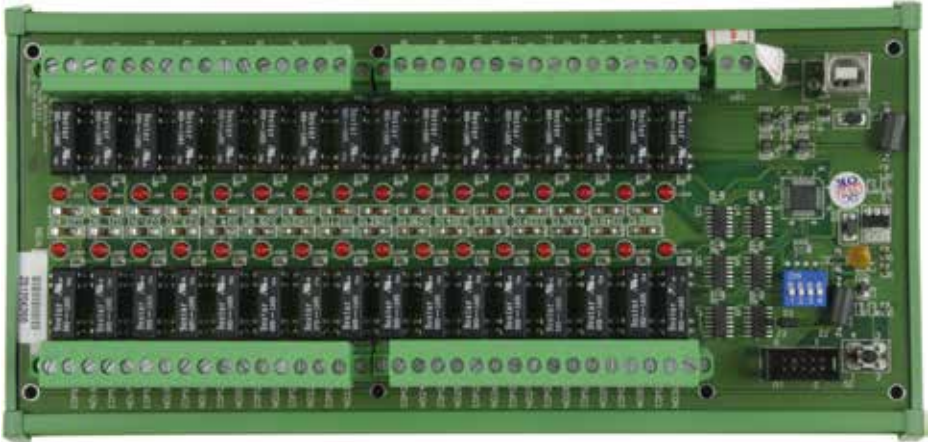


# USB-I/O Manual



**UHR-32 - DIN-Rail-Version**

**UPR-32 - Board-Version**

32 relay output channels

## Data

**Product Code:**

AUSB32R USB 32 RELAY BOARD  
UHR-32 - DIN-Rail-version  
UPR-32 - Board-version

**Bus:** USB 2.0

**Description:**

32 relay output channels

32 Relays - 1 x COM/NO

Max switching current: 1000mA  
Max contact rating for relay: 30V DC

32 LED correspond to I/O ports activation status  
Connections via Pluggable Screw Terminals

**Features:**

For direct DIN-Rail mounting  
Also as board without DIN Rail adapter available

High Speed 8051  $\mu$ C Core  
USB 2.0 Function Controller  
Support USB ID 0~14 SET

POWER External DC+5V (max 5.2V) 1,5A

**Software/Driver:**

Windows-Vista 7/8 will use HID-interface and sample for programming, Linux driver and sample for programming.

**Package includes the following items:**

USB 32R Board  
USB cable  
Software and Manual CD  
The DIN-Rail-version comes with a EMI-Protection-kit  
This kit is optional in the board-version!

Operating temperature range: 0 ~ 55C.  
Relative humidity rage: 0 ~ 90%.  
Size: 250 mm x 120 mm x 55 mm

## Security Note

This device should not be used in applications where failure may result in death or injury without proper consideration and design of associated system architecture and redundant safety features. Connection and repairs are allowed only by a specialist.

When used in a machine or plant, is to ensure that after installation continues to the relevant provisions, rules and guidelines are complied with!

These products come into contact voltage, therefore to consider the applicable VDE regulations VDE 0550 / 0551, VDE 0700, VDE 0711, especially VDE 0100 and VDE 0860.

## J1 USB Connection USB-B

A suitable cable is included

VCC	+5 VDC (USB VBUS POWER)
D-	Data -
D+	Data +
SGND	Signal Ground



**B**

USB wiring is very sensitive against EMI errors (mainly sparking when opening contacts). The U-EMI-1 Kit is included with the DIN rail version and includes two Würth folded cores for the USB data cable and a Ferrite sleeve for the power cable. If you are using a HUB, it should be protected the connection HUB/computer (U-EMI-2)! The kit includes two Würth folded cores for the USB data cable.

The cores must be mounted as shown in the illustrations, as close as possible on the connectors.

But also the avoidance of errors is very important. Therefore, the careful layout and installation of the wiring is very important!



Folding core on the USB cable to the computer



Folding core on the USB cable to the USB IO  
1 or 2 x through the core

## Power Supply - TB5



TB5 - External 5V DC		
1	EXT+V	5V+
2	SGND	5V-

**MAX 5,2V - higher voltage can  
kill the CPU!**

Ferrite sleeve on the power supply cable

The power-supply of our products must be 5V external DC. It is to pay attention to correct polarity. Otherwise, the product could be damaged. If the board is by wrong power supply except function, you can try new store the firmware.

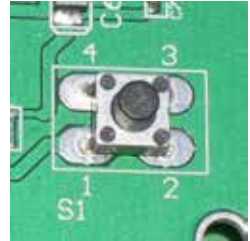
The U-EMI-1 Kit is included with the DIN rail version, and includes ferrite sleeve, shielded by EMI for the power cable. Are shown on the image above.

For earlier versions of the Decision-USB IO was also the possibility of the power supply via the USB bus. To get greater stability, this connection was removed. The USB bus power is not always able to provide enough power for the Relais switched on! The result is a loss of connection or "hanging" USB module. An external power supply provides a secure power supply!

USB Power Management in Windows - In Window System, USB communication might disconnect under Power Saving Mode or Sleep Mode. When connecting USB boards on PC, please make sure windows power management set in case interference with USB communication.

