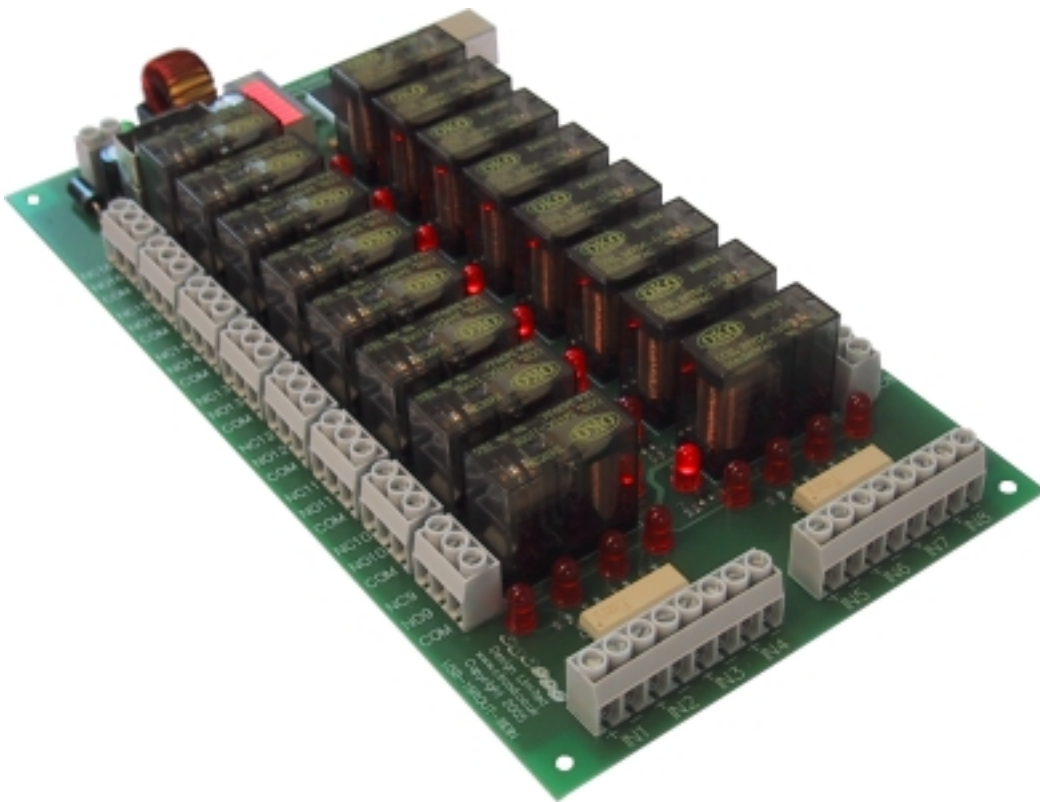


Automation and Control Products

USB-R16OUT-8IDIN

**USB Interface
16 Relay Drives
8 Isolated Digital inputs**



USB 16ROUT8IDIN Hardware	3
Specification	3
Installation guide	4
Windows USB Device Driver	5
Control Software	8
Software Interface	10
cd_Heartbeat	11
cd_SetInterface	12
cd_RelayOn	13
cd_RelayOff	14
cd_Get16Relays	15
cd_Set16Relays	16
cd_Get8Inputs	17
cd_EnableWatchdog	18
Error Codes	19

