO_SetUSBNotify(USB_DLL_CALLBACK pUSB_CallBack)
Description	This function sets a function pointer to lib. When monitored the pull-out or plug-in of USB2MPort device, pUSB_CallBack will be called.

Parameters	pUSB_CallBack	Function pointer，must define as bool function(BYTE iDevIndex, DWORD iDevStatus)
		iDevIndex：USB2MPort device No.
		iDevStatus：USB2MPort action ， 0x80,plug-in;0,pull-out
Return value	bool	Return true if successful

➢ USBIO_SetTrigNotify

Prototype	bool USBIO_SetTrigNotify(USB_DLL_CALLBACK pTrig_CallBack)			
Description	This function sets a function pointer to lib. When monitored the trigger event of USB2MPort device, pTrig_CallBack will be called.			
Parameters	pTrig_CallBack	Function pointer，must define as bool function(BYTE iDevIndex, DWORD iType)		
		iDevIndex：USB2MPort device No.		
		iType High16bit	unused	
		iType Low 16bit	0x00AA	IO Trigger
			0xXXB1	SPI Slaver receive trigger
Return value	bool	Return true if successful		

➢ USBIO_GetMaxNumofDev

Prototype	BYTE USBIO_GetMaxNumofDev(void)	
Description	This function returns the MaxNum of USB2MPort that can be opened simultaneously.	
Parameters	None	
Return value	BYTE	the MaxNum of USB2MPort

➢ USBIO_GetSerialNo

Prototype	BYTE USBIO_GetSerialNo(BYTE byIndex,char* lpBuff)	
Description	This function gets the serial number of USB2MPort and status according to device No.	
Parameters	byIndex	USB2MPort device No.
	lpBuff	a char pointer to save serial number

Return value	BYTE	0，USB2MPort no exist;
		1，USB2MPort unused;
		2, USB2MPort in using;
		Other: undefined

➢ USBIO_OpenDevice

Prototype	BYTE USBIO_OpenDevice(void)	
Description	This function opens a USB2MPort device and returns the device No.	
Parameters	None	
Return value	BYTE	If return is 0xFF, No USB2MPort can be opened

➢ USBIO_ResetDevice

Prototype	BYTE USBIO_ResetDevice(BYTE byIndex,BYTE byDevID)	
Description	This function resets the interface to initial setting	
Parameters	byIndex	USB2MPort device No.
	byDevID	Interface ID，see usbio.h
Return value	bool	Return true if successful

➢ USBIO_OpenDeviceByNumber

Prototype	BYTE USBIO_OpenDeviceByNumber(char* pSerialString)	
Description	This function opens the USB2MPort device with specified serial No. and returns device No.	
Parameters	pSerialString	A char pointer to save specified serial No.
Return value	BYTE	If return is 0xFF, it proves no USB2MPort can be opened

➢ USBIO_GetWorkMode

Prototype	bool USBIO_GetWorkMode (BYTE byIndex, BYTE*lpMode)	
Description	This function gets the work mode of USB2MPort device	
Parameters	byIndex	USB2MPort device No.
	lpMode	A byte to pointer to save the work mode of USB2MPort device If its value is 1, work at the upgrade mode If its value is 2, work at the normal mode
Return value	bool	Return true if successful

➢ USBIO_GetVersion

Prototype	bool USBIO_GetVersion(BYTE byIndex,BYTE byType,BYTE* lpBuffer)	
Description	This function gets the version number and build date/time	
Parameters	byIndex	USB2MPort device No.
	byType	Version selection; 0 for lib version information 1 for driver version information 2 for firmware version information
	lpBuffer	A byte pointer to save the information of version
Return value	bool	Return true if successful

➢ USBIO_CloseDevice

Prototype	bool USBIO_CloseDevice(BYTE byIndex)	
Description	This function closes the USB2MPort device according to device No.	
Parameters	byIndex	USB2MPort device No.
Return value	bool	Return true if successful

➢ USBIO_CloseDeviceByNumber

Prototype	bool USBIO_ CloseDeviceByNumber (char* pSerialString)	
Description	This function closes the USB2MPort device according to serial No.	
Parameters	pSerialString	A char pointer to save serial No.
Return value	bool	Return true if successful

➢ USBIO_I2cAutoGetAddress

Prototype	bool USBIO_I2cAutoGetAddress(BYTE byIndex,BYTE* pbyDevAddr)	
Description	This function gets the address of I2c slaver device that connected to USB2MPort device.	
Parameters	byIndex	USB2MPort device No.
	pbyDevAddr	A byte pointer to save slaver address. If its value is 0，No I2c slaver device find.
Return value	bool	Return true if successful
remark	I2c slaver address is at bit7~bit1，bit0 is R/W bit，don’t care	

