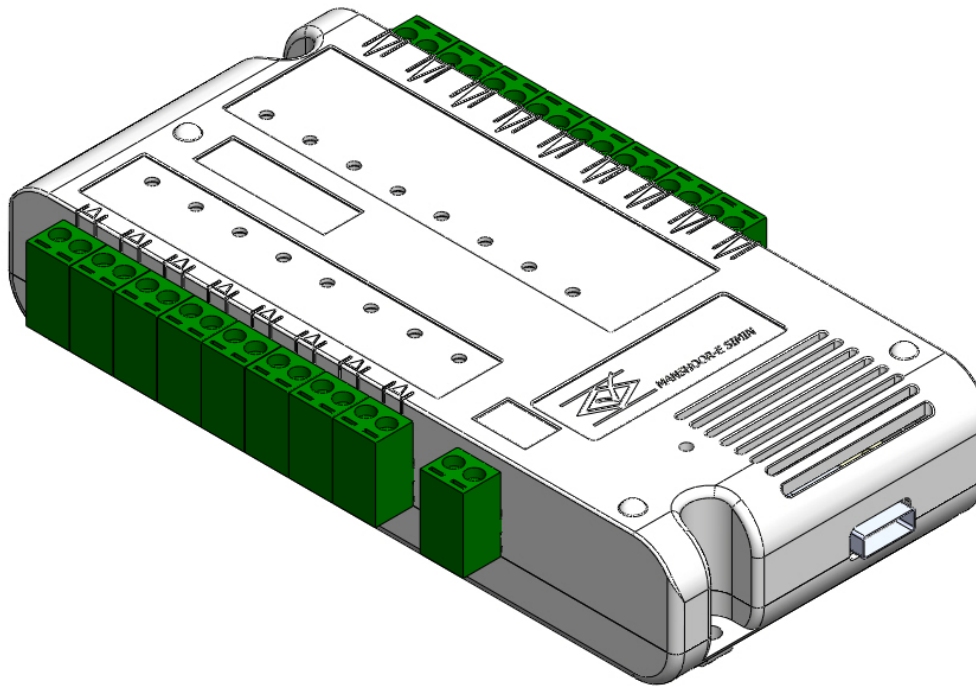


NANO Input/Output 8 Channels



**I/O Interface from a PC's USB port
Installation and Users Manual
&
Software Guide**

**Available exclusively from
Manshoore Simin Ltd Co.
www.msbbs.com**

Copyright ©2011 Manshoore Simin Ltd Co.

Contents

1. Introduction
2. Getting Started
3. Electronic Schematic
4. Software Sample
5. Software Options
 - 5.1. Output
 - 5.2. Input
 - 5.3. WD_Enable(Watch Dog Enable).
 - 5.4. Delay_WD(Delay Watch Dog)
 - 5.5. Delay_AfterWD(Delay After Watch Dog)
 - 5.6. Outchange(only for After Watch Dog)
 - 5.7. Output Status After WD(Output Status After Watch Dog)
 - 5.8. Output Status After Reset
 - 5.9. Set_Setting
 - 5.10. Get_Setting
 - 5.11. Change USB ID
6. Writing your own software for NANO IO Interface
 - 6.1. Using NANO IO Interface by Delphi.

1. Introduction

The Nano Digital I/O Board Kit enables your PC application to allowing you to read up to 8 optically isolated inputs and control up to 8 isolated outputs. Inputs can be a voltage from 5-12VDC and outputs can be either mechanical or solid state relays that can switch up to 300MA loads, such as cooling fans, solenoids, heating elements and more.

Simple parallel input/output control for ease of use or you can use the interface to minimize I/O pin usage. The logic circuits operate from 3.3V to 5V making them compatible with PC USB port variable voltage range.

The microcontroller in the kit, is used for industrial environments.

Inputs can be configured to handle a different range of input voltages by changing a couple of components.

2. Getting Started

By connecting IO Interface to USB PC port, the HID device driver will install automatically. In other word, windows will automatically recognize it and configure it's driver accordingly; Windows 2000, XP, Vista and Windows 7, support the device.

After install, insert the device CD into CD drive and go to CD root, then copy the root folder in to your hard disk and execute the "File Register" exe file and click on "Register File" button and select "MSUSBI.ocx" then ok if the file registers successfully. Then execute "HID_Device_Sample.exe".

