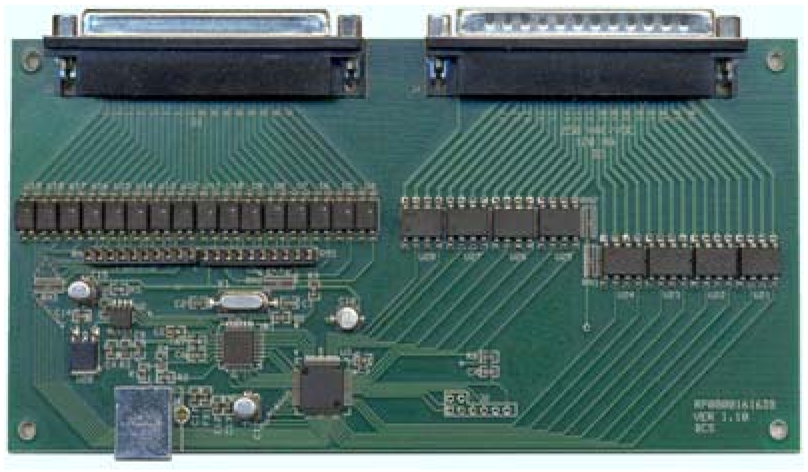
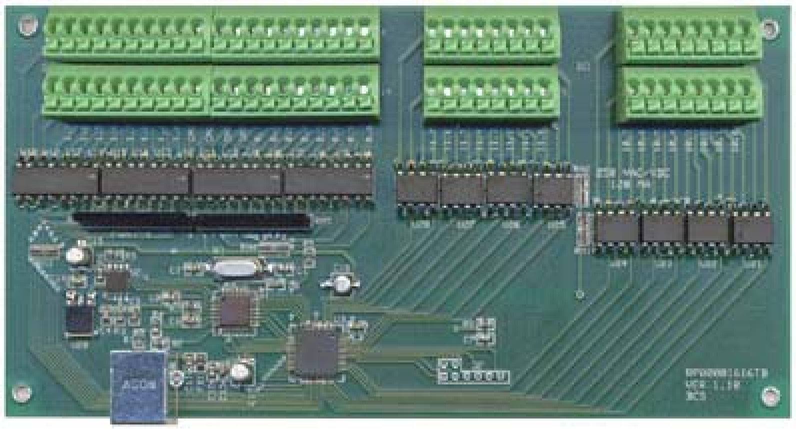
RP0001616DB
RP0001616TB





INEXPENSIVE, RELIABLE USB PRODUCTS

www.bcsideas.com

Table of Contents

The RP00001616DB/TB has 16 buffered Digital Inputs and 16 buffered Digital Outputs. The unit is controlled through a USB connection. The board is bus powered and will draw a maximum of 250 ma from the USB Bus. This requires that the RP00001616DB/TB be connected directly to a computer's USB port or to a port of a powered USB hub. Reading and writing the DIO is done through a DLL (dynamic link library). Therefore most popular programming languages (C++builder, VisualC, Visual Basic, NI's Measurement Studio, etc.) will be able to access the routines needed to control the DIO board. The board can also be accessed from an action step in NI's TestStand using the DLL Flexible Prototype Adapter.

The digital inputs are optically isolated transistors. Both leads of the opto-isolator are made available to the user. This gives the designer the advantage of being able to use a variety of DC voltages on the inputs. They require a current limiting resistor in series with each input and work with DC voltages only. Each input is sensitive to 2ma. Two 820-ohm SIP resistor networks have been supplied to act as pull ups for the Digital Inputs. When they are installed, the user can then connect the cathode of any of the inputs to the ground supplied on the input connector to signal an input event. The Resistor networks can be installed in the socket labeled RN5 for Port 0 or RN6 for Port 1.

See figures 1 through 3 for the correct pinouts for each model.

IMPORTANT! Failure to use the appropriate current limiting resistor in series with an input will destroy that input and void the warranty.

The digital outputs are solid state relays. Each output is completely separate from the others giving the designer flexibility in the types of devices that can be controlled. They switch loads up to 250 Vac/Vdc at up to 120 ma. The maximum on resistance is 30 ohms.

IMPORTANT! Placing a load larger than 250 Vac/Vdc at up to 120 ma on an output will destroy that output and void the warranty.