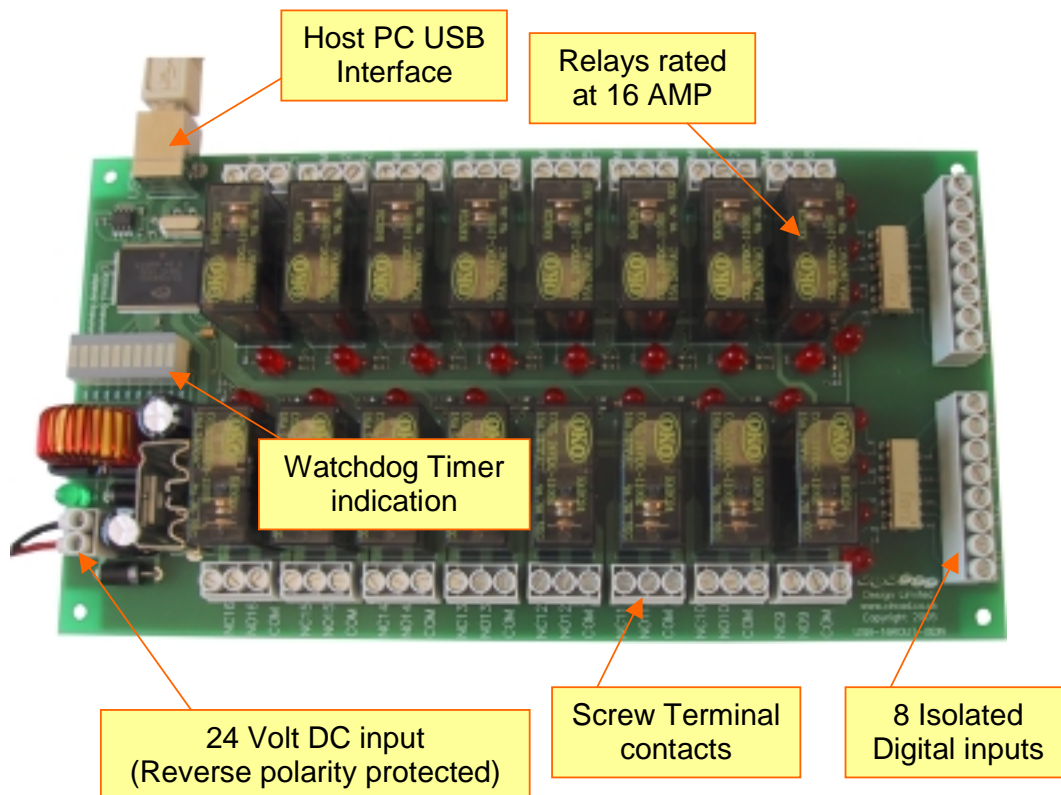
USB 16ROUT8IDIN Hardware

The 16ROUT8IDIN is a general-purpose Computer controlled relay board offering 16 Open Contact Relay outputs rated at 16 AMPS, with 8 (1.5KV) Isolated Digital inputs.

Your PC connects to the 16ROUT8IDIN via the USB interface. Software Device Drivers are supplied with sample programs (including source code) for interfacing to C++, C#, Delphi and Visual Basic.

Specification

- ☆ 16 K4B changeover relays rated at 16A 240V AC, 16A 24V DC.
- ☆ Screw Terminal Connections
- ☆ Power Supply required, DC rated at 24 Volts 750mA
- ☆ 8 Isolated Digital Inputs, trigger voltage 12 ~ 24 Volts.
- ☆ Optional Watchdog Timer will automatically shut off relays.
- ☆ Easy to use software interface.

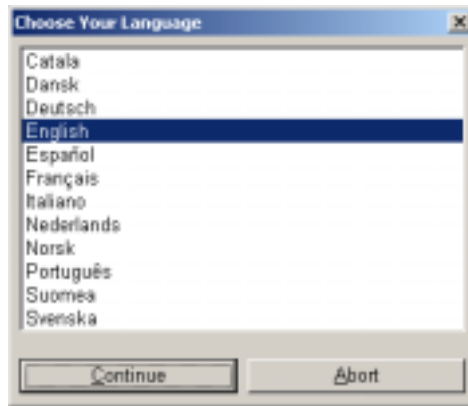


Installation guide

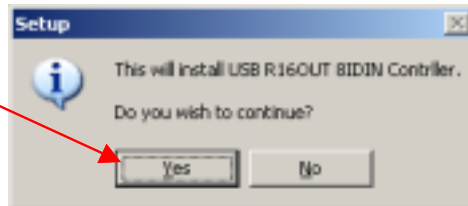
Before connecting the Controller to your PC, please install the software and ensure you have the 16ROUT8IDIN CD in the drive.

The Installation application will start automatically, if it does not please run **setup.exe** from the CD.

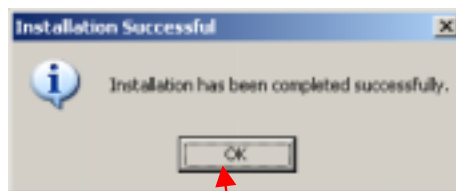
First select language. Selecting a language will only affect messages from the installation program, not the controller application.



Next you will be prompted to install USB R16OUT 8DIN Controller, answer **Yes** to continue.



The Software will then install to your hard disk.