3. Electronic Schematic

The NANO IO Interface has 8 input channels and 8 output channels. Each channel has 2 pin which are used to makes connections to the external device.

Speed rate for this product is 10 ms in get input or set output.

This do not need any Power Supplier, rather is supplied from PC USB Port. Designed into the interface that USB voltage first convert to 12V then regulate to 5V for reason, the voltage level do not dependent on USB voltage.

Figure1 illustrates the interface pin out.

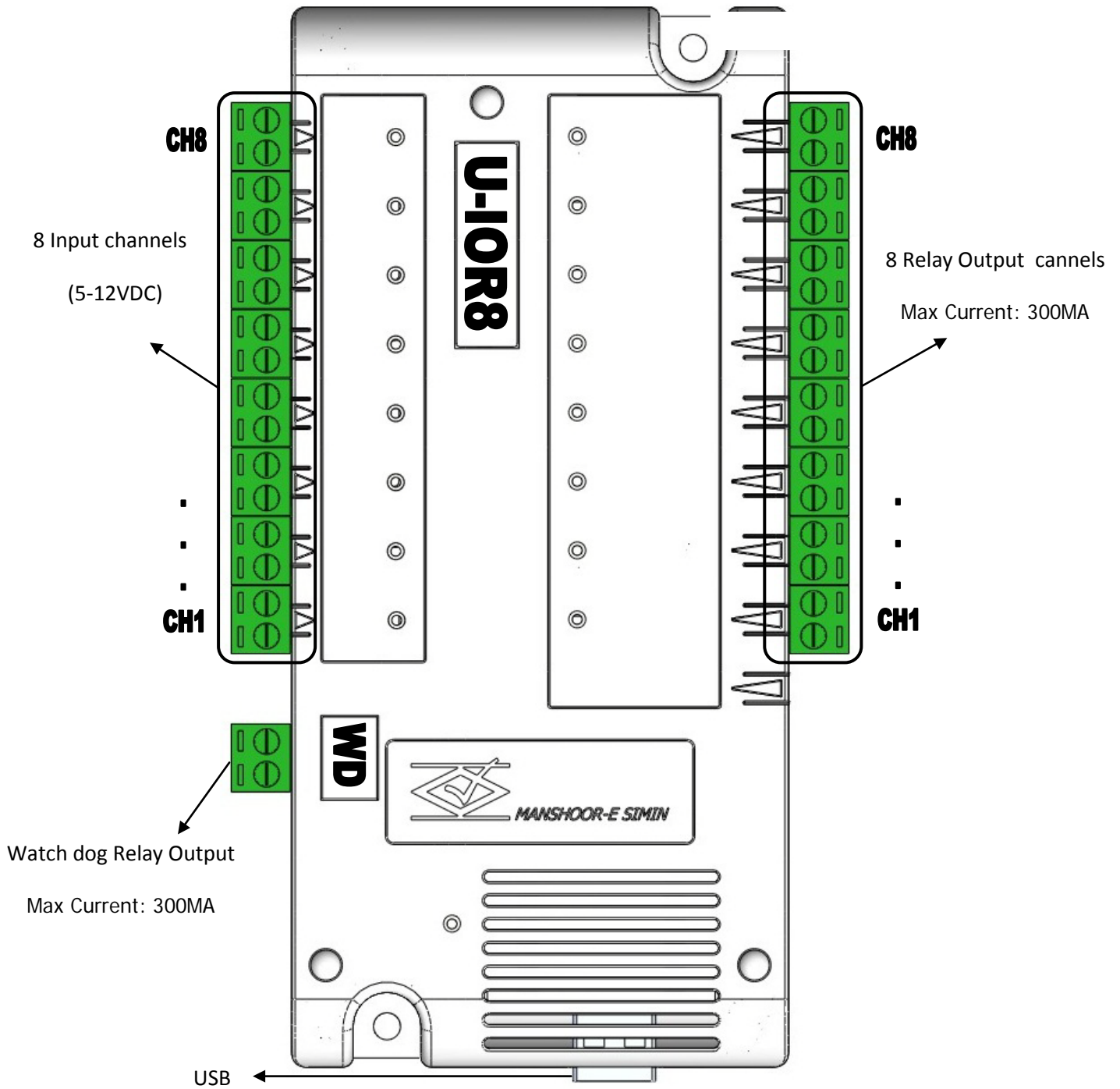


Figure 1

Pin	Description
Inputs	8 isolated channels with opto coupler; Voltage range: 5-12 Volte DC
Outputs	8 mechanical relay channels; Max current: 300MA
Watch Dog	Mechanical relay output; Max current: 300MA

4. Software Sample

The Nano IO Interface is supplied with this ready sample which makes it very easy for the beginner to get quickly up and running with device control.