Digital Inputs

Absolute Maximum Ratings		
Parameter	**Value**	**Units**
Total Device Power Dissipation @ 25C	200	mW
DC/Average Forward Input Current	50	mA
Reverse Input Voltage	6	V
LED Power Dissipation @ 25C	70	mW

Electrical Characteristics				
Parameter	**Min**	**Typ**	**Max**	**Units**
Input Forward Voltage (Forward Current = 20 mA)		1.2	1.4	V
Reverse Leakage Current (Reverse Voltage = 4.0 V)			10	uA
Input Output Isolation Voltage			5000	Vac (rms)
Rise Time		4	18	uS
Fall Time		3	18	uS

Digital Outputs

Electrical Characteristics				
Parameter	**Min**	**Typ**	**Max**	**Units**
Switching Voltage (Load Current = 50mA)			250	V
Switching Current			120	mA
On Resistance		20	30	Ohm
Turn On Time			1	mS
Turn Off Time			1	mS
Dialectric Strength	1500			V(rms)

Dimensions

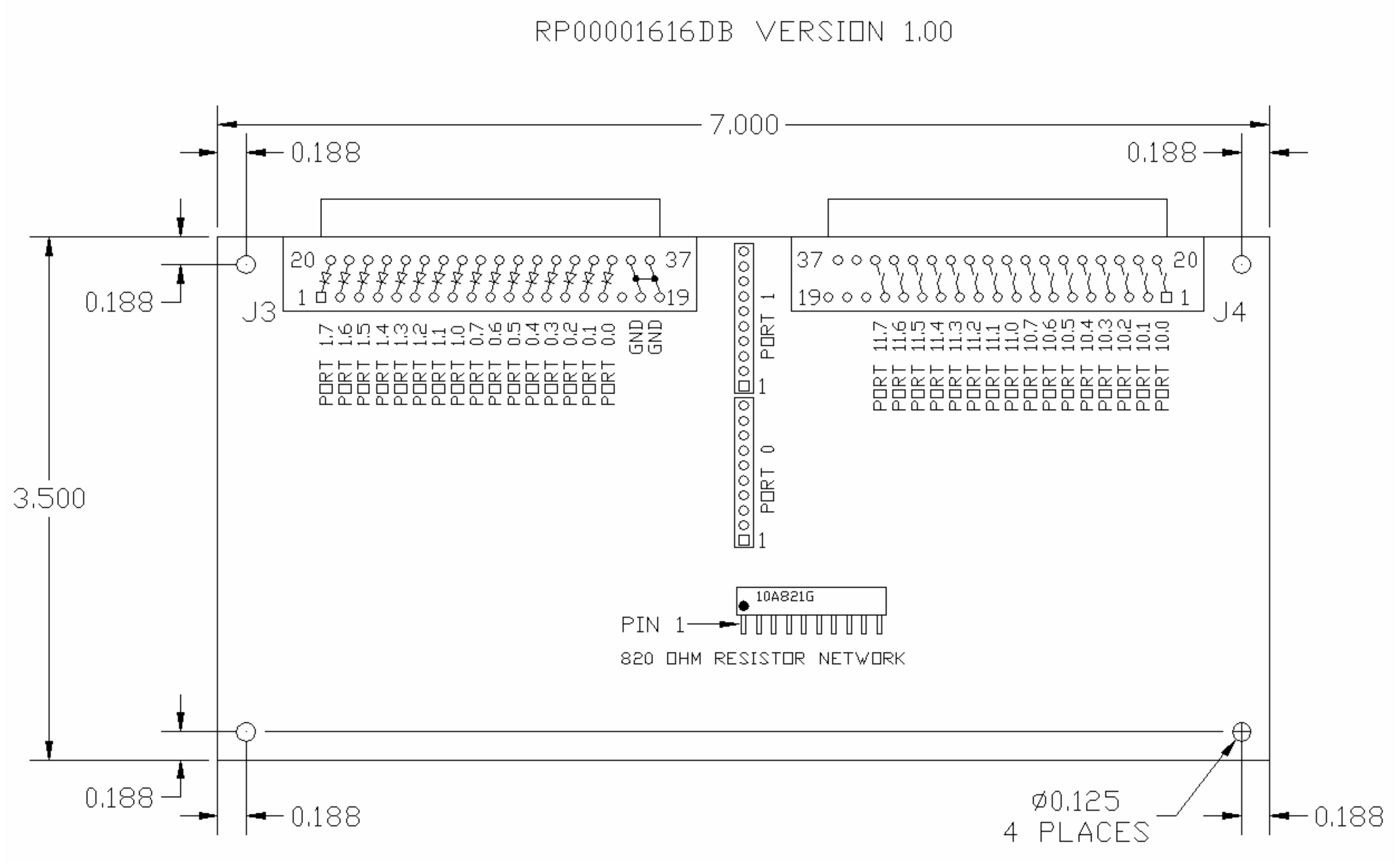


Figure 1 - RP00001616DB Version 1.00

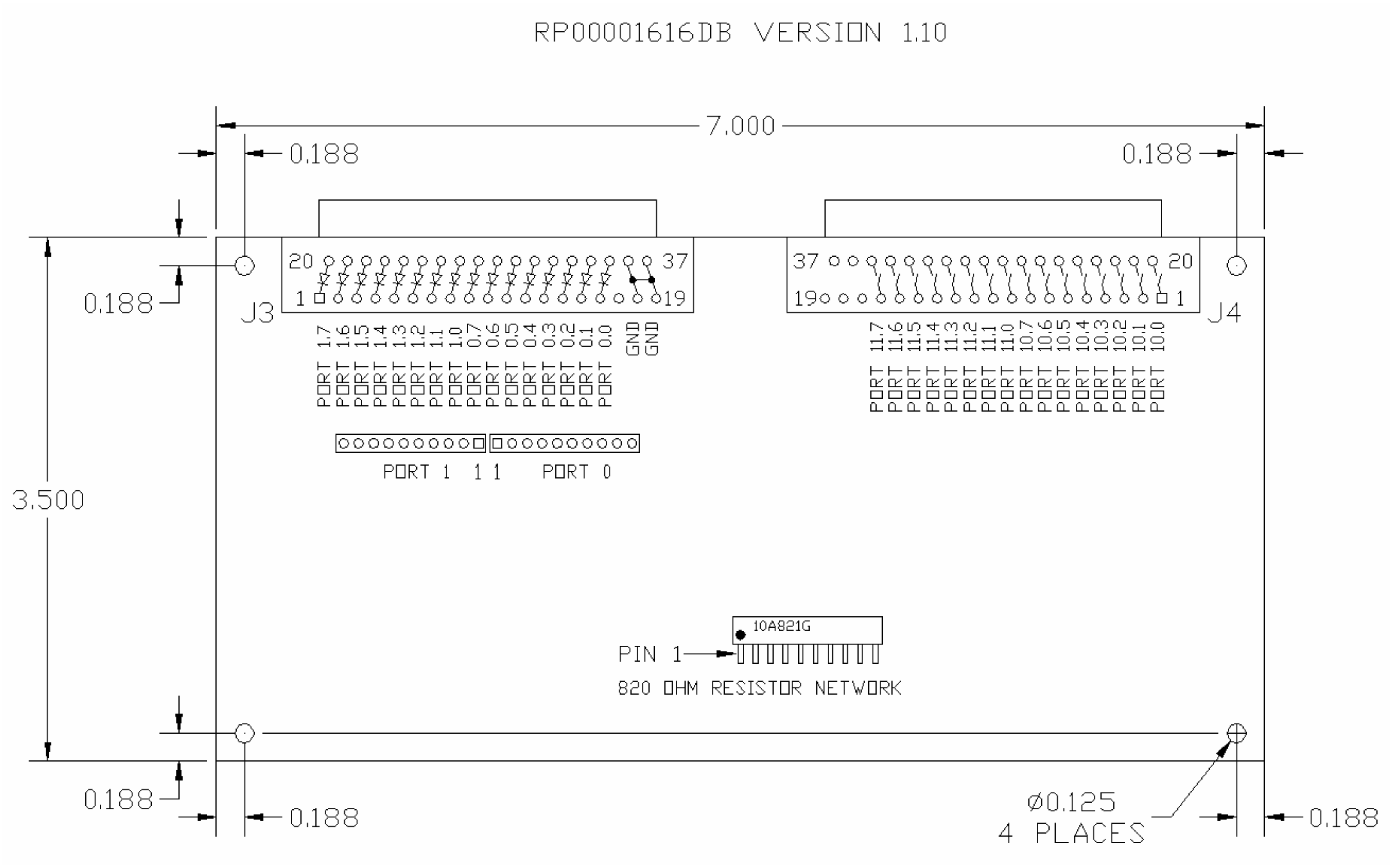


Figure 2 - RP00001616DB Version 1.10

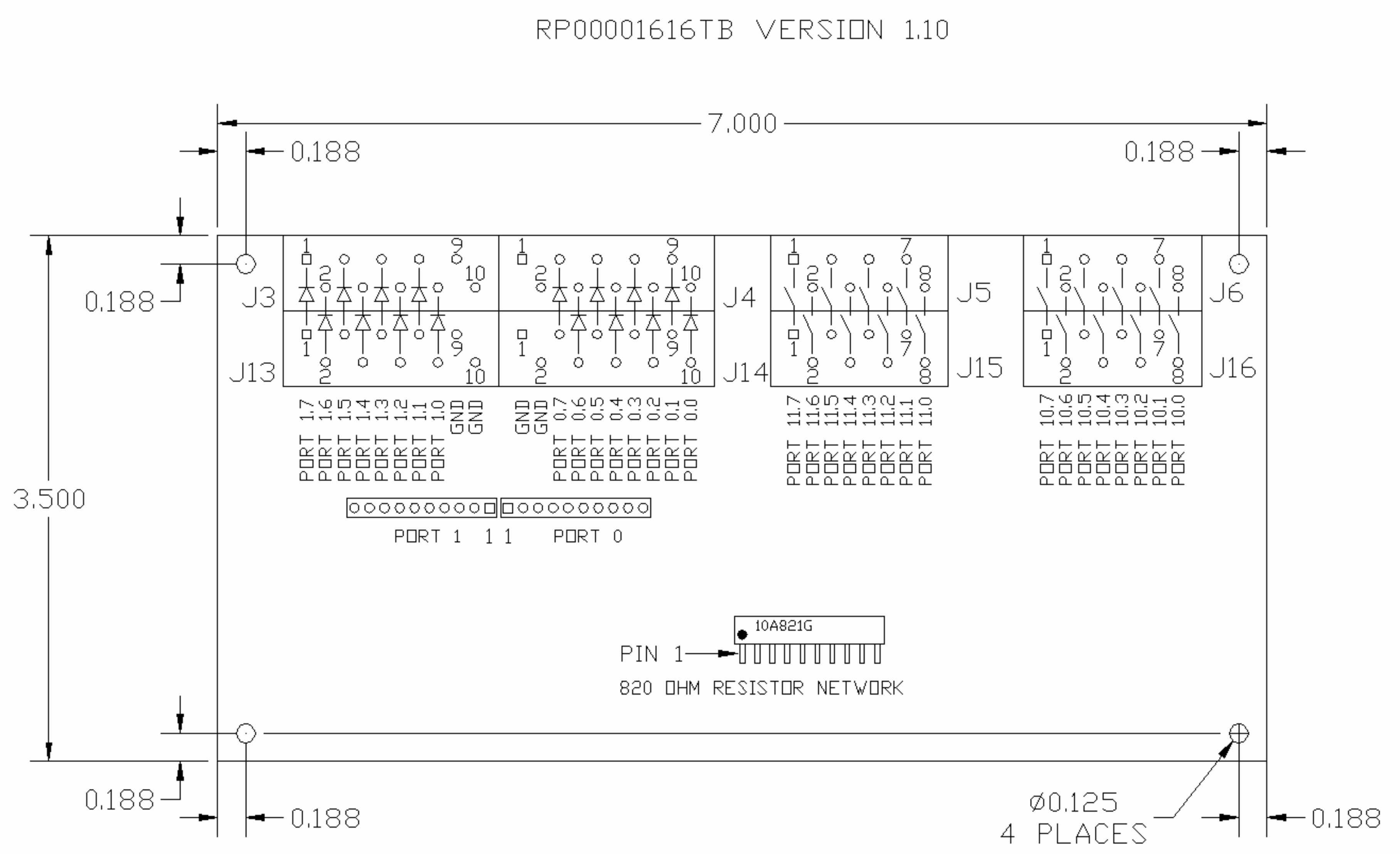


Figure 3 - RP00001616TB Version 1.10

Pinout

The DIO connectors for the RP00001616DB are both 37 Dsub connectors. The input connector is female and labeled J3. The output connector is male and labeled J4. Figures 1 and 2 show the pinouts for each connector.

The DIO connectors for the RP00001616TB are spring cage terminal blocks. The input connectors are labeled J3, J13, J4 and J14. The output connectors are labeled J5, J15, J6 and J16. Figure 3 shows the pinouts for each connector.

Warranty

The RP00001616DB/TB is warranted for 1 year. If within the first year of ownership the RP00001616DB/TB fails while being used within the specifications of the board it will be replaced with a new one. The user will be responsible for shipping the old board back to BCS. If it is determined that the board has been misused in any way the warranty will be void.

