

GPIO Driver 使用方法

1. 文件包含

名称	修改日期	类型	大小
Binary	2019-08-19 16:17	文件夹	
Demo_Code	2019-08-19 16:18	文件夹	
Dll_Source_Code	2019-08-19 16:03	文件夹	

文件包含有：Binary、Demo_Code、Dll_Source_Code

1) Binaries 中包含 32 位/64 位的可调用文件，以及两个 C++编写的 Demo 可执行文件。如下：

名称	修改日期	类型	大小
GPIO_RW_Demo.exe	2019/8/19 16:11	应用程序	4,188 KB
SioGpioDriver64.dll	2019/8/19 15:51	应用程序扩展	2,771 KB
WinIo64.dll	2014/3/22 4:31	应用程序扩展	48 KB
WinIo64.sys	2014/3/22 4:31	系统文件	12 KB

其中，**请确保调用文件在执行时与可执行文件(.exe)位于同一目录**。另外，两种 EXE 分别是静态调用编译出的和动态调用编译出的。

2) Demo_Code 中包含调用函数的 C++例代码、此示例产生的可执行文件、Visual Studio 的编译设置。

名称	修改日期	类型
GPIO_RW_Demo_Dynamic	2019/8/19 16:18	文件夹
GPIO_RW_Demo_Static	2019/8/19 16:19	文件夹
Visual Studio 2015	2019/8/19 16:18	文件夹

名称	修改日期	类型
CurrentSettings.vssettings	2018/5/10 15:27	Visual Studio Set...

3) Source 中包含了 SioGpioDriver 的原工程(包含代码)、编译设置。

名称	修改日期	类型
SioGpioDriver	2019/8/19 15:04	文件夹
Visual Studio 2015	2019/8/19 13:56	文件夹

此为 API 的核心代码，如有需要，可自行修改。

2. 调用方式

1) C/C++静态调用。

```
5  #include "stdafx.h"
6  #include "GPIO_RW_Demo.h"
7  #include "GPIO_RW_DemoDlg.h"
8  #include "afxdialogex.h"
9  #include "SioGpioDriver.h" #1
10
11 #ifdef _M_X64
12     #pragma comment(lib, "rely/x64/SioGpioDriver.lib")
13 #else
14     #pragma comment(lib, "rely/x32/SioGpioDriver.lib")
15 #endif #2
16
17 #ifdef _DEBUG
18     #define new DEBUG_NEW
19 #endif
20
21
22 // CGPIO_RW_DemoDlg 对话框
23
24
25
26 CGPIO_RW_DemoDlg::CGPIO_RW_DemoDlg(CWnd* pParent /*=NULL*/)
27 : CDialogEx(IDD_GPIO_RW_DEMO_DIALOG, pParent)
28 {
29     m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
30 }
31
32 void CGPIO_RW_DemoDlg::DoDataExchange(CDataExchange* pDX)
33 {
34     CDialogEx::DoDataExchange(pDX);
35 }
36
37 BEGIN_MESSAGE_MAP(CGPIO_RW_DemoDlg, CDialogEx)
38     ON_WM_PAINT()
39     ON_WM_QUERYDRAGICON()
40     ON_BN_CLICKED(IDC_BUTTON1, &CGPIO_RW_DemoDlg::OnBnClickedButton1)
41     ON_BN_CLICKED(IDC_BUTTON2, &CGPIO_RW_DemoDlg::OnBnClickedButton2)
42     ON_BN_CLICKED(IDOK, &CGPIO_RW_DemoDlg::OnBnClickedOk)
43 END_MESSAGE_MAP()
44
45
46 // CGPIO_RW_DemoDlg 消息处理程序
47
48 BOOL CGPIO_RW_DemoDlg::OnInitDialog()
49 {
50     CDialogEx::OnInitDialog();
51
52     // 设置此对话框的图标。 当应用程序主窗口不是对话框时，框架将自动
53     // 执行此操作
54     SetIcon(m_hIcon, TRUE); // 设置大图标
55     SetIcon(m_hIcon, FALSE); // 设置小图标
56
57     // TODO: 在此添加额外的初始化代码
58     if (!SioGpioDriverOpen()) #3
59     {
```

```

156     rsv = SioGpioGetVal(PinNumber_hex);
157     if (-1 == rsv) #4
158     {
159         GetDlgItem(IDC_EDIT3)->SetWindowText(_T("Read Failed!"));
160         return;
161     }
162     if (((CButton*)GetDlgItem(IDC_RADIO1))->GetCheck())
163         value = 0;
164     else
165         value = 1;
166
167     if (SioGpioSetVal(PinNumber_hex, value))
168     {
169         if (value)
170             GetDlgItem(IDC_EDIT3)->SetWindowText(_T("Write bit to 1 OK !"));
171         else
172             GetDlgItem(IDC_EDIT3)->SetWindowText(_T("Write bit to 0 OK !"));
173     }
174 }
175
176
177 void CGPIO_RW_DemoDlg::OnBnClickedOk()
178 {
179     // TODO: 在此添加控件通知处理程序代码
180     SioGpioDriverClose(); #5
181     CDialogEx::OnOK();
182 }