Installation, as described above, is painless and easy requiring only a Windows 2000, XP, Vista or Windows7, computer with fairly modest specifications. To run the software double click on the "HID_Device_Sample.exe". The "HID_Device_Sample" environment screen will then appear providing the workspace for doing operation. By running the software, last interface will be selects automatically that indicated in the "Select Interface" combo box. User can selects the interface from combo box or insert device index and then click on "Select Device" button.

By click on "Device Name" button, after insert index of the device, the interface name show in the "Name" edit box.

To show number of interfaces that connect to USB port, click on "Device Count".

When the interface would be connect or disconnect on USB port, click on "Update Device" button to refresh list of interfaces. (figure2)

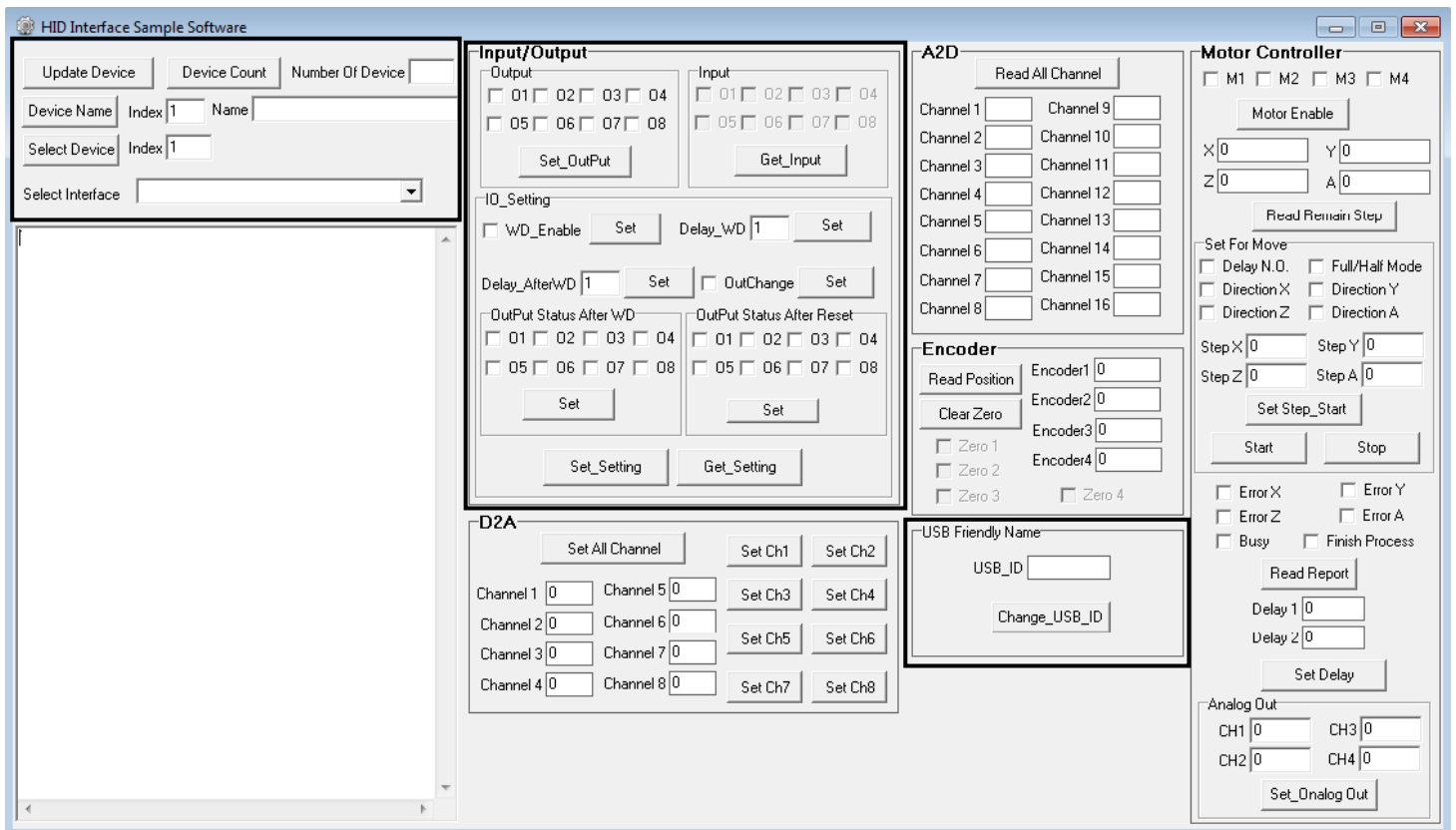


Figure 2

5. Software Options

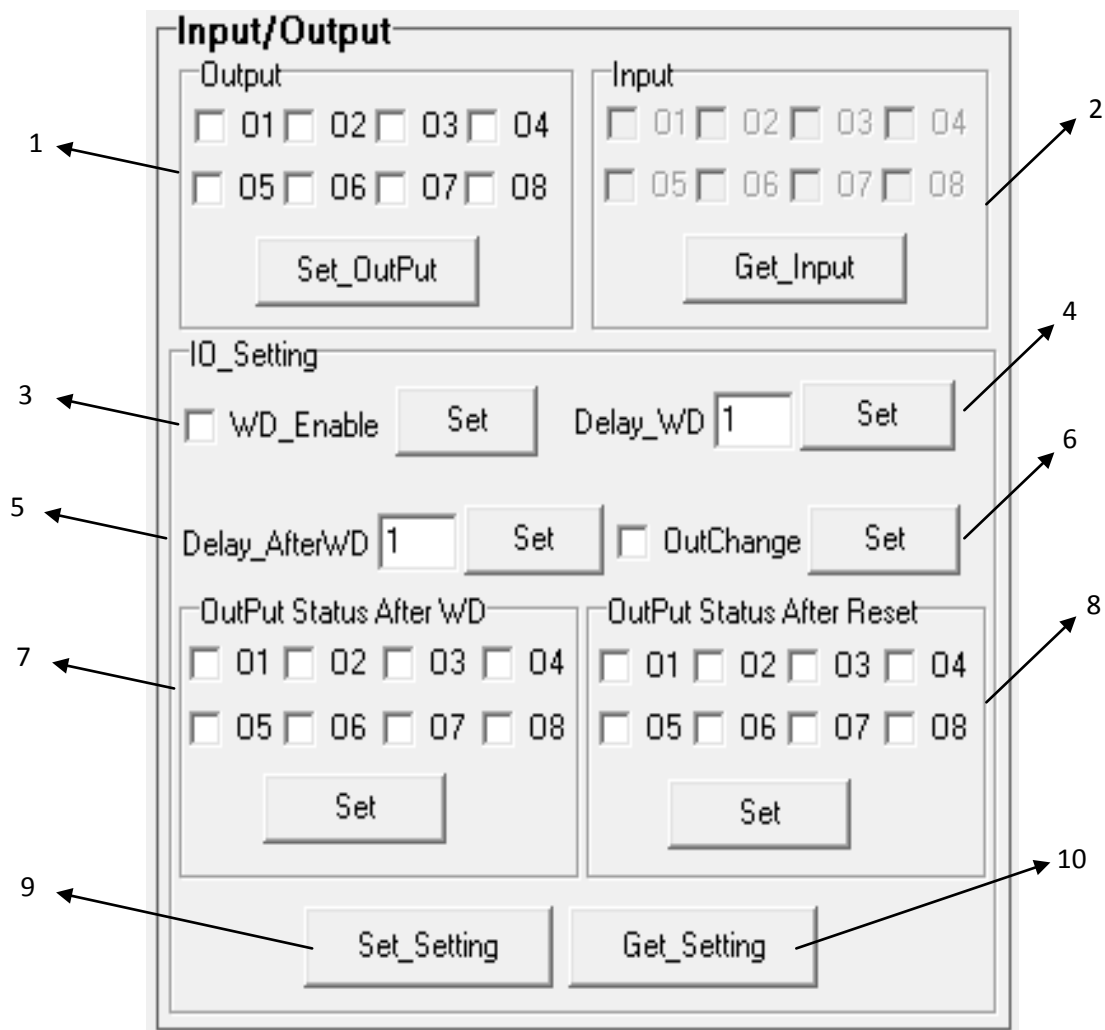


Figure 3

5.1. Output

To connect or disconnect output channels separately, have to check or uncheck "01"... "08" check box then click on the "Set_Output" Button.

