How to use JS Automation I/O driver

1 Introduction

JS Automation I/O driver package provide the basic I/O system driver and Dll to link with programming language (such as Visual Basic、Visual C++、C++ Builder、LabVIEW…) for Windows 95/98 、Windows NT3.51/4.0、and Windows 2000 professional/Server platform.

To match the addressing mode of control cards, both 8 /16 bit direct address mode and 16 bit index mode are build-in functions, the user may adjust the jumper on the card and program as it is at your convenience.

Bit mode is added for the users to change or read any bit in byte or word data, it saves much time consuming work in the programming environment which is lack of bit manipulation function.

2 Function description of Dll

2.1 driver open / close (NT/2000)

- **JSDIOInit**

  Function: To initiate I/O Driver (on NT platform , you must call this function once to initialize driver, on Windows95/98 it is optional.)

  I/p parameters: none

  Return : 0 for success,1 for fail.

- **JSDIOClose**

  Function: To close I/O Driver (on NT platform , you must call this function once to initialize driver, on Windows95/98 it is optional.)

  I/p parameters: none

  Return : none

2.2 Direct I/O function

- **JSReadChar**

  Function: To read a byte(8 Bits) data

  I/p parameters: I/O a ddress(16 Bits) of the data to be read

  Return : Data(8Bits)

- **JSWriteChar**

  Function: To write a byte(8 Bits) data

  I/p parameters: (1) I/O a ddress(16 Bits) of the data to be written

  (2) Data(8 Bits) to be written

  Return : none

- **JSReadWord**

  Function: To read a word(16 Bits) data

  I/p parameters: I/O a ddress(16 Bits) of the data to be read

  Return : Data(16Bits)

- **JSWriteWord**

  Function: To write a word(16 Bits) data

I/p parameters: (1) I/O a ddress(16 Bits) of the data to be written
(2) Data(16 Bits) to be written

Return : none

- **JSReadLong**

  Function: To read a double word(32 Bits) data

  I/p parameters: I/O address(16 Bits) of the data to be read

  Return : Data(32Bits)

- **JSWriteLong**

  Function: To write a double word(32 Bits) data

  I/p parameters: (1) I/O address(16 Bits) of the data to be written
  (2) Data(32 Bits) to be written

  Return : none

2.3 Index I/O function

- **JSReadIntelligentWord**

  Function: To read a word(16 Bits) data

  I/p parameters: (1) I/O address(16 Bits) of the Index register
  (2) Index value(16 Bits)

  Return : Data(16Bits)

- **JSWriteIntelligentWord**

  Function: To write a word(16 Bits) data

  I/p parameters: (1) I/O address(16 Bits) of the Index register
  (2) Index value(16 Bits)
  (3) Data(16 Bits) to be written

  Return : none

- **JSReadIntelligentLong**

  Function: To read a double word(32 Bits) data

  I/p parameters: (1) I/O address(16 Bits) of the Index register
  (2) Index value(16 Bits)

  Return : Data(32Bits)

- **JSWriteIntelligentLong**

  Function: To write a double word(32 Bits) data

  I/p parameters: (1) I/O address(16 Bits) of the Index register
  (2) Index value(16 Bits)
  (3) Data(32 Bits) to be written

  Return : none

2.4 **Auxiliary fumction** (for bit maniplation)

- **GetBitFrByte**

  Function: To get a bit data from byte

  I/p parameters: (1) Source data(Byte)

(2) Bit position to be get, range from 0-7

Return : 0 for false, 1 for true

- **SetBitToByte**

  Function: To set a bit data to byte

  I/p parameters: (1) Source data(Byte)r

  (2) Bit position to be get, range from 0-7

  (3) Bit data to be set, range from 0-1

  Return : none

- **GetBitFrWord**

  Function: To get a bit data from word

  I/p parameters: (1) Source data(Word)r

  (2) Bit position to be get, range from 0-15

  Return : 0 for false, 1 for true

- **SetBitToWord**

  Function: To set a bit data to word

  I/p parameters: (1) Source data(Word)