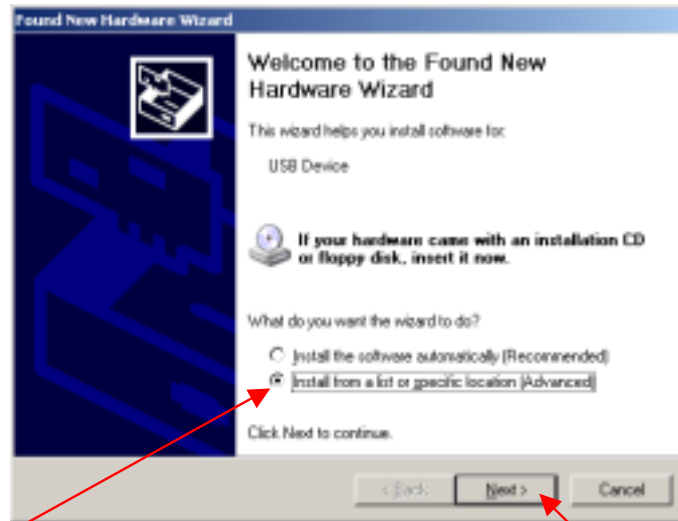


After successful installation press **OK** to continue.

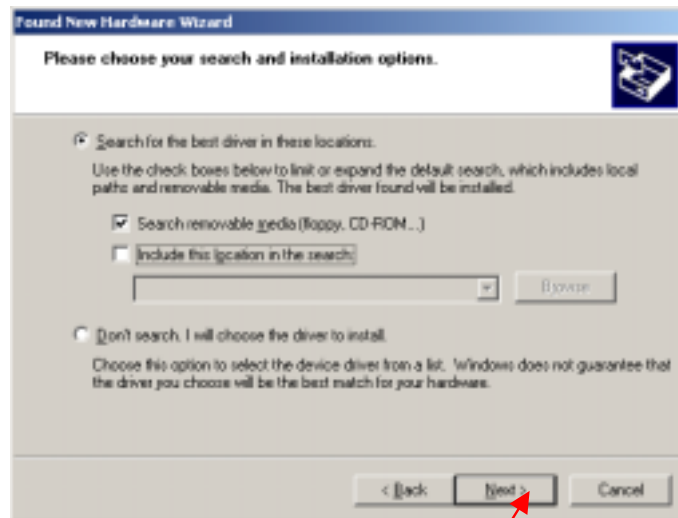
Windows USB Device Driver

Connect the USB-R16OUT-8IDIN controller to your Host PC using a USB lead, then power on the controller.

The Host computer will “PING” to indicate it has detected a new USB device; Windows will automatically open the New Hardware Wizard.

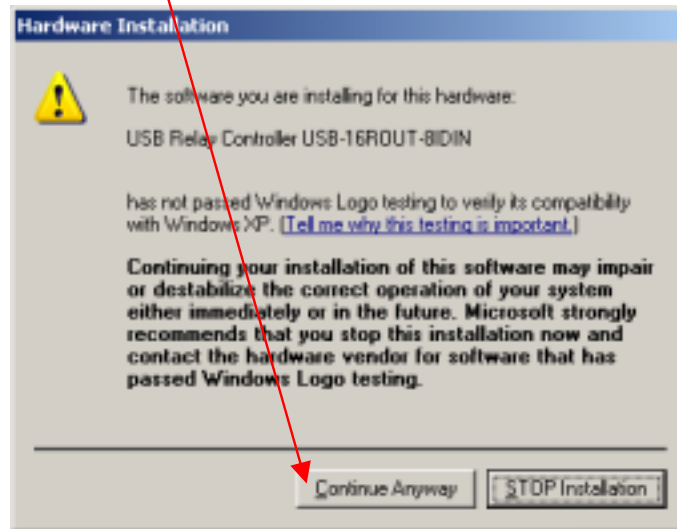


Select, install from a specific location, and then press **Next**.