5.2. Input

To check of Input Channels, click on the "Get_Input" button .

5.3. WD_Enable (Watch Dog Enable).

Watch dog output relay designed in the kit, user can enable or disable it, for enabling watch dog , user must be set a timer before that, relay be connect and set a timer after relay connection, that after pass this timer the relay be disconnect.

Setting these two timers (before and after relay connection) described below.

5.4. Delay_WD (Delay Watch Dog)

This time be set before enabling watch dog, that after this time watch dog relay will be connect; for set this timer click on "Set" button in front of Delay_WD.

5.5. Delay_AfterWD (Delay after Watch Dog)

This time be set after enabling watch dog , that after this time watch dog relay that, would be connect, will be disconnect. For set this timer click on "Set" button in front of "Delay_AfterWD".

5.6. Outchange (only occur after Watch Dog)

If out change be checked and set with click on "Set" button in front of "Out change" then output channels will be change after that, occurring watchdog, to status that, user defined in "Output Status AfterWD" panel else after occurring watch dog, output channels status do not change.

5.7. Output Status after WD (Output Status after Watch Dog)

By check or uncheck "O1"... "O8" check box and click on "Set" button in this section, if "OutChange" would be enable, after occurring watch dog, output channels status, convert to this status.

5.8. Output Status after Reset

By check or uncheck "O1"... "O8" check box and click on "Set" button in this section, after turn on or reset the device, output channels status, convert to this status.

5.9. Set_Setting

After changing all of these setting that, described above, by click on "Set_Setting " button all of these setting will be apply.

5.10. Get_Setting

By click on the "Get_Setting" button all of these setting that described, will be read and show.

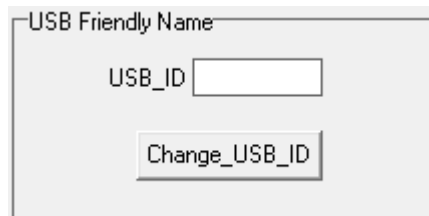


Figure 4

5.11. Change USB ID(Figure 4)

By connecting more than, one similar interface type into USB port, all of these interfaces have only one friendly name in the list, for distinct of these similar names, user can change ID of the interface, default ID for these interfaces is 0000. User can change these four characters. For example if USB friendly name in the device list is "M.S.D/A 0000", user can change "0000" to any character like "Test", after this change, the interface must be disconnect and then connect to the USB port, by connecting , USB friendly name in the device list change to "M.S.D/A Test". With this change user can distinct between two similar interfaces.

6. Writing your own software for NANO IO Interface

Provided with this interface an ocx(Active X) file, called "MSUSBI.ocx".. This file encapsulates the functions used by sample software in communicating with IO interface across the USB interface into any simple functions easily understood and used in custom software. Although the ocx is written in "Delphi" it can be used (called) by programs written in a number of popular languages, the popular of

which are Visual BASIC, C#, Visual C. Below described are the techniques to use the ocx in Delphi and Visual Basic. If you program in another language please refer to your compiler manual on the details of importing and calling a ocx functions, which will be very similar to the techniques described below.

6.1. Using NANO IO Interface by Delphi.

Using NANO IO Interface with your own programs written in Delphi is very simple. At the head of your program, before using any of the NANO IO Interface function, you must Import "MSUSBI.ocx" in your programming language and insert the ocx in your program form....