# S1 Reset Button

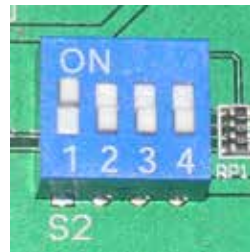
To reset the "hanging" USB-Module



# S2 USB ID

Set different ID for each board

1	2	3	4	Card ID
ON	ON	ON	ON	--
OFF	ON	ON	ON	14
ON	OFF	ON	ON	13
OFF	OFF	ON	ON	12
ON	ON	OFF	ON	11
OFF	ON	OFF	ON	10
ON	OFF	OFF	ON	9
OFF	OFF	OFF	ON	8
ON	ON	ON	OFF	7
OFF	ON	ON	OFF	6
ON	OFF	ON	OFF	5
OFF	OFF	ON	OFF	4
ON	ON	OFF	OFF	3
OFF	ON	OFF	OFF	2
ON	OFF	OFF	OFF	1
OFF	OFF	OFF	OFF	0



# Multiple Boards Connect

When you need to connect more than 3 boards on one PC, please make sure the following below

1. Set different ID for each board.
2. Supply external 5V to each USB board.
3. Supply external 5V to USB hub.

Please make sure your external 5V power supply enough for the USB I/O boards. If input voltage is below 4.8V for USB I/O board, it can't work normally and sometimes it will cause device manager keeping refreshing itself or can't recognize the device.

**TB2/3****Relay-Output****TB4/5**

Pin	Signal	Description
1	COM 0	Relay Ch. 00 - Output
2	NO 0	Relay Ch. 00 - Output
3	COM 1	Relay Ch. 01 - Output
4	NO 1	Relay Ch. 01 - Output
5	COM 2	Relay Ch. 02 - Output
6	NO 2	Relay Ch. 02 - Output
7	COM 3	Relay Ch. 03 - Output
8	NO 3	Relay Ch. 03 - Output
9	COM 4	Relay Ch. 04 - Output
10	NO 4	Relay Ch. 04 - Output
11	COM 5	Relay Ch. 05 - Output
12	NO 5	Relay Ch. 05 - Output
13	COM 6	Relay Ch. 06 - Output
14	NO 6	Relay Ch. 06 - Output
15	COM 7	Relay Ch. 07 - Output
16	NO 7	Relay Ch. 07 - Output

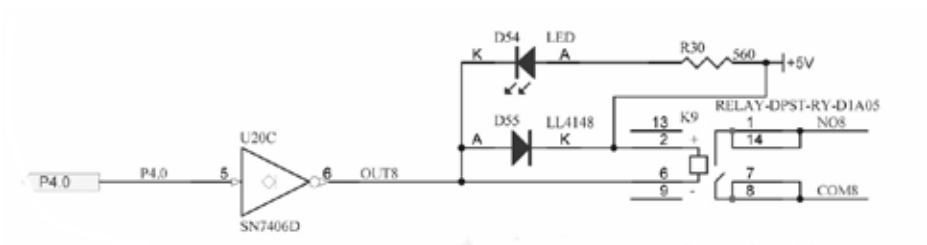
Pin	Signal	Description
1	COM 16	Relay Ch. 16 - Output
2	NO16	Relay Ch. 16 - Output
3	COM 17	Relay Ch. 17 - Output
4	NO 17	Relay Ch. 17 - Output
5	COM 18	Relay Ch. 18 - Output
6	NO 18	Relay Ch. 18 - Output
7	COM 19	Relay Ch. 19 - Output
8	NO 19	Relay Ch. 19 - Output
9	COM 20	Relay Ch. 20 - Output
10	NO 20	Relay Ch. 20 - Output
11	COM 21	Relay Ch. 21 - Output
12	NO 21	Relay Ch. 21 - Output
13	COM 22	Relay Ch. 22 - Output
14	NO 22	Relay Ch. 22 - Output
15	COM 23	Relay Ch. 23 - Output
16	NO 23	Relay Ch. 23 - Output

Pin	Signal	Description
1	COM 8	Relay Ch. 08 - Output
2	NO 8	Relay Ch. 08 - Output
3	COM 9	Relay Ch. 09 - Output
4	NO 9	Relay Ch. 09 - Output
5	COM 10	Relay Ch. 10 - Output
6	NO 10	Relay Ch. 10 - Output
7	COM 11	Relay Ch. 11 - Output
8	NO 11	Relay Ch. 11 - Output
9	COM 12	Relay Ch. 12- Output
10	NO 12	Relay Ch. 12 - Output
11	COM 13	Relay Ch. 13 - Output
12	NO 13	Relay Ch. 13 - Output
13	COM 14	Relay Ch. 14 - Output
14	NO 14	Relay Ch. 14 - Output
15	COM 15	Relay Ch. 15 - Output
16	NO 15	Relay Ch. 15 - Output

Pin	Signal	Description
1	COM 24	Relay Ch. 24 - Output
2	NO 24	Relay Ch. 24 - Output
3	COM 25	Relay Ch. 25 - Output
4	NO 25	Relay Ch. 25 - Output
5	COM 26	Relay Ch. 26 - Output
6	NO 26	Relay Ch. 26 - Output
7	COM 27	Relay Ch. 27 - Output
8	NO 27	Relay Ch. 27 - Output
9	COM 28	Relay Ch. 28- Output
10	NO 28	Relay Ch. 28 - Output
11	COM 29	Relay Ch. 29 - Output
12	NO 29	Relay Ch. 29 - Output
13	COM 30	Relay Ch. 30 - Output
14	NO 30	Relay Ch. 30 - Output
15	COM 31	Relay Ch. 31 - Output
16	NO 31	Relay Ch. 31 - Output

# Relay-Out

## Schematic



## Contact protection circuits

### Contact protection circuits

When a reed relay is used with an electromagnetic relay or solenoid, the energy stored will cause an inverse voltage when the reed contacts break. The voltage, although dependent on the inductance value, sometimes reaches as high as several hundred volts and becomes a major factor to deteriorate the contacts.