➢ USBIO_I2cGetConfig

Prototype	bool USBIO_I2cGetConfig(BYTE byIndex,BYTE* pbyDevAddr,BYTE* pbyRate,DWORD* pdwMilliseconds)	
Description	This function gets the config of I2c port	
Parameters	byIndex	USB2MPort device No.
	pbyDevAddr	A byte pointer to save slaver address
	pbyRate	A byte pointer to save i2c frequency. If its value is 0, frequency is 100K If its value is 1, frequency is 200K If its value is 2, frequency is 300K If its value is 3, frequency is 400K If its value is 4, frequency is 800K
	pdwMilliseconds	A DWORD pointer to save i2c R/W timeout,unit:MS Low 16bit for read timeout; High 16bit for write timeout;
Return value	bool	Return true if successful
remark	I2c slaver address is at bit7~bit1，bit0 is R/W bit，don’t care	

➢ USBIO_I2cSetConfig

Prototype	bool USBIO_I2cSetConfig (BYTE byIndex,BYTE byDevAddr,BYTE byRate,DWORD dwMilliseconds)	
Description	This function sets the config of i2c port	
Parameters	byIndex	USB2MPort device No.
	byDevAddr	I2c slaver address

	byRate	I2c frequency index，0~4 represents 100K，200K，300K，400K，800k respectively.
	dwMilliseconds	I2c R/W timeout,unit: MS ; Low 16 bit for read timeout; High 16 bit for write timeout;
Return value	bool	Return true if successful
remark	I2c slaver address is at bit7~bit1，bit0 is R/W bit，don't care	

➢ USBIO_I2cRead

Prototype	bool USBIO_I2cRead(BYTE byIndex，BYTE byDevAddr，BYTE* lpParaBuffer，BYTE byParaSize，BYTE* lpReadBuffer,WORD wReadSize)	
Description	This function reads the data from i2c slaver .	
Parameters	byIndex	USB2MPort device No.
	byDevAddr	I2c slaver address
	lpParaBuffer	A byte pointer to save the command data send to i2c slaver before read, if none, set byParaSize = 0
	byParaSize	The length of command data
	lpReadBuffer	A byte pointer to save the data read from i2c slaver
	wReadSize	The length of data read from i2c slaver
Return value	bool	Return true if successful
remark	I2c slaver address is at bit7~bit1，bit0 is R/W bit，don't care	

➢ USBIO_I2cWrite

Prototype	bool USBIO_I2cWrite (BYTE byIndex，BYTE byDevAddr，BYTE* lpParaBuffer，BYTE byParaSize，BYTE* lpWrtieBuffer,WORD wWriteSize)	
Description	This function writes data to i2c slaver	
Parameters	byIndex	USB2MPort device No.
	byDevAddr	I2c slaver address
	lpParaBuffer	A byte pointer to save the command data send to i2c slaver before write, if none, set byParaSize = 0
	byParaSize	The length of command data
	lpWriteBuffer	A byte pointer to save the data write to i2c slaver
	wWriteSize	The length of data write to i2c slaver
Return value	bool	Return true if successful
remark	I2c slaver address is at bit7~bit1，bit0 is R/W bit，don't care	

➢ USBIO_I2cReadEEProm

Prototype	bool USBIO_I2cReadEEProm(BYTE byIndex，BYTE byDevAddr，BYTE byTypeIndex，DWORD dwOffset，BYTE* lpReadBuffer，WORD wReadSize)	
Description	This function reads data from eeprom	
Parameters	byIndex	USB2MPort device No.
	byDevAddr	I2c slaver address,here must be 0xA0
	byTypeIndex	EEPROM type index，0~12 represent 24C01~24C4096 respectively
	dwOffset	The start address of EEPROM

	lpReadBuffer	A byte pointer to save the data read from eeprom
	wReadSize	The length of data read from eeprom
Return value	bool	Return true if successful
remark	I2c slaver address is at bit7~bit1，bit0 is R/W bit，don’t care	

➢ USBIO_I2cWriteEEProm

Prototype	bool USBIO_I2cWriteEEProm(BYTE byIndex，BYTE byDevAddr，BYTE byTypeIndex，DWORD dwOffset，BYTE* lpReadBuffer，WORD wReadSize)	
Description	This function writes data to eeprom	
Parameters	byIndex	USB2MPort device No.
	byDevAddr	I2c slaver address,here must be 0xA0
	byTypeIndex	EEPROM type index，0~12 represent 24C01~24C4096 respectively
	dwOffset	The start address of EEPROM
	lpWriteBuffer	A byte pointer to save the data write to eeprom
	wWriteSize	The length of data write to eeprom
Return value	bool	Return true if successful
remark	I2c slaver address is at bit7~bit1，bit0 is R/W bit，don’t care	

