Pionway API

Pionway 模块的上位机应用程序编程接口，提供了基本的通信函数及一些基本的系统配置函数，用户软件通过调用这些函数可以实现 Pionway 模块的 PC 与 FPGA 之间的双向通信。

版权声明

修订记录

修订日期	修订内容
20170519	Rev.A最初版本
20220209	Rev.B更新模板，增加SetTimeout()函数

目录

使用说明

使用 Pionway API 开发应用程序时，您需要先将 PionwayDLL.h、Pionway.lib 和 Pionway.dll 这三个文件拷贝至工程目录下，然后在要使用这些函数的程序文件中包含 PionwayDLL.h，即#include “PionwayDLL.h”。例如，在 Visual Studio 工程中，您需要将以上三个文件拷贝至.vcxproj 文件所在目录中，然后在程序文件的头部加入#include “PionwayDLL.h”语句。

使用该库时，您需要先创建一个 CPionway 实例，然后调用 GetDeviceCount()方法确定有多少 Pionway 设备可用。接下来，您可以调用 GetDeviceListXXX()方法确定已连接的 Pionway 设备的具体信息，然后调用 OpenBySerial()方法根据指定的序列号打开某个 Pionway 设备。(如果没有提供序列号，则此方法将打开第一个可用设备)。

ConfigureFPGA()方法用于将配置文件下载到 FPGA。如果您已实例化了 Host Interface，并且正确配置了 MUX 和 Endpoint，那么接下来，您就可以在程序中调用 Wire、Trigger 和 Pipe 的相关方法与 Pionway 设备进行通信了。

数据结构

pwTDeviceInfo 结构体

描述 Pionway 设备的一些属性。(为公有属性)

deviceID

char deviceID[PW_MAX_DEVICE_ID_LENGTH]

用户可配置的设备 ID，最大字符串长度为 PW_MAX_DEVICE_ID_LENGTH，最后一个字符是 NULL 字符串结尾。

serialNumber

char serialNumber[PW_SERIAL_NUMBER_LENGTH]

Pionway 产品序列号，最大字符串长度为 PW_SERIAL_NUMBER_LENGTH，最后一个字符是 NULL 字符串结尾。

productName

char productName[PW_MAX_PRODUCT_NAME_LENGTH]

Pionway 产品名称，最大字符串长度为 PW_MAX_PRODUCT_NAME_LENGTH，最后一个字符是 NULL 字符串结尾。

productID

char productID[PW_MAX_PRODUCT_ID_LENGTH]

Pionway 的产品 ID，最大字符串长度为 PW_MAX_PRODUCT_NAME_LENGTH，最后一个字符是 NULL 字符串结尾。

usbSpeedMode

int usbSpeedMode

Pionway 设备当前运行的 USB 速度模式。

firmwareMajorVersion

int firmwareMajorVersion

Pionway 设备的固件主版本号。

firmwareMinorVersion

int firmwareMinorVersion

Pionway 设备的固件次版本号。

hdlMajorVersion

int hdlMajorVersion

Pionway 设备的 HDL 主版本号。

hdlMinorVersion

int hdlMinorVersion

Pionway 设备的 HDL 次版本号。

类成员数据数据

productID

标准的预定义值如下:

PW_BOARD_UNKNOWM = 0,
PW_BOARD_XMS6301_LX45 = 1,
PW_BOARD_XMS6301_LX75 = 2,
PW_BOARD_XMS6301_LX100 = 3,
PW_BOARD_XMS6301_LX150 = 4,
PW_BOARD_XMS6302_LX45 = 5,
PW_BOARD_XMS6302_LX75 = 6,
PW_BOARD_XMS6302_LX100 = 7,
PW_BOARD_XMS6302_LX150 = 8

usbSpeed

定义值如下:

PW_USBSPEED_UNKNOWN =0,
PW_USBSPEED_FULL =1,
PW_USBSPEED_HIGH =2,
PW_USBSPEED_SUPER =3

CPionway 类

CPionway 类方法

GetDeviceCount

CPionway::ErrorCode CPionway::GetDeviceCount(int *deviceCount)