Installation

Install the RP00001616DB/TB as follows:

1. Plug the RP00001616DB/TB into a computer’s USB Port or a powered USB Hub. The operating system will acknowledge new hardware.

2. When prompted, browse the New Hardware Wizard to the subdirectory **\USB_Driver** on the CDROM.

3. Select the file **FTD2xx.inf**. The operating system will then load the necessary files for the RP00001616DB/TB to work on the computer. The system will acknowledge the installation of the new hardware.

4. To finish installation, browse to the **Root** directory on the CDROM. Run the program called **Setup.exe**. This will install the code example, documentation and an executable for testing the functionality of the RP00001616DB/TB.

Software

There are 3 files included to assist in using RP00001616DB/TB. They are RP2005.dll, FTD2xx.dll and RP2005.lib.

FTD2xx.dll – This file is used by RP2005.dll and needs to reside on the machine in order to communicate with the digital IO. The file will be placed in the System32 subdirectory by the setup program located on the CDROM.

RP2005.dll – This file contains all the functions needed to use the RP00001616DB/TB. A brief description of each function can be found below. Sample code is also supplied and can be found at <Install Dir>\BCS\RP2005\Software\Sample Code\. The default for <Install Dir> is C:\Program Files\. This file can be included in a software project or it can be used directly by a test executive such as National Instruments TestStand. The file will be placed in the System32 subdirectory by the setup program located on the CDROM.

RP2005.lib – This file is used by the software project that will be created to use the RP00001616DB/TB. RP2005.dll was written using Visual C/C++ therefore the supplied import library file will work with Microsoft programming products. Other development environments like National Instruments Measurement Studio or Boland C++Builder will not be able to use this import library. Both products have an easy to use utility for creating a compatible import library. The file should be placed in the software project's folder. Include this file in the project. Sample code has been supplied to show how to use the functions in RP2005.dll. The file will be placed in <Install Dir>\BCS\RP2005\Software\Sample Code\. The default for <Install Dir> is C:\Program Files\.

Functions in RP2005.DLL

The following functions are available in RP2005.DLL.

RP_ListDIO

Get information concerning the devices currently connected. This function returns the number of devices connected, and each device's serial number and product description.

unsigned long **RP_ListDIO** (int * *iNumBrds*, char * *SN*, char * *Desc*)

Parameters

iNumDev	The number of RP2005 devices currently attached to USB
SN	Comma delimited string containing the serial number of each RP2005 device currently attached to USB
Desc	Comma delimited string containing the device description of each RP2005 device currently attached to USB

Return Value
0 if successful, otherwise the return value is an error code.

Remarks
This function is used to return each device's serial number. The serial number is then used by **RP_Open** to obtain a handle for subsequent reading and writing of DIO.

Examples
Sample code shows how to get the number of devices, each serial number and each description.

```
unsigned long ulErrCode;
int iNumDevs;
char SN[256];
char Desc[256];

ulErrCode = RP_ListDIO( &numDevs, SN, Desc );
if (ulErrCode == 0)
{
  // Do something
}
else
{
  // Handle error
}
```

RP_OpenDIO

Open the device and return a handle which will be used for subsequent reading and writing of DIO.

unsigned long **RP_OpenDIO** (char * *SN*, unsigned long **hDIO*)

Parameters

SN	The serial number for the device.
hDIO	Pointer to a variable of type long where the handle will be stored. This handle must be used to access the device.

Return Value
0 if successful, otherwise the return value is an error code.

Remarks

Example
This sample shows how to open a device.

```
unsigned long hDIO; // handle of an open device
unsigned long ulErrCode;
char SN[256];
char Desc[256];

ulErrCode = RP_ListDIO( &numDevs, SN, Desc );
if (( ulErrCode == 0 ) && ( numDevs == 1 ))
{
  ulErrCode = RP_OpenDIO( SN, & hDIO );
  if ( ulErrCode == 0 )
  {
    // Do something
  }
  else
  {
    // Handle error
  }
}
else
{
  // Handle error
}
```

RP_ CloseDIO

Close an open device.

unsigned long **RP_ CloseDIO**(unsigned long *hDIO*)

Parameters

hDIO	Handle of the device to close.

Return Value
0 if successful, otherwise the return value is an error code.