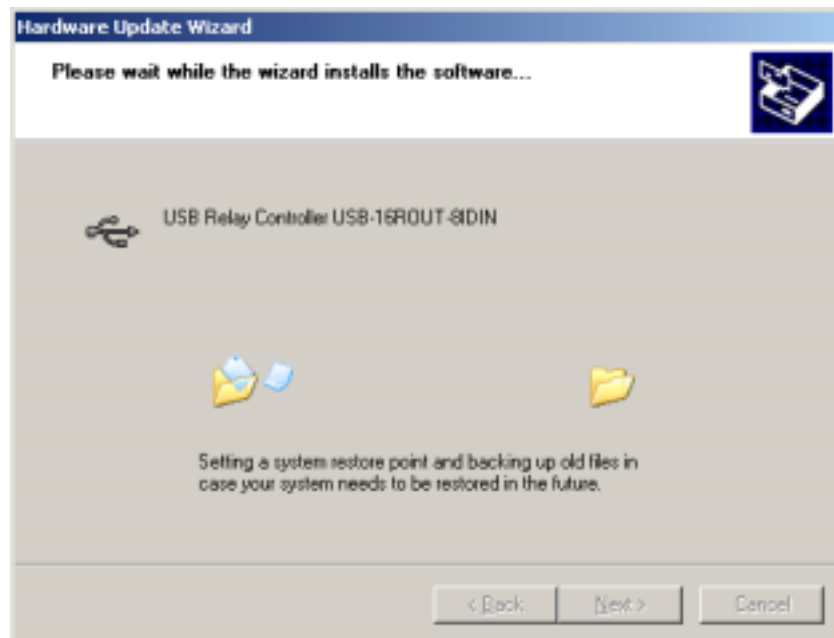


Ensure that the EzBIOS Install CD is present in the drive, and Search removable media is ticked, then Press **Next**.

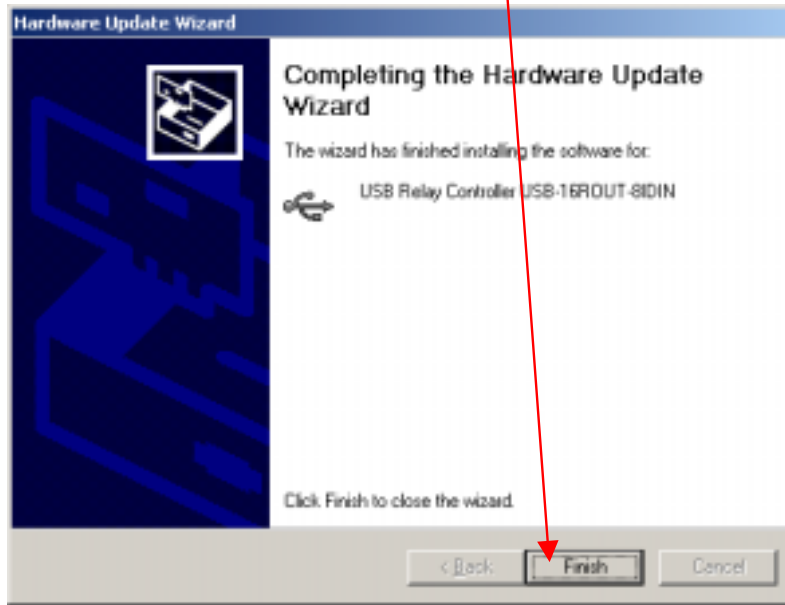
If you have Windows XP, the following driver warning will appear, please select **Continue Anyway**.