参数:

[in] deviceCount 指向被分配的整形变量的指针，代表当前已连接且未被打开的设备数量。

返回:

NO_ERROR 操作成功。

FAILED 操作失败。

描述:

查询当前已连接且未被打开的设备数量。同时，该方法还建立了所有设备的序列号和型号的列表，随后用户可分别调用 GetDeviceListSerial()和 GetDeviceListModel()方法获得序列号和型号的列表。

OpenBySerial

CPionway::ErrorCode CPionway::OpenBySerial(std::string str = "")

参数:

[in] str 要打开的设备的序列号字符串。

返回:

NO_ERROR 操作成功。

FAILED 操作失败。

描述:

在与某个 Pionway 设备进行通信之前，必须先使用该方法打开此设备。如果此设备已被打开，则该设备将被关闭，直到再次打开该设备。

打开符合给定序列号字符串的 Pionway 设备。如果没有提供序列号（或空字符串），则打开第一个可用的设备。

SetTimeout

CPionway::ErrorCode CPionway::SetTimout(int timeout)

参数:

[in] timeout 以毫秒为单位的 Endpoint 传输超时时间。

返回:

NO_ERROR	操作成功。
INVALID_PARAMETER	输入参数错误
DEVICE_NOT_OPEN	Pionway 设备没有开启
FAILED	操作失败。

描述:

此方法设置 USB 事务在与目标设备通信时使用的超时值。

请注意，这个方法的设置作用域是全局的。默认情况下，超时设置为 10 秒。

timeout 通常应用于 Pipe 传输和 FPGA 配置传输。

GetDeviceListSerial

std::string CPionway::GetDeviceListSerial(int num)

参数:

[in] num 要获取序列号的设备编号(0~deviceCount-1)。

返回:

某个指定设备的序列号字符串。

描述:

获取某个已连接设备的序列号。在此之前，用户必须先调用 GetDeviceCount()方法，以获取当前已连接且未被打开的设备数量 deviceCount。

如果 num 不在设备数量的范围内，该方法将会返回一个空的字符串。该序列号可能会在随后被用来打开设备（通过调用 OpenBySerial()方法）。

GetDeviceListModel

CPionway::BoardModel　CPionway::GetDeviceListModel(int num)

参数:

[in] num　　要获取序列号的设备编号(0~deviceCount-1)。

返回:

某个指定设备的型号（枚举类型）。

描述:

获取某个已连接设备的型号。在此之前，用户必须先调用 GetDeviceCount()方法，以获取当前已连接且未被打开的设备数量 deviceCount。如果某个设备是未知设备或者 num 不在设备数量的范围内，该方法将会返回 BOARD_UNKNOWM。

GetDeviceInfo

CPionway::ErrorCode CPionway::GetDeviceInfo(Pionway_TDeviceInfo *info)

参数:

[in] info 指向 pwTDviceInfo 结构体的指针。

返回:

NO_ERROR	操作成功。
DEVICE_NOT_OPEN	设备未被打开。

描述:

使用设备的有关信息填充结构体 pwTDviceInfo。

IsPionwayEnabled

bool　CPionway:: IsPionwayEnabled ()

返回:

true	Pionway 设备在线。
false	Pionway 设备不在线。

描述:

该方法检查在 FPGA 设计中是否已经实例化了 Host Interface。 如果检测到，则 Pionway SDK 支持已被启用，Endpoint 功能可用。

ConfigureFPGA

CPionway::ErrorCode CPionway::ConfigureFPGA(const std::string strFilename)

参数:

[in]	strFilename	包含配置文件的文件名的字符串。

返回:

NO_ERROR	操作成功。
DEVICE_NOT_OPEN	设备未被打开。
FILE_ERROR	在打开或读取文件过程中发生错误。
INVALID_BITSTREAM	比特流文件（.bit）格式不正确。
DONE_NOT_HIGH	配置操作完成后，FPGA 的 DONE 信号没有被声明。
TRANSFER_ERROR	下载期间，USB 传输发生错误。
COMMUNICATION_ERROR	与固件通信时发生错误。
UNSUPPORTED_FEATURE	此设备或此配置中不支持配置调用。