- DC: Protection circuit with a diode
- DC and AC: Protection circuit with Varistor or RC-element

The suppressor must be made to the load.

## LED

1. The LED1 is an indicator to show the power is supplied normally.
2. The LED2 is an indicator to warning the USB link status. When it lights, it means USB connection works normally, otherwise it is fail.



## Installation

The decision-computer USB devices use the HID (human interface device). The HID belongs to the generic device class is integrated in the operating system. If a new HID device is connected, no driver installation is required. The functions for access and control of HID hid.dll you can find in the Windows System32 folder.

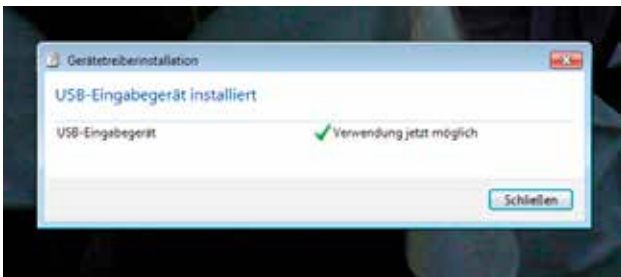
## Windows 7/8 installation example



1. Power supply 5V connect

2. USB connect

3. USB input device - device driver software is successfully installed



4. USB input device - use now possible



5. In the Control Panel, you can find the Decision-USB module now

6. Ready to use



# SOFTWARE-PROGRAMMING UNDER WINDOWS AND LINUX

On Windows, we offer a function library and dll file as programming help. See the manual „USBII\_Manual.pdf“ and demo code in VB/VC / Delphi on the decision-Studio CD.

We offer a C-source Linux users for direct access to the USB devices. See „Dcihid 0.5.1.tgz“ manual and example.

## DIAGNOSTICS UNDER WINDOWS/XP

USB test Program.exe is a diagnostic tool to test USB devices on Windows/XP.  
The USB test software can be found on the decision-Studio CD.

The examples and drivers be developed continuously. See the latest on the decision-computer-Merz „Service CD“.

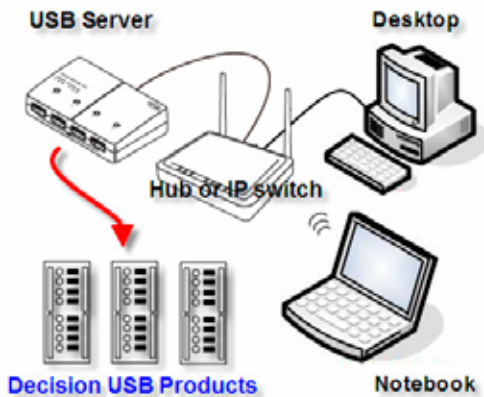
An important way to get more informations you find at <http://www.usb-industrial.com>

Software support on the short way: <http://www.usb-industrial.com/support.html>

### USB Industrial.com Overview:

<b>Windows Support</b>	2010/04 USBII.dll 2.0.0.4	This package includes Dynamic-link library which is developed by Decision Computer to communicate with the USB Series Device. It can be included in multiple computer language (VB6, VC6, VB.NET, C# Delphi) under Windows.
<b>Watchdog Timer</b>		This watchdog timer is a kind of software timer that triggers a system reset or other corrective action if the main program, due to some fault condition. The intention is to bring the system back from the unresponsive state into normal operation. This function is new released and please contact us to get further information.
<b>VCP driver</b>	( For LABKIT Only )	Virtual COM port (VCP) drivers cause the USB device to appear as an additional COM port available to the PC. Application software can access the USB device in the same way as it would access a standard COM port. This function is only implemented in USBLABKIT
<b>Linux Support</b>	dcihid - 0.5.1 Basic function library and demo program 2009.05.01	This package includes a c library and a demo program which is developed by Decision Computer to communicate with the USB Series Device under Linux. It also includes a ReadMe file to demonstrate how to use it and package's format is .tgz.
<b>Firmware Update</b>	Firmware Hex file Download	This Package includes a driver and a software which is developed by Decision Computer to update the newest firmware into the USB Series Device. When new version of firmware is released, user can follow the instructions to update the firmware.
<b>LabVIEW Support</b>	LabVIEW 8 LabVIEW 2009	This package includes manual and examples which demonstrate how to connect and develop USB Series Device under LabVIEW, which is a well-known platform and development environment for a visual programming language from NI.
<b>ProfilAB Support</b>		This package includes manual and examples which demonstrate how to connect and develop USB Series Device under ProfilAB, which is a well-known platform and development environment for a visual programming language from Abacom.
<b>Init Value Setting Tool</b>	(For Output Channel)	The Init Value Setting Tool is a software tool to set init value for output channel. User can use this tool to plan output channel as default high or default low when power on.
<b>Data Acquisition and Remote Monitoring Tool</b>		The Data Acquisition and Remote Monitoring Tool (DARMT) is a software tool to record high/low state reports at local computer, and transmit them to FTP site to achieve data acquisition and remote monitoring

## USB by LAN or Wireless



The remote control of Decision USB products by LAN or wireless with a remote-PC is very simple with a multi port USB Server

Because no driver should be installed to the installation and programming is very easy.