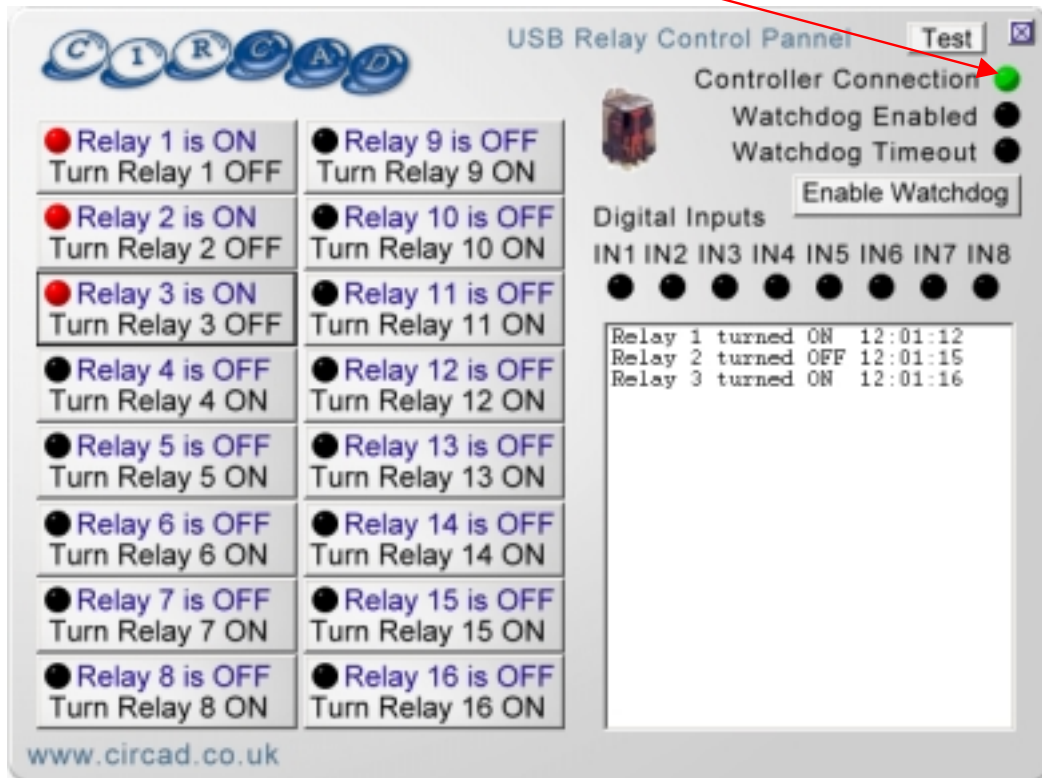
Windows will now install device drivers for the USB-16ROUT-8IDIN controller.



Press **Finish** to complete the Driver installation.



After installing the device driver, start the 16ROUT8IDIN Application (please see Page 8). The **Controller Connection** LED will light.



Control Software

An example Windows application is supplied with the 16ROUT8IDIN controller demonstrating the use of the DLL.

After installing the controller software from CD, a new ICON will appear on your desktop

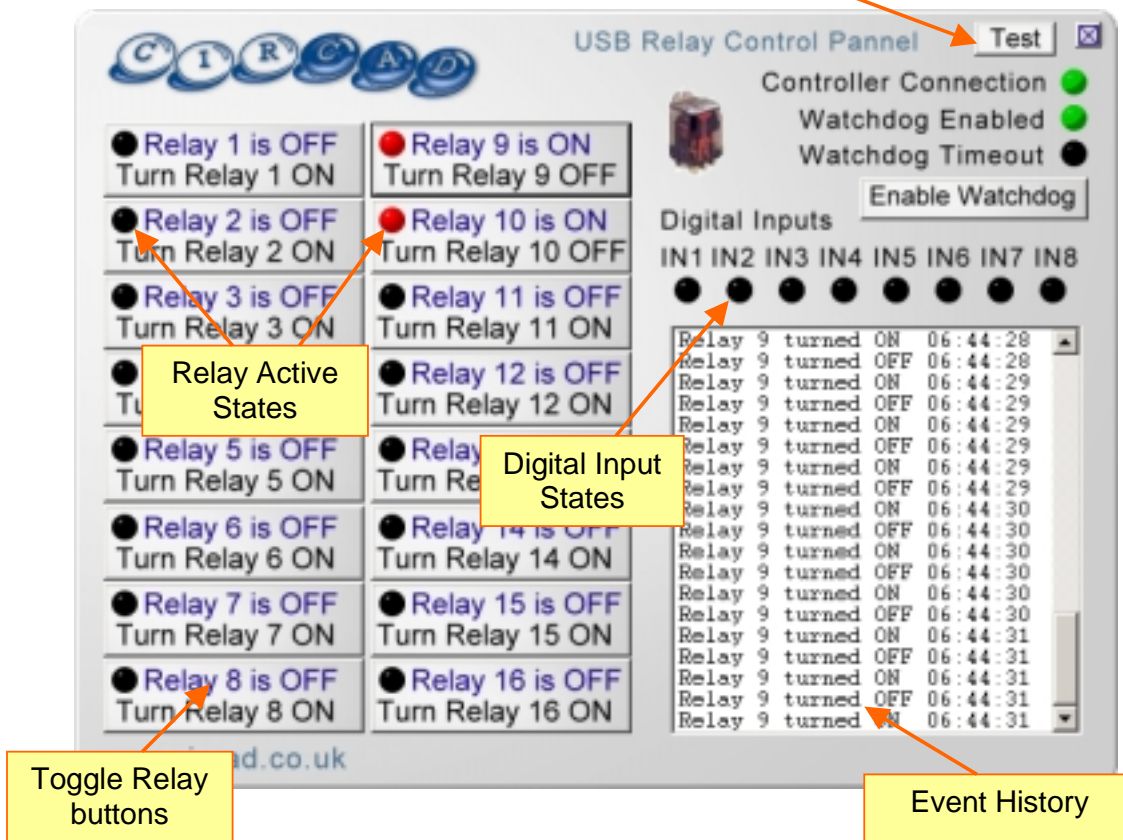


16Rout8IDin Application.Ink

Click on this link to start the control software.