//If Connect or disconnect a interface to USB port call this function//

```
MSUSBDevice1.DeviceUpdate();  
////////////////////////////////////
```

//Check how many device is connect to USB port//

```
integer := MSUSBDevice1.DeviceCount();  
////////////////////////////////////
```

// Get friendly name of USB interface//

```
string := MSUSBDevice1.ListOfDevice(Device index:integer);  
////////////////////////////////////
```

//Select the interface via Device number before doing any work//

```
if not MSUSBDevice1.SelectDevice(Device NO:integer) then  
  ShowMessage('Device Could not Select');  
////////////////////////////////////
```

//Set Output //

```
if not MSUSBDevice1.IO_Set_OutPut(Output1, Output2, Output3, Output4, Output5, Output6,  
Output7, Output8:Boolean) then  
  Memo1.Lines.Add('Failed To Send.');
```

//Get Input //

```
if not MSUSBDevice1.IO_Get_Input(I1,I2,I3,I4,I5,I6,I7,I8:Boolean) then  
  Memo1.Lines.Add('Failed To Send.')
```

else
begin
 Input1 := I1;
 Input2 := I2;
 Input3 := I3;
 Input4 := I4;
 Input5 := I5;
 Input6 := I6;
 Input7 := I7;
 Input8 := I8;

```
end;  
////////////////////////////////////
```

//WD Enable //

If not MSUSBDevice1.IO_WD_Enable(WD_Enable:Byte) then
Memo1.Lines.Add('Failed To Send.');

//Delay WD/

if not MSUSBDevice1.IO_Delay_WD(Delay:Byte) then
Memo1.Lines.Add('Failed To Send.');

//Delay After WD //

if not MSUSBDevice1.IO_Delay_AfterWD(DelayAfter:Byte) then
Memo1.Lines.Add('Failed To Send.');

//Out Change //

If not MSUSBDevice1.IO_OutChange_AfterWD(OutChange_AfterWD:Byte) then
Memo1.Lines.Add('Failed To Send.');

//Output Status After WD//

if not MSUSBDevice1.IO_Preset_WD(O1,O2,O3,O4,O5,O6,O7,O8:Byte) then
Memo1.Lines.Add('Failed To Send.');

//Output Status After Reset//

if not MSUSBDevice1.IO_Preset_OutPut(O1,O2,O3,O4,O5,O6,O7,O8:Byte) then
Memo1.Lines.Add('Failed To Send.');

//Set Setting //

POut := POut + (OutPut1Reset and \$01):Byte;
POut := POut + (OutPut2 Reset and \$01)*2:Byte;
POut := POut + (OutPut3Reset and \$01)*4:Byte;
POut := POut + (OutPut4Reset and \$01)*8:Byte;
POut := POut + (OutPut5Reset and \$01)*16:Byte;
POut := POut + (OutPut6Reset and \$01)*32:Byte;
POut := POut + (OutPut7Reset and \$01)*64:Byte;
POut := POut + (OutPut8 Reset and \$01)*128:Byte;

POutAfterWD := POutAfterWD + (OutPut1AfterWD and \$01):Byte;
POutAfterWD := POutAfterWD + (OutPut2AfterWD and \$01)*2:Byte;
POutAfterWD := POutAfterWD + (OutPut3AfterWD and \$01)*4:Byte;
POutAfterWD := POutAfterWD + (OutPut4AfterWD and \$01)*8:Byte;
POutAfterWD := POutAfterWD + (OutPut5AfterWD and \$01)*16:Byte;
POutAfterWD := POutAfterWD + (OutPut6AfterWD and \$01)*32:Byte;
POutAfterWD := POutAfterWD + (OutPut7AfterWD and \$01)*64:Byte;
POutAfterWD := POutAfterWD + (OutPut8AfterWD and \$01)*128:Byte;

if not MSUSBDevice1.IO_Set_Setting(Delay, DelayAfterWD, OutChange_AfterWD,

```
POutAfterWD, POut, WD_Enable:Byte) then  
Memo1.Lines.Add('Failed To Send.');
```

//Get Setting //

```
if not MSUSBDevice1.IO_Get_Setting(Delay_WD, Delay_AfterWD,  
OutChange_AfterWD, Preset_WD, Preset_OutPut, WD_Enable:Byte) then
```

```
    Memo1.Lines.Add('Failed To Send.');
```

```
    DelayWatchDog := Delay_WD;  
    DelayAfterWatchDog := Delay_AfterWD;  
    OutChangeAfterWatchDog := OutChange_AfterWD;  
    PresetWatchDog := Preset_WD;  
    PresetOutPut := Preset_OutPut;  
    WatchDogEnable := WD_Enable;
```

```
////////////////////////////////////
```

// Change_USB_ID //

```
if not MSUSBDevice1.Change_USB_ID(USB_ID:string[4])then
```

```
    Memo1.Lines.Add('Failed To Send.');
```

```
////////////////////////////////////
```