Under Windows, are the external USB I/O directly in the Device Manager and can be connect or control such as in the original host PC.

## Firmware Update Manual

USBBootloader.exe is the tool software to update firmware into the USB SerialDevice Board developed by Decision Computer. When you get a new version of firmware (.hex), you can follow the steps to update firmware to the board.

1. Remove the external input signal Voltage and only support device power.
2. Set Board Id 15 (All on) for Update Mode and press the Reset button.
3. Connect PC to the Board by USB
4. If this is the first to use this function, please indicate the driver install path to the Driver Folder to install the driver.
5. Open the Software USBBootloader.exe and press the Open button and indicate the hex file and then press the Download button to update firmware.
6. Set Board Id between 0 ~ 14 and press Reset button and connect PC again.

## Communication J 2/3 - only option!

Connector and Jumper for Serial Communication (J3) To use RS422/RS485/RS232, please connect J2 to extension board by 10 pins flat cable. (Optional)

Enable Serial Port (J3)

J3 is used enable serial port communication, when short the J3, means enable serial port, otherwise, when open the J3, the serial port communication is disable.

## APPENDIX B DATA SHEET

### ■ FEATURES

- 1 Form A Contact
- DIP Terminal
- Application for Fax Modem, Telecommunication, Security Alarm System
- UL File No. E147052



### ■ COIL RATING (at 20 °C)

Nominal Voltage (VDC)	Coil Resistance ( $\Omega \pm 10\%$ )	Nominal Current (mA)	Pick-Up Voltage (VDC)	Drop-Out Voltage (VDC)	Maximum Allowable Voltage (VDC)	Power Consumption (mW)
5	500	10.0	3.75	0.6	15	50
12	1000	12.0	9.0	1.44	30	144
24	2150	11.2	18.0	2.88	44	268

### ■ ORDERING INFORMATION

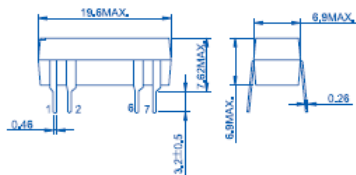
BRD-1A05-D

Coil Voltage	Option
See Coil Rating	Nil: Standard D: With Diode

## SPECIFICATIONS

<b>Contact Arrangement</b>		1 Form A
<b>Contact Material</b>		Ru / Rh
<b>Contact Resistance</b>		Max. 150mΩ (initial)
<b>Contact Rating (at Resistive Load)</b>	<b>Max. Switching Voltage</b>	100VDC
	<b>Max. Switching Current</b>	0.5A
	<b>Max. Switching Power</b>	10W (DC) / 10VA (AC)
	<b>Max. Carrying Current</b>	1A
<b>Dielectric Strength</b>		
<b>Between Coil &amp; Contact</b>		1400 VDC (1 minute )
<b>Between Contacts</b>		250VDC (1 minute )
<b>Operate Time</b>		Max. 1.0m Sec.
<b>Release Time</b>		Max. 0.5m Sec.
<b>Ambient Temperature</b>		-40 °C→+85 °C
<b>Insulation Resistance</b>		Min. 100MΩ at 500VDC
<b>Vibration Resistance</b>		1.5mm D.A. 10-55HZ
<b>Shock</b>	<b>Functional</b>	20G
	<b>Destruction</b>	100G
<b>Mechanical Life</b>		1 x 10 <sup>5</sup> operations (at no load)
<b>Electrical Life</b>		1 x 10 <sup>6</sup> operations ( at rated load )
<b>Weight</b>		Approx. 2g

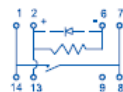
### DIMENSIONS(mm)



General Tolerance ±0,3

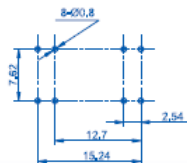
### WIRING DIAGRAM

(Bottom View)



### PC board pattern (mm)

(Bottom View)



General Tolerance ±0,1

# CERTIFICATE




## VERIFICATION OF COMPLIANCE

APPLICANT            DESICION GROUP INC.

ADDRESS             4<sup>th</sup> Floor, No. 31, Alley 4, Lane 36, Sec. 5, Ming-Shen  
East Road, Taipei Postal code: 10576, Taiwan, R.O.C.

EQUIPMENT          USB Automation I/O board

MODEL NAME        AUSB series

TRADE NAME        

REPORT NO.         WSCE1608014

STANDARD(S)      EMI --- EN 55032 CLASS B: 2012  
                              EN 61000-3-2: 2014  
                              EN 61000-3-3: 2013  
                              EMS --- EN 55024: 2010  
                              IEC 61000-4-2 : 2008  
                              IEC 61000-4-3 : 2006+A1: 2007+A2:2010  
                              IEC 61000-4-4 : 2012  
                              IEC 61000-4-5 : 2014  
                              IEC 61000-4-6 : 2013  
                              IEC 61000-4-8 : 2010  
                              IEC 61000-4-11 : 2004

The above equipment was tested by WEISHANG Certification Co., Ltd. for compliance with the requirements set forth in the EUROPEAN COUNCIL Directive 2014/30/EU and the technical standards mentioned above. The results of testing in this report apply only to the product/system, which was tested. Other similar equipment will not necessarily produce the same results due to production tolerance.

Approved By: \_\_\_\_\_

  
Brian Yu / Manager

Issued Date: SEP. 06, 2016



WEISHANG Certification Corp.  
12F.-5, No.27-1, Ln. 168, Kangning St., Xizhi Dist., New Taipei City 221, Taiwan (R.O.C.)