This simple application allows you to turn on and off all 16 relays, and detect the state of the 8 Digital inputs.

The “Test” button will turn on all 16 relays then run through them all turning off one relay at a time. This program is actually used in the factory to verify the controller hardware is operating correctly.

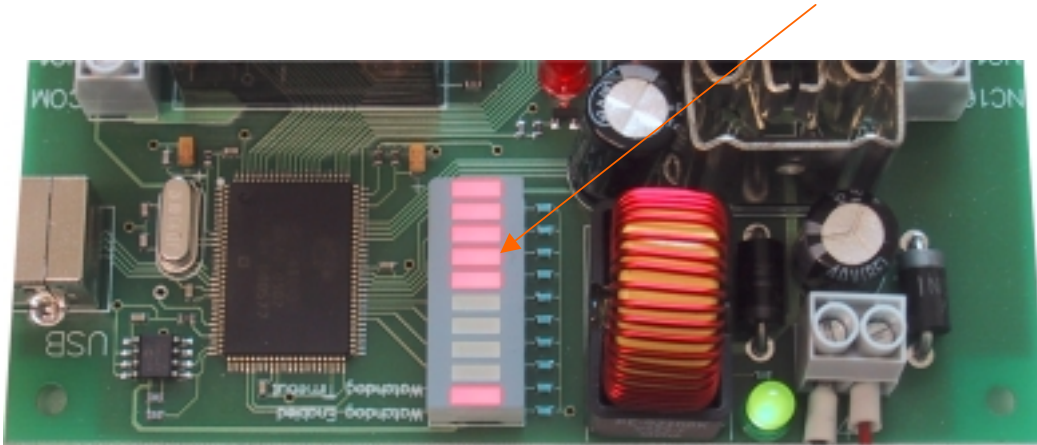


The “Enable Watchdog” button will start the watchdog timer running.

The watchdog timer is used to ensure that if the controller software crashes or the USB communications from the Host PC to the controller is broken all relays will be disengaged.

To use the watchdog timer in your application you must first enable the Watchdog, then maintain a constant access at least once a second.

The watchdog countdown can be seen on the bar graph display.



For more information on how to use the watchdog in your program please see the `cd_EnableWatchdog` function definition.

Software Interface

The software interface is made by these easy to use functions.

```
int cd_Heartbeat(void);
int cd_SetInterface(void);

int cd_RelayOn(unsigned char relayNumber);
int cd_RelayOff(unsigned char relayNumber);

int cd_Get16Relays(unsigned short *relayBits);
int cd_Set16Relays(unsigned short relayBits);

int cd_Get8Inputs(unsigned char *bits,
                 unsigned char *WatchdogEnabled,
                 unsigned char *WatchdogTimeout);

int cd_EnableWatchdog(void);
```

cd_Heartbeat

```
int cd_Heartbeat(void);
```

<Call> **void**

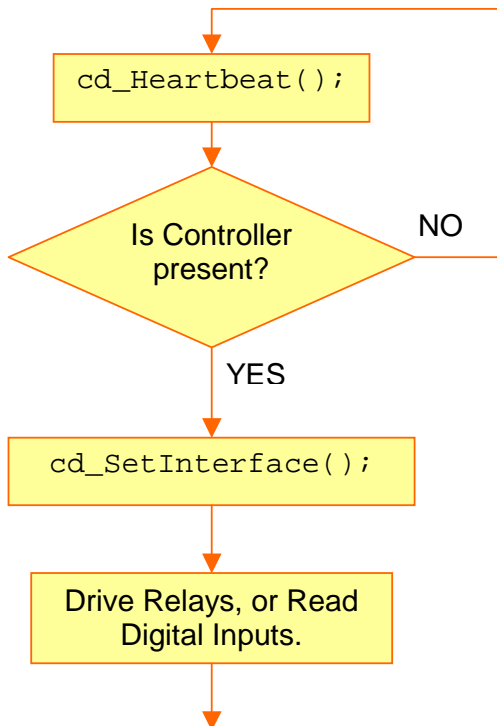
<Return> **TRUE** or **FALSE**.