描述:

此方法将配置文件下载到 FPGA。文件名应该是有效的 Xilinx 比特文件（从 bitgen 生成的.bit 文件）。

IsOpen

bool CPionway::IsOpen()

返回:

true	设备已被打开。
false	设备未被打开。

描述:

查询设备是否已被打开。

Close

CPionway::ErrorCode CPionway::Close()

返回:

NO_ERROR 操作成功。
DEVICE_NOT_OPEN 设备未被打开。

描述:

该方法可以用于关闭设备，以在系统级释放对应的设备，例如，在不销毁这个对象本身的前提下，允许另一个进程使用它，这样就可以保持该对象便于以后重新打开。

SetDeviceID

CPionway::ErrorCode CPionway::SetDeviceID(const std::string *str)

参数:

[in] str 新设备 ID 的字符串。

返回:

NO_ERROR 操作成功。
DEVICE_NOT_OPEN 设备未被打开。

描述:

此方法使用新的字符串修改 XMS 设备 ID 字符串。设备 ID 字符串是最多 32 个字符的用户可编程字符串，可用于标识特定的 XMS 设备。如果超过 32 个字符，该字符串将被截断，只保留前 32 个字符。

FlashEraseSector

CPionway::ErrorCode CPionway::FlashEraseSector (UINT32 address)

参数:

[in] address 要擦除的闪存扇区地址。

返回:

NO_ERROR	操作成功。
FAILED	操作失败。
DEVICE_NOT_OPEN	设备未被打开。
UNSUPPORTED_FEATURE	此设备或配置不支持此方法。
INVALID_PARAMETER	指定的地址对于擦除操作无效。
TIMEOUT	操作超时。

描述:

擦除系统闪存中的单个扇区。要擦除的扇区是由提供的地址决定的。扇区内的任何地址将指定整个扇区被擦除。指定的扇区必须在设备闪存用户区域的可接受范围内。有关闪存布局的详细信息，请参见指定设备的用户手册。

当操作完成时，该方法阻塞并返回。扇区的擦除时间不等，但每个扇区大约为 1 秒。
有关 Flash 布局和限制，请参阅指定设备的用户手册。

另请参见:

FlashWrite
FlashRead

FlashWrite

```
CPionway::ErrorCode CPionway::FlashWrite (    UINT32  address,
                                              UINT32  length,
                                              UINT8   *buf)
```

参数：

[in] address 要开始写操作的字节地址。

[in] length 要写入数据的长度（字节）。

[in] buf 指向要写入数据的缓存的指针。

返回：

NO_ERROR	操作成功。
FAILED	操作失败。
DEVICE_NOT_OPEN	设备未被打开。
UNSUPPORTED_FEATURE	此设备或配置不支持此方法。
INVALID_PARAMETER	指定的地址或长度无效。
TIMEOUT	操作超时。

描述：

从指定的地址开始将数据写入系统闪存。地址和长度必须都是系统闪存页面大小的整数倍。此方法不会擦除扇区，因此，在写入之前，您必须使用 FlashEraseSector 方法擦除扇区。

有关 Flash 布局和限制，请参阅指定设备的用户手册。

另请参见：

FlashEraseSector

FlashRead

FlashRead

```
CPionway::ErrorCode CPionway::FlashRead (   UINT32  address,
                                            UINT32  length,
                                            UINT8   *buf)
```

参数：

[in] address 要开始读操作的字节地址。

[in] length 要读取数据的长度（字节）。

[in] buf 指向读取数据的缓存的指针。

返回：

NO_ERROR	操作成功完成。
FAILED	操作无法完成。
DEVICE_NOT_OPEN	设备未被打开。
UNSUPPORTED_FEATURE	此设备或配置不支持此方法。
INVALID_PARAMETER	指定的地址或长度无效。
TIMEOUT	操作超时。

描述：

从指定地址开始读取系统闪存中的数据。地址和长度必须都是系统闪存页面大小的整数倍。

有关 Flash 布局和限制，请参阅指定设备的用户手册。

另请参见：

FlashEraseSector

FlashWrite

UpdateWireIns

CPionway::ErrorCode CPionway::UpdateWireIns()