# DECLARATION OF CONFORMITY

For the following equipment :

Equipment : USB Automation I/O board

Model Name: AUSB series

Applicant: DESICION GROUP INC.

Address: 4<sup>th</sup> Floor, No. 31, Alley 4, Lane 36, Sec. 5, Ming-Shen East Road,  
Taipei Postal code: 10576, Taiwan, R.O.C.

Is herewith confirmed to comply with the requirements set out in the Council Directive on the Approximation of the Laws of the Member States relating to Electromagnetic Compatibility (2014/30/EU). For the evaluation regarding the electromagnetic compatibility, the following standards were applied :

EN 55032 CLASS B: 2012

EN 61000-3-2: 2014

EN 61000-3-3: 2013

EN55024: 2010

IEC 61000-4-2: 2008

IEC 61000-4-3: 2006+A1: 2007+A2:2010

IEC 61000-4-4: 2012

IEC 61000-4-5: 2014

IEC 61000-4-6: 2013

IEC 61000-4-8: 2010

IEC 61000-4-11: 2004

The following manufacturer/importer is responsible for this declaration :

Person responsible for marking this declaration :

*Casper Kan Chang*

201609 6

(Place)

(Date)

(Signature)



## **A.1 Copyright**

Copyright DECISION COMPUTER INTERNATIONAL CO., LTD. All rights reserved. No part of SmartLab software and manual may be produced, transmitted, transcribed, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of DECISION COMPUTER INTERNATIONAL CO., LTD.

Each piece of SmartLab package permits user to use SmartLab only on a single computer, a registered user may use the program on a different computer, but may not use the program on more than one computer at the same time.

Corporate licensing agreements allow duplication and distribution of specific number of copies within the licensed institution. Duplication of multiple copies is not allowed except through execution of a licensing agreement. Welcome call for details.

## **A.2 Warranty Information**

SmartLab warrants that for a period of one year from the date of purchase (unless otherwise specified in the warranty card) that the goods supplied will perform according to the specifications defined in the user manual. Furthermore that the SmartLab product will be supplied free from defects in materials and workmanship and be fully functional under normal usage.

In the event of the failure of a SmartLab product within the specified warranty period, SmartLab will, at its option, replace or repair the item at no additional charge. This limited warranty does not cover damage resulting from incorrect use, electrical interference, accident, or modification of the product.

All goods returned for warranty repair must have the serial number intact. Goods without serial numbers attached will not be covered by the warranty.

The purchaser must pay transportation costs for goods returned. Repaired goods will be dispatched at the expense of SmartLab.

To ensure that your SmartLab product is covered by the warranty provisions, it is necessary that you return the Warranty card.

Under this Limited Warranty, SmartLab's obligations will be limited to repair or replacement only, of goods found to be defective a specified above during the warranty period. SmartLab is not liable to the purchaser for any damages or losses of any kind, through the use of, or inability to use, the SmartLab product.

SmartLab reserves the right to determine what constitutes warranty repair or replacement.

Return Authorization: It is necessary that any returned goods are clearly marked with an RA number that has been issued by SmartLab. Goods returned without this authorization will not be attended to.

**USB  
Dynamic Industrial Interface  
V 2.0.1.9**

**A Universal  
Application Programming Interface  
To Data Acquisition Products**

**Users Manual**

Design & Implementation by  
Decision Computer International Company

No parts of this documentation may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording, or otherwise) without the prior written permission of Decision Computer International Company.

2010/04/20



# Contents

1.	Introduction	3
2.	Features	4
3.	Device Type definition	5
4.	Data Types of Function calls	6
5.	Functions to open and close Devices	7
6.	Functions for digital input/output	10
7.	Functions for reset hardware device	16
8.	Functions for analog input/output	17
9.	Functions for watch dog	18
10.	Using USB DII with different programming language	20
	10.1. C++.	20
	10.2 Visual Basic	20
11.	Technical support and Feedback	20

# 1. Introduction

This document provides the USB Dynamic Industrial Interface Specifications, including all function calls, and operating procedures.

## **Disclaimer:**

Decision Computer International Company (DECISION) cannot take responsibility for consequential damages caused by using this software. In no event shall DECISION be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use this product, even if we have been advised of the possibility of such damages.

## **Trademark Acknowledgments:**

Windows 98, Windows ME, Windows 2000, Windows XP, Windows 7, Visual Basic, Visual C++ are registered trademarks of Microsoft Corporation.

# 2. Features

The USB Dynamic Industrial Interface (USBDI) was created to provide a standard way to access the functionality provided by all USB data acquisition products. Specifically, the USBDI provides the following features:

## **Platform-independent**

The library is compatible under Windows 98, Windows ME, Windows 2000, windows XP, Vista, and Win7. The compatibility under these operation systems guarantees that programs written for either operating system will work unchanged on the other, even without recompilation.

## **Abstracts Card Functionality from Card Design**

The interface concentrates on a card's functionality and hides the user from having to know specifics about the card design, for example, which port needs to be accessed in order to access specific functionality. All details of the card implementation are hidden from the user.

## **Multiple Device Support**

You could access device by its name or by its information (device type, id index).

## **Programming Language Independent**

The library provides a language independent way to access the USB industrial I/O cards, by using a Dynamic-Link-Library architecture.