➢ USBIO_SPIGetConfig

Prototype	bool USBIO_SPISetConfig (BYTE byIndex，BYTE * pbyRate，DWORD* pdwMilliseconds)	
Description	This function gets the config of SPI port	
Parameters	byIndex	USB2MPort device No.
	pbyRate	A byte pointer to save SPI config, its value shows as below: bit3~bit0 is SPI frequency index: 0~8 represent 200k，400k，600k，800k，1M，2M，4M，6M，12M respectively. bit5~bit4 is SPI mode: 00: SCK is low level in the idle state, the first edge sampled data in the SCK cycle. 01：SCK is high level in the idle state, the first edge sampled data in the SCK cycle. 10: SCK is low level in the idle state, the second edge sampled data in the SCK cycle. 11: SCK is high level in the idle state, the second edge sampled data in the SCK cycle. bit6 is unused bit7 is m/s : 0 as master; 1 as slaver
	pdwMilliseconds	A DWORD pointer to save SPI R/W timeout,unit:MS Low 16bit for read timeout; High 16bit for write timeout;
Return value	bool	Return true if successful

➢ USBIO_SPISetConfig

Prototype	bool USBIO_SPISetConfig (BYTE byIndex,BYTE byRate,DWORD dwMilliseconds)	
Description	This function sets the config of SPI port	
Parameters	byIndex	USB2MPort device No.
	byRate	Its value shows as below: bit3~bit0 is SPI frequency index: 0~8 represent 200k，400k，600k，800k，1M，2M，4M，6M， 12M respectively. bit5~bit4 is SPI mode: 00: SCK is low level in the idle state, the first edge sampled data in the SCK cycle. 01： SCK is high level in the idle state, the first edge sampled data in the SCK cycle. 10: SCK is low level in the idle state, the second edge sampled data in the SCK cycle. 11: SCK is high level in the idle state, the second edge sampled data in the SCK cycle. bit6 is unused bit7 is m/s : 0 as master; 1 as slaver
	dwMilliseconds	SPI R/W timeout,unit: MS ; Low 16 bit for read timeout; High 16 bit for write timeout;
Return value	bool	Return true if successful

➢ USBIO_SPITest

Prototype	bool USBIO_SPITest(BYTE byIndex,BYTE* lpWriteBuffer,BYTE* lpReadBuffer,BYTE byTestSize)	
Description	This function does the test of SPI-loop. MISO and MOSI must connect together first.	
Parameters	byIndex	USB2MPort device No.
	lpWriteBuffer	A byte pointer to save SPI data sent to MOSI
	lpReadBuffer	A byte pointer to save SPI data received from MISO
	byTestSize	The length of test data, must no more than 8
Return value	bool	Return true if successful

➢ USBIO_SPIRead

Prototype	bool USBIO_SPIRead(BYTE byIndex，BYTE* lpComBuffer，BYTE byComSize，BYTE* lpBuffer，WORD wBuffSize)	
Description	This function reads data from SPI device	
Parameters	byIndex	USB2MPort device No.
	lpComBuffer	A byte pointer to save command data send to SPI device before read. If none , byComSize should set to 0.
	byComSize	The length of command data.
	lpReadBuffer	A byte pointer to save the data read from SPI device
	wReadSize	The length of data to read from SPI device.
Return value	bool	Return true if successful

➢ USBIO_SPIWrite

Prototype	bool USBIO_SPIWrite(BYTE byIndex，BYTE* lpComBuffer，BYTE byComSize，BYTE* lpWriteBuffer,WORD wWriteSize)	
Description	This function writes data to SPI device	
Parameters	byIndex	USB2MPort device No.
	lpComBuffer	A byte pointer to save command data send to SPI device before write. If none , byComSize should set to 0.
	byComSize	The length of command data.
	lpWriteBuffer	A byte pointer to save the data write to SPI device
	wWriteSize	The length of data write to SPI device.
Return value	bool	Return true if successful

➢ USBIO_TrigGetConfig*

Prototype	bool USBIO_TrigGetConfig(BYTE byIndex,BYTE* pbySelect)	
Description	This function gets the interrupt type of IRQ pin.	
Parameters	byIndex	USB2MPort device No.
	pbySelect	A byte pointer to save IRQ interrupt type,its value shows as below: 0: raising trigger 1: falling trigger 2: high level trigger 3: low level trigger
Return value	bool	Return true if successful