  (2) Bit position to be get, range from 0-15

  (3) Bit data to be set, range from 0-1

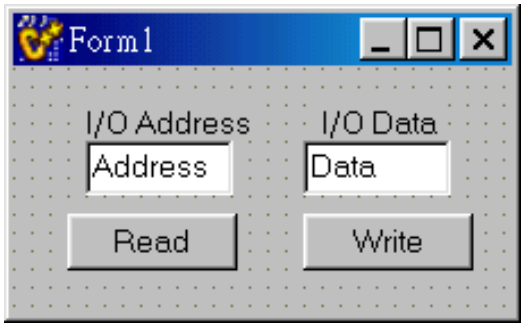
  Return : none

3 Application examples

3.1 Use Dll inVisual Baisc

- Step1 Open a new project

  Setup a new form



- Step 2 Declare functions

  In Module(.bas) needs the following statements:

```
Public Declare Function JSDIOInit Lib "jsioport.dll" () As Byte
Public Declare Sub JSDIOClose Lib "jsioport.dll" ()
Public Declare Function JSReadChar Lib "jsioport.dll" (ByVal Port As Integer) As Byte
Public Declare Sub JSWriteChar Lib "jsioport.dll" (ByVal Port As Integer, ByVal Data As Byte)
```

- Step 3 Initiate I/O resource in form_load

  To initiate I/O resource in Form_Load() as follows:(Windows 95/98 does not required)

```
Private Sub Form_Load()
JSDIOInit
End Sub
```

- Step 4 Add program code

  for example:To add a Read button:

```
Private Sub Read_Click()
Data = JSReadChar(Address)
End Sub
```

  To add a Write button:

```
Private Sub Write_Click()
JSWriteChar Address, Data
End Sub
```

- Step 5 Release the resource before program close

  Release I/O resource in Form_ Unload () (Windows 95/98does not required)

```
Private Sub Form_Unload(Cancel As Integer)
JSDIOClose
End Sub
```

3.2 Use dll in C++ Builder

- Step 1 Generate .lib file from .dll file

  Use the utility program **implib.exe** to generate the .lib file from dll.:

  implib jsioport.lib jsioport.dll

- Step 2 Open a new project

  Setup a new form

- Step 3 Add jsioport.lib into project

- Step 4 Declare functions in .h file

```
extern "C" __declspec(dllimport)int __stdcall JSDIOInit();
extern "C" __declspec(dllimport)void __stdcall JSDIOClose();
extern "C" __declspec(dllimport)char __stdcall JSReadChar(int Address);
extern "C" __declspec(dllimport)void __stdcall JSWriteChar(int Address,char Value);
```

- Step 5 Initiate I/O resource

```
void __fastcall TForm1::FormActivate(TObject *Sender)
{
    JSDIOInit();
}
```

- Step 6 Add program code

  For example: To add Readbutton

```
void __fastcall TForm1::ReadClick(TObject *Sender)
{
    char data= JSReadChar(StrToInt(Address->Text));
    Data->Text=IntToStr(data);
}
```

  To add Write button:

```
void __fastcall TForm1::WriteClick(TObject *Sender)
{
    JSWriteChar(StrToInt(Address->Text),StrToInt(Data->Text));
}
```

- Step 7 Release I/O resource before program close

Release I/O resource in FormClose(Windows 95/98 does not required)

```
void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{
    JSDIOClose();
}
```

3.3 Use dll in Visual C++

- Step 1 Open a new project

  Setup a new form

- Step 2 Add jsioport.lib into project

- Step 3 Declare functions

  Add #include "jsioport.h" in program source file

- Initiate I/O resource

```
BOOL CProject1Dlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    ....
    JSDIOInit();
    return TRUE;    // return TRUE    unless you set the focus to a control
}
```

- Step 6 Setup parameters

```
CProject1Dlg::CProject1Dlg(CWnd* pParent /*=NULL*/)
    : CDialog(CProject1Dlg::IDD, pParent)
{
    m_address = 0;
    m_Data = 0;
}
void CProject1Dlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    DDX_Text(pDX, IDC_Address, m_address);
```

```
    DDX_Text(pDX, IDC_Data, m_Data);
}
```

- Step 7 Add message map

```
BEGIN_MESSAGE_MAP(CProject1Dlg, CDialog)
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDRead, OnRead)
    ON_BN_CLICKED(IDWrite, OnWrite)
    ON_WM_CLOSE()
END_MESSAGE_MAP()
```

- Step 8 Add program code

  For example: Add Read button

```
void CProject1Dlg::OnRead()
{
    UpdateData(TRUE);
    m_Data=JSReadChar(m_address);
    UpdateData(FALSE);
}
```

  Add Write button

```
void CProject1Dlg::OnWrite()
{
    // TODO: Add your control notification handler code here
    UpdateData(TRUE);
    JSWriteChar(m_address,m_Data);
    UpdateData(FALSE);
}
```

- Step 9 Release I/O resource

  Release I/O resource before program close(Windows 95/98 does not required)

```
void CProject1Dlg::OnClose()
{
    // TODO: Add your message handler code here and/or call default
    JSDIOClose();
    CDialog::OnClose();
}
```