### 3. Device Type Definition

Below are names for device types and its' corresponding defined value:

USB_16PIO	0x01	// USB 16 Channel Photo Input / 16 Channel Photo Output Board
USB_LABKIT	0x02	// USB LABKIT
USB_16PR	0x03	// USB 16 Channel Photo Input / 16 Channel Relay Output Board
USB_STARTER	0x04	// USB STARTER
USB_8PR	0x06	// USB 8 Channel Photo Input / 8 Channel Relay Output Board
USB_4PR	0x07	// USB 4 Channel Photo Input / 4 Channel Relay Output Board
USB_8PI	0x08	// USB 8 Channel Photo Input Board
USB_8RO	0x09	// USB 8 Channel Relay Output Board
USB_16PI	0x0A	// USB 16 Channel Photo Input Board
USB_16RO	0x0B	// USB 16 Channel Relay Output Board
USB_32PI	0x0C	// USB 32 Channel Photo Input Board
USB_32RO	0x0D	// USB 32 Channel Relay Output Board
USB_IND	0x0E	// USB Industry Board
USB_M_4IO	0x10	// USB Mini 4 I/O

Notice : Please use this function to open USB\_14ADDA or USB\_16ADDA.

### 4. Data Types of Function calls

Since the USBDI was developed in the C++ language, some data types used may not be present in the programming language you want to use. Please find the following data type conversion table for your convenience:

HANDLE	An opaque 32-bit integer
BYTE	A 8-bit unsigned integer
BOOL	A 32-bit integer, either 0 (FALSE) or 1 (TRUE)
DWORD	A 32-bit unsigned integer
HWND	A 32-bit integer representing a valid handle to a Window
LPTSTR	A 32-bit flat pointer to a zero terminated string
LPBOOL	A 32-bit flat pointer to a variable of type BOOL
LPBYTE	A 32-bit flat pointer to a variable of type BYTE
LPDWORD	A 32-bit flat pointer to a variable of type DWORD

Also note that the DLL employs the Standard Call (Pascal) calling mechanism, which is used for all system. USBDI as well and is compatible with VB, VC, Delphi, .NET, and notice the variable with same type name may have different define in different program language. For example, in Visual Basic 6, the width of Integer is 16 bits and the width of Long is 32 bits, but in Visual Basic. Net, the width of Integer becomes 32 bits and the width of Long becomes 64 bits. If you declare variable with different width from our define, it may cause some run-time error.

## 5. Functions to open and close Devices

### hid\_OpenDevice

This function opens a device for further access by USB. Please do not use this function to open USB\_14ADDA or USB\_16ADDA.

#### Declaration

```
HANDLE hid_OpenDevice ( DWORD device_type,  
                        DWORD device_id );
```

#### Parameters

device\_type      The type of the device to open.  
device\_id        Device's id on the Board.

For more information, please see "Device Type Table & ID Table" following below.

#### Return value

A valid handle representing the device, or INVALID\_HANDLE\_VALUE (-1) if an error occurred. For USB\_STARTER, there is no ID selection and device\_id = 0

#### Example

```
HANDLE hDevice = hid_OpenDevice(Device Type, Device Index); if (hDevice == INVALID_  
HANDLE_VALUE)  
{  
  MessageBox (NULL, "Open Failed!", "Error", MB_OK);  
}
```

---

### hid\_CloseDevice

This function closes a device by USB.

#### Declaration

```
BOOL    hid_CloseDevice (HANDLE hDevice)
```

#### Parameters

hDevice    A valid device handle.

#### Return value

TRUE if successful, FALSE otherwise.

#### Example

```
hid_CloseDevice(hDevice);
```

## **com\_OpenDevice**

This function opens a device for further access by Serial Port. Please use this function to open USB\_14ADDA or USB\_16ADDA.

### **Declaration**

```
HANDLE com_OpenDevice ( DWORD device_type,  
                        DWORD device_id,  
                        DWORD port_num );
```

### **Parameters**

device_type	The type of the device to open.
device_id	Device's id on the board. For more information, please see "Device Type Table & ID Table" following below.
port_num	Com Port Num to open.

### **Return value**

A valid handle representing the device, or INVALID\_HANDLE\_VALUE (-1) if an error occurred.

### **Example**

```
HANDLE hDevice = com_OpenDevice(Device Type, Device Index, 1); if (hDevice == INVALID_  
HANDLE_VALUE)  
    MessageBox (NULL, "Open Failed!", "Error", MB_OK);
```

## **com\_CloseDevice**

This function closes a device by Serial Port.

### **Declaration**

```
BOOL com_CloseDevice(HANDLE hDevice)
```

### **Parameters**

hDevice A valid device handle.

### **Return value**

TRUE if successful, FALSE otherwise.

### **Example**

```
com_CloseDevice(hDevice);
```

### **Remarks**

Please see "Serial\_Communication.pdf" to set hardware for serial communication, and USB\_LAB-KIT, USB\_STARTER, USB\_8PR are not supported by serial communication.