Example
This sample shows how to close a device.

```
unsigned long hDIO;
unsigned long ulErrCode;
char SN[256];
char Desc[256];

ulErrCode = RP_ListDIO( &numDevs, SN, Desc );
if (( ulErrCode == 0 ) && ( numDevs == 1 ))
{
  ulErrCode = RP_OpenDIO( SN, & hDIO );
  if ( ulErrCode == 0 )
  {
    // Do something
    ulErrCode = RP_CloseDIO( hDIO );
  }
  else
  {
    // Handle error
  }
}
else
{
  // Handle error
}
```

RP_ReadPort

Read data from the device.

unsigned long **RP_ReadPort**(unsigned long *hDIO,* unsigned char *ucPort,* unsigned char * *ucPVal,* char * *ErrMsg*)

Parameters

hDIO	Handle of the device to read.
ucPort	The number of the port to be read.
ucPVal	Pointer to a variable of type unsigned char which receives the value of the port.
ErrMsg	String containing any error messages.

Return Value

0 if successful, otherwise the return value is an error code.

Remarks

The function does not return until the requested port has been read or read timeout occurs. The read timeout is set to 1 second.

The parameter *ucPort* represents either input port A, input port B, output port A or output port B. A value of 0 will access input port A, a value of 1 will access input port B, a value of 16 (0x10) will access output port A, a value of 17 (0x11) will access output port B. Any other value will return an error.

The parameter *ucPVal* represents the value of the requested port. A value of 0 (00000000 binary) means all 8 bits are active. A value of 255 (11111111 binary) means all 8 bits are off.

Example
This sample shows how to read bit 6 of input port A.

```
unsigned long hDIO; // handle of an open device
unsigned long ulErrCode;
unsigned char ucPort = 0;
unsigned char ucPVal, ucBit6;
char ErrMsg[256];

 ulErrCode = RP_ReadPort(hDIO, ucPort, &ucPVal, ErrMsg);
if (ulErrCode == 0)
{
 ucBit6 = ucPVal & 0x40; // 0100 0000
  // Do something
}
else
{
  // Handle error
}
```

RP_WritePort

Set bits for an output port.

unsigned long **RP_WritePort** (unsigned long *hDIO,* unsigned char u*cPort,* unsigned char *ucPVal,* char * *ErrMsg*)

Parameters

hDIO	Handle of the device to write.
ucPort	The number of the output port to be written.
ucPVal	The value to write to the output port.
ErrMsg	String containing any error messages.

Return Value

0 if successful, otherwise the return value is an error code.

Remarks

The function does not return until the requested port has been written or write timeout occurs. The write timeout is set to 1 second.

The parameter *ucPort* represents either output port A or output port B. A value of 16 (0x10) will access port A and a value of 17 (0x11) will access port B. Any other value will return an error.

The parameter *ucPVal* represents the value that the port will be set to. A value of 0 (00000000 binary) means all 8 bits are on (the solid state relays are closed). A value of 255 (11111111 binary) means all 8 bits are off (the solid state relays are open).

Example

This sample shows how to set bit 3 of output port A to 0.

```
unsigned long hDIO; // handle of an open device
unsigned long ulErrCode;
unsigned char ucPort = 0x10; // Output port A
unsigned char ucPVal;
char ErrMsg[256];

 ulErrCode = RP_ReadPort(hDIO, ucPort, &ucPVal, ErrMsg);
if (ulErrCode == 0)
{
  ucPVal &= 0xf7; // 1111 0111
  ulErrCode = RP_WritePort(hDIO, ucPort, ucPVal, ErrMsg);
  if (ulErrCode == 0)
  {
    ulErrCode = RP_ReadPort(hDIO, ucPort, &ucPVal, ErrMsg);
  }
  else
  {
    // Handle error
  }
}
else
{
  // Handle error
}
```

RP_GetVer

Read software version from the device.

unsigned long **RP_GetVer** (unsigned long *hDIO*, char **SWVer*, char **ErrMsg*)

Parameters

hDIO	Handle of the device to write.
SWVer	String containing the software version of the device.
ErrMsg	String containing any error messages.

Return Value
0 if successful, otherwise the return value is an error code.

Remarks
The function does not return until the requested information has been returned or read timeout occurs. The read timeout is set to 1 second.

Examples

This sample shows how to read the software version.

```
unsigned long hDIO; // handle of an open device
unsigned long ulErrCode;
 char ErrMsg[256];

 ulErrCode = RP_GetVer( hDIO, SWVer, ErrMsg );
if (ulErrCode == 0)
{
// Do something
}
else
{
  // Handle error
}
```

BCS

www.bcsideas.com

General Inquiries
info@bcsideas.com

Sales Information
sales@bcsideas.com

Product Support or Recommendations
support@bcsideas.com