This function is used to test the USB connection to the Relay controller. It will return **TRUE** if a connection has been made, and **FALSE** if no controller was detected.

The DLL will automatically open and close the device handler for the controller.

The first time you connect to the controller you will need to set up the USB endpoint configuration using the `cd_SetInterface` Function, else all other functions will return error 81h indicating a USB Communications failure.

Program flow.



cd_SetInterface

```
int cd_SetInterface(void);
```

<Call> **void**

<Return> 00 = Good, others See Error Codes.

The `cd_SetInterface` Function sets up the endpoint size used by the R16OUT8IDIN USB controller to respond back to the HOST PC. The PC USB interface determines the Endpoint size.

Run the `cd_SetInterface` function each time a new connection is established... i.e. when the controller is powered on, or the cable is first connected to the controller.

You can run this function a number of times on an established connection, however be aware that it does take about 300 milliseconds to run, so it will slow your application down if called repeatedly.

This is the only controller management function you need to run, there is no requirement to open and close a device handle.

Example:

```
void OnTimer(UINT nIDEvent)
{
    if(!cd_Heartbeat() && g_connected)
    {
        g_connected=false;
        return;
    }
    if(cd_Heartbeat() && !g_connected)
    {
        if(ErrorMessage(cd_SetInterface()))
            return;
        g_connected=true;
        return;
    }
}
```

cd_RelayOn

```
int cd_RelayOn(unsigned char relayNumber);
```

<Call> unsigned char (8 Bit) number of relay to engage.

<Return> 00 = Good, others see Error Codes.

Calling this function will engage one Relay; the state of all the other relays will remain unchanged.

Valid relay numbers are in the range of 1 through to 16, any number outside this range will result in an error message.

cd_RelayOff

```
int cd_RelayOff(unsigned char relayNumber);
```

<Call> unsigned char (8 Bit number) number of relay to disengage.

<Return> 00 = Good, others see Error Codes.

Calling this function will disengage one Relay; the state of all the other relays will remain unchanged.

Valid relay numbers are in the range of 1 through to 16, any number outside this range will result in an error message.

cd_Get16Relays

```
int cd_Get16Relays(unsigned short *relayBits);
```

<Call> (*relayBits) Pointer to an unsigned short (16 Bit) variable to hold all 16 relay states.

<Return> 00 = Good, others see Error Codes.

This function is used to get the engaged states of all 16 Relays. The bit number + 1 of the unsigned short result relates to the Relay number.

Bit 0 = Relay 1 State. Set = Engaged, Reset = Disengaged.

Bit 1 = Relay 2 State. Set = Engaged, Reset = Disengaged.

Through to ...

Bit 14 = Relay 15 State. Set = Engaged, Reset = Disengaged.

Bit 15 = Relay 16 State. Set = Engaged, Reset = Disengaged.

Example Code:

```
// Toggle Relay 4
unsigned short relayBits;
if(ErrorMessage(cd_Get16Relays(&relayBits)))
    return;

if(relayBits & 0x0008) // Relay is on so turn it off
    if(ErrorMessage(cd_Set16Relays(relayBits & 0xfff7)))
        return;
else // Relay is off so turn it on
    if(ErrorMessage(cd_Set16Relays(relayBits | 0x0008)))
        return;
```

cd_Set16Relays

```
int cd_Set16Relays(unsigned short relayBits);
```

<Call> (relayBits) unsigned short (16 Bit) to set all 16 Relay states.

<Return> 00 = Good, others see Error Codes.

This function is used to engage or disengage the states of all 16 Relays. The bit number + 1 of the unsigned short result relates to the Relay number.

Bit 0 = Relay 1 State. Set = Engaged, Reset = Disengaged.

Bit 1 = Relay 2 State. Set = Engaged, Reset = Disengaged.

Through to ...

Bit 14 = Relay 15 State. Set = Engaged, Reset = Disengaged.

Bit 15 = Relay 16 State. Set = Engaged, Reset = Disengaged.