## **Device Type Table**

<b>Product</b>	<b>device_type</b>
USB_16PIO	0x01
USB_LABKIT	0x02
USB_16PR	0x03
USB_STARTER	0x04
USB_8PR	0x06
USB_4PR	0x07
USB_8PI	0x08
USB_8RO	0x09
USB_16PI	0x0A
USB_16RO	0x0B
USB_32PI	0x0C
USB_32RO	0x0D
USB_IND	0x0E
USB_M_4IO	0x10

## Device ID Table

( Switch Setting on the Device Board )



Switch Setting	device_id
1, 2, 3, 4 OFF	0
2, 3, 4 OFF, 1 ON	1
1, 3, 4 OFF, 2 ON	2
3, 4 OFF, 1, 2 ON	3
1, 2, 4 OFF, 3 ON	4
2, 4 OFF, 1, 3 ON	5
1, 4 OFF, 2, 3 ON	6
4 OFF, 2, 3, 4 ON	7
1, 2, 3 OFF, 4 ON	8
2, 3 OFF, 1, 4 ON	9
1, 3 OFF, 2, 4 ON	10
3 OFF, 1, 2, 4 ON	11
1, 2 OFF, 3, 4 ON	12
2 OFF, 1, 3, 4 ON	13
1 OFF, 2, 3, 4 ON	14
1, 2, 3, 4 ON	Firmware update

## 6. Functions for digital input/output

### hid\_SetDigitalByte

This function sets or clears a byte on a digital output line by USB.

#### Declaration

```
BOOL hid_SetDigitalByte ( HANDLE hDevice,  
                          DWORD dwPort,  
                          BYTE byPortState  
                          );
```

#### Parameters

hDevice	A valid device handle, previously obtained from hid_OpenDeviceDevice
dwPort	The index of the port on the card to manipulate. The first port has index 0. For more information, please see "Write Address Table" following below.
byPortState	The new state of the port

#### Return value

TRUE if successful, FALSE otherwise.

If an error occurred, GetLastError() may return the following values:

ERROR\_INVALID\_PARAMETER - The handle passed was invalid, or the port number was out of range for the device selected.

#### Example

```
HANDLE hDevice = hid_OpenDevice(0x01,0);  
if (hDevice != INVALID_HANDLE_VALUE)  
{  
    hid_SetDigitalByte( hDevice, 0, 0xFF); // set's all bits on the first port  
    hid_CloseDevice(hDevice);  
}
```



## **com\_SetDigitalByte**

This function sets or clears a byte on a digital output line by Serial Port.

### **Declaration**

```
BOOL com_SetDigitalByte ( HANDLE hDevice,  
                        DWORD dwPort,  
                        BYTE byPortState  
                        );
```

### **Parameters**

hDevice	A valid device handle, previously obtained from com_OpenDevice
dwPort	The index of the port on the card to manipulate. The first port has index 0. For more information, please see "Write Address Table" following below.
byPortState	The new state of the port

### **Return value**

TRUE if successful, FALSE otherwise.

If an error occurred, GetLastError() may return the following values:

ERROR\_INVALID\_PARAMETER - The handle passed was invalid, or the port number was out of range for the device selected.

### **Example**

```
HANDLE hDevice = com_OpenDevice(0x01,0);  
if (hDevice != INVALID_HANDLE_VALUE)  
{  
    com_SetDigitalByte( hDevice, 0, 0xFF); // set's all bits on the first port  
    com_CloseDevice(hDevice);  
}
```

### **Remarks**

Please see "Serial\_Communication.pdf" to set hardware for serial communication, and USB\_LAB-KIT, USB\_STARTER, USB\_8PR are not supported by serial communication.

## Write Address Table

Product	dwPort	Content
USB_16PIO	0x02	OUT07 to OUT00
	0x03	OUT15 to OUT08
USB_LABKIT	0x03	P1D07 to P1D00
	0x03	P1D07 to P1D00
USB_16PR	0x02	OUT07 to OUT00
	0x03	OUT15 to OUT08
USB_8PR	0x01	OUT07 to OUT00
	0x02	DIO7 to DIO0
	0x03	DIO15 to DIO8
USB_4PR	0x02	OUT03 to OUT00
USB_8RO	0x02	OUT07 to OUT00
USB_16RO	0x02	OUT07 to OUT00
	0x03	OUT15 to OUT08
USB_32RO	0x00	OUT07 to OUT00
	0x01	OUT15 to OUT08
	0x02	OUT23 to OUT16
	0x03	OUT31 to OUT24
USB_IND	0x00	Port 0
	0x01	Port 1
	0x02	Port 2
	0x03	Port 3
	0x04	Port 4
	0x05	Port 5
	0x06	Port 6
	0x07	Port 7
	0x08	DIO
	0x0D	IOCONFIG
USB_M_4IO	0x02	OUT03 to OUT00

## hid\_GetDigitalByte

This function reads a complete byte from a digital input port of a device by USB.

### Declaration

```
BOOL hid_GetDigitalByte ( HANDLE hDevice,  
                          DWORD dwPort,  
                          LPBYTE lpbyPortState  
                        );
```

### Parameters

hDevice	A valid device handle, previously obtained from hid_OpenDeviceDevice
dwPort	The index of the port on the card to manipulate. The first port has index 0. For more information, please see "Read Address Table" following below.
lpbyPortState	A pointer to a variable of type BYTE receiving the new state of the port

### Return value

TRUE if successful, FALSE otherwise.

If an error occurred, GetLastError() may return the following values:

ERROR\_INVALID\_PARAMETER – The handle passed was invalid, or the port number was out of range for the device selected.

### Example

```
HANDLE hDevice = hid_OpenDevice(0x01,0); if (hDevice != INVALID_HANDLE_VALUE)  
{  
  hid_GetDigitalByte( hDevice, 0, &byState); // reads the state of the first input port hid_  
  CloseDevice(hDevice);  
}
```

## com\_GetDigitalByte

This function reads a complete byte from a digital input port of a device by Serial Port.