```

- ① #1 引用.h 头文件，声明库中函数，注意，编译时要将文件放在合适的目录下。
- ② #2 调用静态库中函数的定义。
- ③ #3 调用启动函数，安装 Driver。
- ④ #4 使用库中的函数。
- ⑤ #5 程序运行结束前**务必**卸载 Driver。

2) C/C++动态调用。

```

6  typedef int(_stdcall *SioGpioDriverOpen) ();
7  typedef void(_stdcall *SioGpioDriverClose) ();
8
9  typedef int(_stdcall *SioGpioSetVal) (unsigned short PinNum, BYTE Value);
10 typedef int(_stdcall *SioGpioGetVal) (unsigned short PinNum);
11
12 HMODULE hlib;
13
14 SioGpioDriverOpen sioGpioDriverOpen = NULL;
15 SioGpioDriverClose sioGpioDriverClose = NULL;
16 SioGpioSetVal sioGpioSetVal = NULL;
17 SioGpioGetVal sioGpioGetVal = NULL;
18
19 typedef VOID(_stdcall *LPFN_PGNSI) (LPSYSTEM_INFO lpSystemInfo);
20
21 BOOL Is64Bit_OS() //not in use
22 {
23     BOOL bRetVal = FALSE;
24     SYSTEM_INFO si = { 0 };
25     LPFN_PGNSI pGNSI = (LPFN_PGNSI)GetProcAddress(GetModuleHandle(_T("kernel32.dll")), "GetNativeSystemInfo");
26     if (pGNSI == NULL)
27     {
28         return FALSE;
29     }
30     pGNSI(&si);
31     if (si.wProcessorArchitecture == PROCESSOR_ARCHITECTURE_AMD64 ||
32         si.wProcessorArchitecture == PROCESSOR_ARCHITECTURE_IA64)
33     {
34         bRetVal = TRUE;
35     }
36     return bRetVal;
37 }
38
39
40 int InitSioGpioLib(void)
41 {
42     #ifdef _M_X64
43         hlib = LoadLibrary(L"SioGpioDriver64.dll");
44     #else
45         hlib = LoadLibrary(L"SioGpioDriver32.dll");
46     #endif
47
48     sioGpioDriverOpen = (SioGpioDriverOpen)GetProcAddress(hlib, "SioGpioDriverOpen");
49     sioGpioDriverClose = (SioGpioDriverClose)GetProcAddress(hlib, "SioGpioDriverClose");
50     sioGpioSetVal = (SioGpioSetVal)GetProcAddress(hlib, "SioGpioSetVal");
51     sioGpioGetVal = (SioGpioGetVal)GetProcAddress(hlib, "SioGpioGetVal");
52
53     return sioGpioDriverOpen();
54 }
55
56
57 if (sioGpioSetVal(PinNumber_hex, value))
58 {
59     if (value)
60         GetDlgItem(IDC_EDIT3)->SetWindowText(_T("Write bit to 1 OK !"));
61     else
62         GetDlgItem(IDC_EDIT3)->SetWindowText(_T("Write bit to 0 OK !"));
63 }
64
65
66 void CGPIO_RW_DemoDlg::OnBnClickedOk()
67 {
68     // TODO: 在此添加控件通知处理程序代码
69     sioGpioDriverClose();
70     CDialogEx::OnOK();
71 }

```

- ① #1 声明一套函数类型，与所要调用的函数相同。
- ② #2 声明一个句柄，用于调用动态链接库。
- ③ #3 实例化刚刚声明的函数类型。
- ④ #4 使用句柄和“LoadLibrary()”函数来调用动态链接库文件。
- ⑤ #5 使用函数对象承接句柄中引用的函数
- ⑥ #6 程序开始时调用“开始函数”，启动并安装 Driver。
- ⑦ #7 使用寄存器读写函数，也可使用本库中的其它函数。
- ⑧ #8 程序结束时调用“关闭函数”，终止并卸载 Driver

3. IO 寄存器读写，实现读取 GPIO 状态和控制 GPO 的高低电平

GPIO 地址已映射于 IO 空间，只需访问 GPIO 的标号即可，比如访问 GPI15，调用函数时输入参数 15 即可。GPI（DI）寄存器只读，GPO（DO）寄存器可读写，规格书已写明 GPIO 是 GPI（DI）还是 GPO（DO），请见下表。

注意：下表示例仅作参考使用，用户须以具体产品规格书提供的 GPIO 地址对应关系调整 IO 映射空间，才能正确映射，从而正确访问。

功能	GPIO 名称	功能	GPIO 名称
DIO	SIO_GP15	DO0	SIO_GP47
DI1	SIO_GP16	DO1	SIO_GP51
DI2	SIO_GP33	DO2	SIO_GP64
DI3	SIO_GP37	DO3	SIO_GP65