返回：

NO_ERROR	操作成功完成。
DEVICE_NOT_OPEN	设备未被打开。
DEVICE_OFF_LINE	设备不在线。
TIMEOUT	操作超时。

描述：

此方法在所有 WireIn 值已使用 SetWireInValue()更新后调用。后者调用只更新类中的数据结构中保存的值。实际上，此方法同时提交对 XMS 的更改，以保证所有的 WireIn 会同时更新。

另请参见：

UpdateWireOuts
SetWireInValue
GetWireInValue
GetWireOutValue

SetWireInValue

```
CPionway::ErrorCode CPionway::SetWireInValue (  int epAddr,
                                                UINT32 val,
                                                UINT32 mask = 0xffffffff)
```

参数：

[in] epAddr 要设置的 WireIn 端点地址。

[in] val 要设置的 WireIn 新值。

[in] mask 应用于 WireIn 新值的掩码。

返回：

NO_ERROR	操作成功完成。
DEVICE_NOT_OPEN	设备未被打开。
INVALID_ENDPOINT	端点地址不在 WireIn 端点地址范围内。
DEVICE_OFF_LINE	设备不在线。
TIMEOUT	操作超时。

描述：

此方法用来设置某个指定地址的 WireIn 端点的值，并将其保存在内部数据结构中，然后在必要时通过调用 UpdateWireIn()方法进行更新。通过调用此方法，在每个端点的基础上更新这些值。另外，通过使用可选掩码，可以实现独立更新某些指定位，而不改变其它位的状态。例如：

```
* //  将地址为 0x03 的 WireIn 端点的值设置为 0x35。
* xem->SetWireInValue(0x03, 0x35);
* //将地址为 0x07 的 WireIn 端点的第 3 位的值设置为 1。
* xem->SetWireInValue(0x07, 0x04, 0x04);
* //  提交更新至 XMS。
* xem->UpdateWireIns();
```

另请参见：

UpdateWireOuts

UpdateWireIns

GetWireInValue

GetWireOutValue

GetWireInValue

```
CPionway::ErrorCode CPionway::GetWireInValue ( int      epAddr,
                                               UINT32   *val)
```

参数：

[in] epAddr 要设置的 WireIn 端点地址。

[in] val 指向获取的 WireIn 值的指针。

返回：

NO_ERROR	操作成功完成。
DEVICE_NOT_OPEN	设备未被打开。
INVALID_ENDPOINT	端点地址不在 WireIn 端点地址范围内。
DEVICE_OFF_LINE	设备不在线。
TIMEOUT	操作超时。

描述：

从 XMS 中获取特定的 WireIn 端点的值。

另请参见：

UpdateWireOuts

UpdateWireIns

GetWireOutValue

SetWireInValue

UpdateWireOuts

CPionway::ErrorCode CPionway::UpdateWireOuts()

返回：

NO_ERROR	操作成功完成。
DEVICE_NOT_OPEN	设备未被打开。
DEVICE_OFF_LINE	设备不在线。
TIMEOUT	操作超时。

描述：

此方法用来查询 XMS 的所有 WireOut 端点的当前值。所有 WireOut 会被同时捕获和读取。

另请参见：

UpdateWireIns
SetWireInValue
GetWireInValue
GetWireOutValue

GetWireOutValue

CPionway::ErrorCode CPionway::GetWireOutValue (int epAddr,
UINT32 *val)

参数:

[in] epAddr 要查询的 WireOut 端点地址。

[in] val 指向查询到的 WireOut 值的指针。

返回:

NO_ERROR	操作成功完成。
DEVICE_NOT_OPEN	设备未被打开。
INVALID_ENDPOINT	端点地址不在 WireOut 端点地址范围内。
DEVICE_OFF_LINE	设备不在线。
TIMEOUT	操作超时。

描述:

此方法用来获取某个指定地址的 WireOut 端点（Endpoint）的值。例如:

```
* //捕获 XMS 中的所有 WireOut 端点的状态
* XMS -> UpdateWireOuts();
*   //分别从地址为 0x21 和 0x27 的 WireOut 端点获取值。
* UINT32   a,b;
* XMS -> GetWireOutValue(0x21, &a);
* XMS -> GetWireOutValue(0x27, &b);
```

另请参见:

UpdateWireOuts
UpdateWireIns
GetWireInValue
SetWireInValue

ActivateTriggerIn

CPionway::ErrorCode CPionway::ActivateTriggerIn (int epAddr,
int bit)

参数:

[in] epAddr 要激活的 TriggerIn 端点地址。

[in] bit 要激活的 TriggerIn 端点的特定位。

返回:

NO_ERROR	操作成功完成。
DEVICE_NOT_OPEN	设备未被打开。
INVALID_ENDPOINT	端点地址不在 TriggerIn 端点地址范围内。
DEVICE_OFF_LINE	设备不在线。
TIMEOUT	操作超时。

描述:

此方法用来激活 XMS 中的某个指定地址的 TriggerIn 端点的某个特定位。

另请参见:

UpdateTriggerOuts

IsTriggered

UpdateTriggerOuts

CPionway::ErrorCode CPionway::UpdateTriggerOuts()

返回:

NO_ERROR	操作成功完成。
DEVICE_NOT_OPEN	设备未被打开。
DEVICE_OFF_LINE	设备不在线。
TIMEOUT	操作超时。

描述:

此方法用来捕获 XMS 中的所有 TriggerOut 端点的状态，以判定是否有任何一个 TriggerOut 被激活（自从上次调用此方法以后）。

另请参见:

ActivateTriggerIn

IsTriggered

IsTriggered

```
CPionway::ErrorCode CPionway::IsTriggered ( int      epAddr,
                                            UINT32   mask,
                                            bool     *is_triggered)
```

参数：

[in] epAddr　　要查询的 TriggerOut 端点的地址。
[in] mask　　要查询的 TriggerOut 端点的特定位。
[in] isTriggered　　指向 TriggerOut 的激活状态的指针。

返回：

NO_ERROR	操作成功完成。
DEVICE_NOT_OPEN	设备未被打开。
DEVICE_OFF_LINE	设备不在线。
TIMEOUT	操作超时。

描述：

此方法用来查询 XMS 中的某个指定地址的 TriggerOut 端点的某一位或某几位是否被激活（自从上次调用 UpdateTriggerOuts()方法以后）。例如：

```
* // 捕获 XMS 中的所有 TriggerOut 端点的状态。
* XMS -> UpdateTriggerOuts();
* // 查询地址为 0x61 的 TriggerOut 端点中的第 7 位是否被激活。
* bool   isTriggered;
* XMS -> IsTriggered(0x61, 0x80, &isTriggered);
```

另请参见：

ActivateTriggerIn
UpdateTriggerOuts

WriteToPipeIn

```
CPionway::ErrorCode CPionway::WriteToPipeIn ( int           epAddr,
                                              long          length,
                                              unsigned char *data,
                                              long          *count)
```

参数：

[in] epAddr 要写入数据的 PipeIn 端点地址。

[in] length 要写入的数据长度（字节）。

[in] data 指向存放要写入数据的缓存的指针。

[in] count 实际写入的数据量（字节）。

返回：

NO_ERROR	操作成功完成。
FAILED	操作无法完成。
DEVICE_NOT_OPEN	设备未被打开。
INVALID_ENDPOINT	端点地址不在 PipeIn 端点地址范围内。
DEVICE_OFF_LINE	设备不在线。
TIMEOUT	操作超时。

描述：

此方法将指定的数据缓存中的数据写入到给定地址的 PipeIn 端点。

长度（length）以字节为单位。固件限制了每次传输的最大长度。但是，如果需要，此方法将执行多次传输，以完成给定长度的传输。长度必须是最小传输大小的整数倍，具体如下：

- USB 3.0：16 的倍数

ReadFromPipeOut

```
CPionway::ErrorCode CPionway::ReadFromPipeOut ( int           epAddr,
                                                long          length,
                                                unsigned char *data,
                                                long          *count)
```