➢ USBIO_TrigSetConfig*

Prototype	bool USBIO_TrigSetConfig(BYTE byIndex,BYTE bySelect)	
Description	This function sets the interrupt type of IRQ pin	
Parameters	byIndex	USB2MPort device No.
	bySelect	IRQ interrupt type, its value shows as below: 0: raising trigger 1: falling trigger 2: high level trigger 3: low level trigger
Return value	bool	Return true if successful

➢ USBIO_WaitForTrig*

Prototype	bool USBIO_WaitForTrig(BYTE byIndex)	
Description	This function enables the trigger of IQR pin.	
Parameters	byIndex	USB2MPort device No.
Return value	bool	Return true if successful

➢ USBIO_ExitTrig*

Prototype	bool USBIO_ExitTrig(BYTE byIndex)	
Description	This function disables the trigger of IRQ pin.	
Parameters	byIndex	USB2MPort device No.
Return value	bool	Return true if successful

➢ USBIO_SetCE*

Prototype	bool USBIO_SetCE(BYTE byIndex, bool bHigh)	
Description	This function set the level of CE1 output	
Parameters	byIndex	USB2MPort device No.
	bHigh	If true,set CE1 output high; if false, set CE1output low.
Return value	bool	Return true if successful

➢ USBIO_GetCE*

Prototype	bool USBIO_GetCE(BYTE byIndex, BYTE* pbyLevel)	
Description	This function gets the level of CE1 output	
Parameters	byIndex	USB2MPort device No.

	pbyLevel	A byte pointer to save CE1 level. Its value shows as below:
		1: CE1 output is high;
		0: CE1 output is low;

Return value	bool	Return true if successful

➢ USBIO_GetADCConfig*

Prototype	bool USBIO_GetADCConfig(BYTE byIndex，BYTE* pbyMask，BYTE* pbyIOSelect)	
Description	This function gets the config of ADC port	
Parameters	byIndex	USB2MPort device No.

	pbyMask	A byte pointer to save ADC channle switch, its value shows as below: bit7~bit4: unused bit3 : 1/0 ADC channel 4 on/off bit2 : 1/0 ADC channel 3 on/off bit1 : 1/0 ADC channel 2on/off bit0 : 1/0 ADC channel 1 on/off
	pbyIOSelect	A bytepointer to save the pin seletion of 4 ADC channels. The 1st byte is for ADC channel 1. The 2nd byte is for ADC channel 2. The 3rd byte is for ADC channel 3. The 4th byte is for ADC channel 4. High nibble of each byte is for positive pole selection of ADC channel . its value represents J7-03 ~J7-10,internal temperature sensor and VDD respectively. Low nibble of each byte is for negative pole selection of ADC channel . its value represents J7-03 ~J7-10,internal reference voltage and GND respectively.
Return value	bool	Return true if successful

➢ USBIO_SetADCConfig*

Prototype	bool USBIO_SetADCConfig(BYTE byIndex，BYTE byMask，BYTE* pbyIOSelect)	
Description	This function sets the config of ADC channels.	
Parameters	byIndex	USB2MPort device No.
	byMask	ADC channle switch, its value shows as below: bit7~bit4: unused bit3 : 1/0 ADC channel 4 on/off bit2 : 1/0 ADC channel 3 on/off bit1 : 1/0 ADC channel 2on/off bit0 : 1/0 ADC channel 1 on/off
	*byIOSelect	A byte pointer to save the pin selection of four ADC channels. The 1st byte is for ADC channel 1. The 2nd byte is for ADC channel 2. The 3rd byte is for ADC channel 3. The 4th byte is for ADC channel 4. High nibble of each byte is for positive pole selection of ADC channel . its value represents J7-03 ~J7-10,internal temperature sensor and VDD respectively. Low nibble of each byte is for negative pole selection of ADC channel . its value represents J7-03 ~J7-10,internal reference voltage and GND respectively.
Return value	bool	Return true if successful

➢ USBIO_ADCRead*

Prototype	bool USBIO_ADCRead(BYTE byIndex，WORD* lpReadBuffer，WORD wBuffSize)	
Description	This function reads the data sampled from ADC channels .	
Parameters	byIndex	USB2MPort device No.
	lpReadBuffer	A word pointer to save the data read from ADC channel.
	wBuffSize	The length of data read from ADC channel. wBuffSize must equal the multiples of ADC channels opened.
Return value	bool	Return true if successful

