GPIO_PWN 说明

- GPIO_PWN 是一个 visual studio 解决方案, 包括项目 libgpiopwn (C/C++), libgpiopwn-clidemo (C/C++), GPIO_PWN-repl (Qt, 编译前需要安装 Qt 和 Qt VS 插件, 根据安装修改项目选项中的 Qt Project Settings > Qt Installation)
- libgpiopwn 是一个静态库, 也可以编译为 DLL (在 libgpiopwn 的项目属性里, General > Configuration Type 改为.dll, Preprocessor > Preprocessor Definitions 定义宏 **LIB_BC_BUILD_AS_DLL**) (或者取消 libgpiopwn.h 第 5 行的注释), DLL 版本主要是为了 C#的 P/Invoke, 如果是 C/C++程序, 请选择静态版本
- libgpiopwn 依赖 libusb, 编译应用程序前需要手动添加 libusb 的路径 (参考项目 libgpiopwn-clidemo 的项目设置)
- 对于 C#, 运行前把 libgpiopwn.dll 放到应用程序的同一目录. 根据.NET 项目在编译时的 Platform target 选择 32 位还是 64 位版本的 libgpiopwn.dll, Platform target 是 Any CPU 的话, 如果勾选了 Prefer 32-bit 就用 32 位版本的 libgpiopwn.dll, 否则使用 64 位. 如果 Platform target 是 x86 就用 32 位的, x64 就用 64 位
- 在 prjects 目录下各个项目目录中的 build\out\x64-Release 包含预编译的二进制文件

libgpiopwn 说明

- libgpiopwn 中包括 C 函数, 兼容 C++, 简而言之, *#include <libgpiopwn.h>*就可以了
- libgpiopwn 中的 *bool* 类型来自*<stdbool.h>*, 各种整数类型来自*<stdint.h>*
- 函数:
  - *bool libgpiopwn_Init()*
    - 初始化 libgpiopwn, 返回的 *bool* 代表初始化是否成功
  - *void libsiogpio_Deinit()*
    - 卸载 libgpiopwn
  - *void libsiogpio_SetTransferTimeout (uint32_t arg_timeout_ms)*
    - 设置与硬件通信的超时
  - *uint8_t libgpiopwn_GetFirmwareVersion ()*
    - 获取固件版本 (正常为 0x6e)
  - *uint16_t libgpiopwn_GPIO_GetAllInputChannelsLevel ()*

- *uint16_t libgpiopwn_GPIO_GetAllOutputChannelsLevel ()*
  - 获取所有 gpio 输入/输出通道当前的电平, 返回的 uint16_t 的第 1 位 (offset 0)代表第 1 通道, 第 2 位 (offset 1)代表第 2 通道
- *void libgpiopwn_GPIO_SetOutputChannelLevel (uint8_t arg_channel_order, bool arg_level)*
  - 设置 gpio 输出通道的电平, 第一个参数从 1 到 16 选择通道, 第二个参数 true 或 false 选择设置的高或低电平
- *uint8_t libgpiopwn_PWN_GetAllChannelsEndis ()*
  - 查看所有(共 4 个)pwn 通道的开启/关闭情况, 对应返回的 uint8_t 中的第 1 到 4 位
- *void libgpiopwn_PWN_EndisChannel (uint8_t arg_channel_order, bool arg_tof)*
  - 开启/关闭 pwn 通道, 第一个参数从 1 到 4 选择通道, 第二个参数选择开启或关闭
- *uint32_t libgpiopwn_PWN_GetChannelFrequency (uint8_t arg_channel_order)*
- *void libgpiopwn_PWN_SetChannelFrequency (uint8_t arg_channel_order, uint32_t arg_channel_frequency)*
  - 获取/设置 pwn 通道的频率, 第一个参数从 1 到 4 选择通道, 第二个参数选择频率, 单位位 Hz, 例如 1000Hz 的话第二个参数输入 1000, 不是频率都可以选择, 频率越高 pwn 的分辨率就越低, 如果第二个参数不可行, 固件会自动矫正为最临近的可行的频率
- *uint8_t libgpiopwn_PWN_GetChannelDutyRatio (uint8_t arg_channel_order)*
- *void libgpiopwn_PWN_SetChannelDutyRatio (uint8_t arg_channel_order, uint8_t arg_duty_ratio)*
  - 获取/设置 pwn 通道占空比, 第一个参数从 1 到 4 选择通道, 第二个参数从 0 到 100 选择设置的占空比
- *float libgpiopwn_ADC_GetChannelVoltage (uint8_t arg_channel_order)*
  - 读取 adc 通道读数, 第一个参数从 1 到 6 选择通道
- *uint8_t libgpiopwn_DAC_GetAllChannelsEndis ()*
  - 查看所有(共 2 个)dac 通道的开启/关闭情况, 对应返回的 uint8_t 中的第 1 到 2 位
- *void libgpiopwn_DAC_EndisChannel (uint8_t arg_channel_order, bool arg_tof)*
  - 开启/关闭 dac 通道, 第一个参数从 1 到 2 选择通道, 第二个参数选择开启或关闭
- *uint16_t libgpiopwn_DAC_GetChannelOutputVoltageWeight (uint8_t arg_channel_order)*

    - 获取 dac 通道输出电压的权重, dac 输出的电压为 3.3x(权重/4095)伏, 第一个参数从 1 到 2 选择通道
- *void libgpiopwn_DAC_SetChannelOutputVoltageWeight (uint8_t arg_channel_order, uint16_t arg_outvolt_weight)*
    - 设置 dac 通道输出电压的权重, 第一个参数从 1 到 2 选择通道, 第二个参数选择设置的权重, 从 0 到 4095