参数：

[in] epAddr 要读取数据的 PipeOut 端点地址。
[in] length 要读取的数据长度（字节）。
[in] data 指向存放读取数据的缓存的指针。
[in] count 实际读取的数据量（字节）。

返回：

NO_ERROR	操作成功完成。
FAILED	操作无法完成。
DEVICE_NOT_OPEN	设备未被打开。
INVALID_ENDPOINT	端点地址不在 PipeOut 端点地址范围内。
DEVICE_OFF_LINE	设备不在线。
TIMEOUT	操作超时。

描述：

此方法将数据从指定地址的 PipeOut 端点传输至给定的数据缓存。

长度（length）以字节为单位。固件限制了每次传输的最大长度。但是，如果需要，此方法将执行多次传输，以完成给定长度的传输。长度必须是最小传输大小的整数倍，具体如下：

- USB 3.0：16 的倍数

WriteToBlockPipeIn

```
CPionway::ErrorCode CPionway::WriteToBlockPipeIn ( int           epAddr,
                                                   int           blocksize,
                                                   long          length,
                                                   unsigned char *data,
                                                   long          *count)
```

参数：

[in] epAddr	要写入数据的 BlockPipeIn 端点地址。
[in] blocksize	数据块大小（字节）。
[in] length	要写入的数据长度（字节）。
[in] data	指向存放要写入数据的缓存的指针。
[in] count	实际写入的数据量（字节）。

返回：

NO_ERROR	操作成功完成。
FAILED	操作无法完成。
DEVICE_NOT_OPEN	设备未被打开。
INVALID_ENDPOINT	端点地址不在 BlockPipeIn 端点地址范围内。
INVALID_BLOCKSIZE	指定的 blocksize 无效。
DEVICE_OFF_LINE	设备不在线。
TIMEOUT	操作超时。

描述：

此方法将指定的数据缓存中的数据写入到给定地址的 BlockPipeIn 端点。

块大小（bolcksize）以字节为单位，取决于设备类型及工作模式：

- USB 3.0 FullSpeed：2 的幂[16…64]
- USB 3.0 HighSpeed：2 的幂[16…1024]
- USB 3.0 SuperSpeed：2 的幂[16…16384]

长度（length）以字节为单位。固件限制了每次传输的最大长度。但是，如果需要，此方法将执行多次传输，以完成给定长度的传输。长度必须是最小传输大小的整数倍，具体如下：

- USB 3.0：16 的倍数

ReadFromBlockPipeOut

```
CPionway::ErrorCode CPionway::ReadFromBlockPipeOut ( int           epAddr,
                                                     int           blocksize,
                                                     long          length,
                                                     unsigned char *data,
                                                     long          *count)
```

参数：

[in] epAddr	要读取数据的 BlockPipeOut 端点地址。
[in] blocksize	数据块大小（字节）。
[in] length	要读取的数据长度（字节）。
[in] data	指向存放读取数据的缓存的指针。
[in] count	实际读取的数据量（字节）。

返回：

NO_ERROR	操作成功完成。
FAILED	操作无法完成。
DEVICE_NOT_OPEN	设备未被打开。
INVALID_ENDPOINT	端点地址不在 BlockPipeOut 端点地址范围内。
INVALID_BLOCKSIZE	指定的 blocksize 无效。
DEVICE_OFF_LINE	设备不在线。
TIMEOUT	操作超时。

描述：

此方法将数据从指定地址的 BlockPipeOut 端点传输至给定的数据缓存。

块大小（bolcksize）以字节为单位，取决于设备类型及工作模式：

- USB 3.0 FullSpeed：2 的幂[16…64]
- USB 3.0 HighSpeed：2 的幂[16…1024]
- USB 3.0 SuperSpeed：2 的幂[16…16384]

长度（length）以字节为单位。固件限制了每次传输的最大长度。但是，如果需要，此方法将执行多次传输，以完成给定长度的传输。长度必须是最小传输大小的整数倍，具体如下：

- USB 3.0：16 的倍数