➢ USBIO_GetPWMConfig*

Prototype	bool USBIO_GetPWMConfig(BYTE byIndex，BYTE* pbyRate，BYTE* pbyNum，WORD*pwDuty)	
Description	This function gets the config of PWM port	
Parameters	byIndex	USB2MPort device No.
	pbyRate	A byte pointer to save PWM frequency index. Its value (0~10)represents 1k，2k，4k，6k，8k，10k，20k，40k，50k，60k，100k respectively.
	pbyNum	A byte pointer to save the status of PWM channels. Its value show as below:

		bit7: 1 / 0 PWM on/off bit6~0: 0 all the PWM channel off 1 PWM channel 1 on 2 PWM channel 1 ,2 on 3 PWM channel 1 ,2,3 on 4 PWM channel 1,2,3,4 on
	pwDuty	A word pointer to save the duty of each PWM channel. Duty unit: 0.001. The 1st word: the duty of PWM channel 1 The 2nd word: the duty of PWM channel 2. The 3rd word: the duty of PWM channel 3. The 4th word: the duty of PWM channel 4.
Return value	bool	Return true if successful

➢ USBIO_SetPWMConfig

Prototype	bool USBIO_SetPWMConfig(BYTE byIndex，BYTE byRate，BYTE byNum，WORD*pwDuty)	
Description	This function sets the config of PWM port	
Parameters	byIndex	USB2MPort device No.
	byRate	PWM frequency index，its value(0~10)repersents 1k，2k，4k，6k，8k，10k，20k，40k，50k，60k，100k respectively.

	byNum	the status of PWM channels. Its value show as below: bit7: 1 / 0 PWM on/off bit6~0: 0 all the PWM channel off 1 PWM channel 1 on 2 PWM channel 1 ,2 on 3 PWM channel 1 ,2,3 on 4 PWM channel 1,2,3,4 on
	pwDuty	A word pointer to save the duty of each PWM channel. Duty unit: 0.001. The 1st word: the duty of PWM channel 1 The 2nd word: the duty of PWM channel 2. The 3rd word: the duty of PWM channel 3. The 4th word: the duty of PWM channel 4.
Return value	bool	Return true if successful

➢ USBIO_StartPWM*

Prototype	bool USBIO_StartPWM(BYTE byIndex)	
Description	This function switches on PWM.	
Parameters	byIndex	USB2MPort device No.
Return value	bool	Return true if successful

➢ USBIO_StopPWM*

Prototype	bool USBIO_StopPWM(BYTE byIndex)	
Description	This function switches off PWM.	
Parameters	byIndex	USB2MPort device No.
Return value	bool	Return true if successful

➢ USBIO_GetGPIOConfig*

Prototype	bool USBIO_GetGPIOConfig(BYTE byIndex,BYTE* pbyValue)	
Description	This function gets the direction of GPIO.	
Parameters	byIndex	USB2MPort device No.
	pbyValue	A byte pointer to save GPIO direction. The bit0~bit7of its value represents J7-10~J7-03 respectively. 1/0: input/output
Return value	bool	Return true if successful

➢ USBIO_SetGPIOConfig*

Prototype	bool USBIO_SetGPIOConfig(BYTE byIndex,BYTE byValue)	
Description	This function sets the direction of GPIO port.	
Parameters	byIndex	USB2MPort device No.
	byValue	GPIO direction. The bit0~bit7of its value represents J7-10~J7-03 respectively. 1/0: input/output
Return value	bool	Return true if successful

➢ USBIO_GPIORead*

Prototype	bool　USBIO_GPIORead(BYTE byIndex,BYTE* pbyValue)	
Description	This function gets the levels of GPIO port.	
Parameters	byIndex	USB2MPort device No.
	pbyValue	A byte pointer to save GPIO levels. The bit0~bit7of its value represents J7-10~J7-03 respectively. 1/0: High level/low level.
Return value	bool	Return true if successful

➢ USBIO_GPIOWrite*

Prototype	USBIO_GPIOWrite(BYTE byIndex,BYTE byValue,BYTE byMask)	
Description	This function sets the levels of GPIO port.	
Parameters	byIndex	USB2MPort device No.
	byValue	GPIO levels. The bit0~bit7of its value represents J7-10~J7-03 respectively. 1/0: High level/low level.
	byMask	GPIO Mask，The bit0~bit7of its value represents J7-10~J7-03 respectively. If the bit is 1，the corresponding pin is only read.
Return value	bool	Return true if successful

Thread Safety

The USB2MPort library and associated functions are not thread safe. This means that calling library functions simultaneously from multiple threads may have undesirable effects.

To use the library functions in more than one thread, the user should do the following: Call library functions from within a critical section such that only a single function is being called at any given time. If a function is being called in one thread, then the user must prevent another thread from calling any function until the first function returns.