Example Code:

```
// Toggle Relay 4
unsigned short relayBits;
if(ErrorMessage(cd_Get16Relays(&relayBits)))
    return;

if(relayBits & 0x0008) // Relay is on so turn it off
    if(ErrorMessage(cd_Set16Relays(relayBits & 0xfff7)))
        return;
else // Relay is off so turn it on
    if(ErrorMessage(cd_Set16Relays(relayBits | 0x0008)))
        return;
```

cd_Get8Inputs

```
int cd_Get8Inputs(unsigned char *bits,
                 unsigned char *watchdogEnabled,
                 unsigned char *watchdogTimeout);
```

<Call> (*bits) unsigned char pointer (8 bits). Pointer to variable to hold the value of the states of the 8 digital inputs.

(*watchdogEnabled) unsigned char pointer (bits). Pointer to variable to hold state of Watchdog Enabled.

(*watchdogTimeout unsigned char pointer (bits). Pointer to variable to hold state of Watchdog Timeout.

<Return> 00 = Good, others see Error Codes.

This function is used to read the state of the 8 digital inputs. The bit number + 1 = the digital input state.

Bit 0 = Digital Input 1 (State = 1 for ON , 0 for OFF)

...

Bit 7 = Digital Input 8 (State = 1 for ON , 0 for OFF)

This function is also used to detect the watchdog status. If the watchdog is not being used, you must still send a valid pointer to two unsigned char variables when calling.

Watchdog Enabled = **TRUE** if enabled, **FALSE** is not.

Watchdog Timeout = **TRUE** if the watchdog has timed out **FALSE** if it is still running.

The state of both of these variables can be seen on the LED Bar Graph on the controller.

Call this function to reset the watchdog timer. If you are using the watchdog it is essential to run this command at least once a second. If the watchdog times out, it must be reset by using the `cd_EnableWatchdog` before any other command will be accepted by the controller.

cd_EnableWatchdog

```
int cd_EnableWatchdog(void);
```

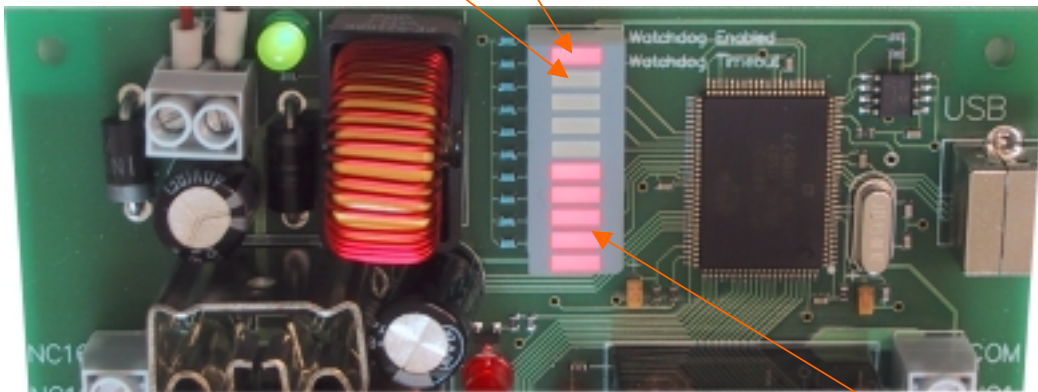
<Call> Void.

<Return> 00 = Good, others see Error Codes.

This function is used to enable the watchdog timer and reset the timeout. If enabled you need to call this function (or the `cd_Get8Inputs`) At least once a second.

Once the watchdog timer has expired all the Relays will be disengaged. They can not be reset until the `cd_EnableWatchdog` function has been called again.

The Watchdog timeout and enable states are displayed on the bar graph LED on the R16OUT8IDIN controller.



The countdown timer is indicated by the remainder of the LED Bar Graph.

Error Codes

All functions will return the same set of error codes.

Return codes:

- 00h Good Function call.
No failure.

- 80h Watchdog time out failure.
Occurs when the watchdog has countdown has expired.
Function `cd_EnableWatchdog()` will reset the Counter
and re-start relay control.

- 81h USB communications failure.
A failure to communicate with the controller. Ensure that
the `cd_Heartbeat()` function returns a TRUE value
before attempting any other function call.

- 87h Relay Out of range (<1).
Relay must be in the range of 1 to 16;

- 88h Relay Out of range (>16).
Relay must be in the range of 1 to 16.