### **Declaration**

```
BOOL com_GetDigitalByte ( HANDLE hDevice,  
                        DWORD dwPort,  
                        LPBYTE lpbyPortState  
                        );
```

### **Parameters**

hDevice	A valid device handle, previously obtained from com_OpenDevice
dwPort	The index of the port on the card to manipulate. The first port has index 0. For more information, please see "Read Address Table" following below.
lpbyPortState	A pointer to a variable of type BYTE receiving the new state of the port

### **Return value**

TRUE if successful, FALSE otherwise.

If an error occurred, GetLastError() may return the following values:

ERROR\_INVALID\_PARAMETER – The handle passed was invalid, or the port number was out of range for the device selected.

### **Example**

```
HANDLE hDevice = com_OpenDevice(0x01,0);  
if (hDevice != INVALID_HANDLE_VALUE)  
{  
    com_GetDigitalByte( hDevice, 0, &byState); // reads the state of the first input port  
    com_CloseDevice(hDevice);  
}
```

### **Remarks**

Please see "Serial\_Communication.pdf" to set hardware for serial communication, and USB\_LAB-KIT, USB\_STARTER, USB\_8PR are not supported by serial communication.

## Read Address Table

Product	dwPort	Content
USB_16PIO	0x00	IN07 to IN00
	0x01	IN15 to IN08
USB_LABKIT	0x02	P0D07 to P0D00
USB_STARTER	0x02	P0D07 to P0D00
USB_16PR	0x00	IN07 to IN00
	0x01	IN15 to IN08
USB_8PR	0x00	IN07 to IN00
	0x02	DIO7 to DIO0
	0x03	DIO15 to DIO8
	0x10	JP9/JP10 Settings
USB_4PR	0x00	IN03 to IN00
USB_8PI	0x00	IN07 to IN00
USB_16PI	0x00	IN07 to IN00
	0x01	IN15 to IN08
USB_32PI	0x00	IN07 to IN00
	0x01	IN15 to IN08
	0x02	IN23 to IN16
	0x03	IN31 to IN24
USB_IND	0x00	Port 0
	0x01	Port 1
	0x02	Port 2
	0x03	Port 3
	0x04	Port 4
	0x05	Port 5
	0x06	Port 6
	0x07	Port 7
	0x08	DIO
	0x0D	IOCONFIG

	0x10	Port 0 default value
	0x11	Port 1 default value
	0x12	Port 2 default value
	0x13	Port 3 default value
	0x14	Port 4 default value
	0x15	Port 5 default value
	0x16	Port 6 default value
	0x17	Port 7 default value
	0x18	Port DIO default value
	0x19	Input/output default setting
USB_M_4IO	0x00	IN03 to IN00

### **Remarks**

In USB\_8PR, we provide 2 digital ports for user to define either as input or output. It can be defined by Jumper 10 and Jumper 11 on the board. And we can use `hid_GetDigitalByte` / `com_GetDigitalByte` function to read Jumper State to determine which port is either input or output.

`hid_GetDigitalByte( hDevice, 0x10, &byState);` // or use `com_GetDigitalByte` for serial communication

When JP9 is closed, DIO7 - DIO0 is for Input.      The fifth bit of `byState` is 0

When JP9 is opened, DIO7 - DIO0 is for Output.      The fifth bit of `byState` is 1

When JP10 is closed, DIO15 – DIO8 is for Input.      The sixth bit of `byState` is 0

When JP10 is opened, DIO15 – DIO8 is for Output.      The sixth bit of `byState` is 1

## 7. Functions for reset hardware device

### **hid\_ResetHW**

This function directly resets the hardware device by USB. And all channels on the board will load default value. If you need to control the device again, please use hid\_open to get the handle again.

#### **Declaration**

BOOL hid\_ResetHW(HANDLE hDevice)

#### **Parameters**

hDevice A valid device handle.

#### **Return value**

TRUE if successful, FALSE otherwise.

#### **Example**

```
hid_ResetHW (hDevice);
```

---

### **com\_ResetHW**

This function directly resets the hardware device by Serial Port. And all channels on the board will load default value.

#### **Declaration**

BOOL com\_ResetHW(HANDLE hDevice)

#### **Parameters**

hDevice A valid device handle.

#### **Return value**

TRUE if successful, FALSE otherwise.

#### **Example**

```
com_ResetHW(hDevice);
```

## 8. Functions for analog input/output

### hid\_GetAnalogChannel

This function reads a complete word from an analog input port of a device by USB.

#### Declaration

```
BOOL hid_GetAnalogChannel ( HANDLE hDevice,  
                           DWORD dwPort,  
                           LPDWORD lpdwPortState  
                           );
```

#### Parameters

hDevice	A valid device handle, previously obtained from hid_OpenDeviceDevice
Port	The index of the port on the card to manipulate. The first port has index 0.
lpdwPortState	A pointer to a variable of type DWORD receiving the new state of the port

#### Return value

TRUE if successful, FALSE otherwise.

If an error occurred, GetLastError() may return the following values:

ERROR\_INVALID\_PARAMETER - The handle passed was invalid, or the port number was out of range for the device selected.

#### Example

```
HANDLE hDevice = hid_OpenDevice(0x02,0); // USB_LABKIT  
if (hDevice != INVALID_HANDLE_VALUE)  
{  
    hid_GetAnalogChannel ( hDevice, 0, &dwState); // reads the state of the first analog input port  
    hid_CloseDevice (hDevice);  
}
```

#### Remarks

This function now only enable in USB\_LABKIT and USB\_STARTER device. The range of dwPort is from 0~7.

















