TE3200

Dual Programmable
0-30 VDC/2A Power Supply

User's Manual

Team Solutions, Inc.

27071 Cabot Road, Suite 124

Laguna Hills, CA 92653

Handling and Safety

Team Solutions, Inc. (hereafter written as "Team") carefully tests and inspects every product before shipment.

The power supply is a delicate and electro-statically sensitive electronic instrument. Handle and install with extreme care. Do not remove it from its original box, packing, and ESD protective wrapping until you are ready to install it. You should save the original packaging for subsequent removal and storage, or future return to Team for repair.

Store the software CD in a place safe from heat or direct sunlight. Software is downloadable from our Internet web site at www.team-solutions.com.

Warranty

Team's products are warranted against defects in materials and workmanship for a period of 12 months. Team shall repair or replace (at Team's discretion) any defective product during the stated warranty period. If repair is needed, please call (949)348-7766 for a return authorization.

Customer Support

If you need assistance at any time with the installation or use of this product, call Team technical support at (949) 348-7766. Product information and software is available on our Internet web site at www.team-solutions.com.

Disclaimer

Team's products are not intended for medical use, including certification, calibration, or testing of medical instruments or devices.

In no event shall Team or any of its representatives be liable for any consequential damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other loss) arising out of the use of or inability to use this product, even if Team has been advised of the possibility for such damages.

Copyright

Trademarks

Borland C++	Borland Corporation
IBM or IBM-PC/AT	International Business Machines Corp. (IBM).

Table of Contents

Handling and Safety i
Warranty i
Customer Support i
Disclaimer i
Copyright i
Trademarks i
Chapter 1 – General Information 7
1.1 Overview 7
1.2 Firmware Capability 7
1.3 Graphical User Interface 8
1.4 Applicable Specification 8
1.5 Web Site Information 8
Company Contact Information: 9
Chapter 2 – PXI Description 11
2.1 Overview 11
2.2 PXI Capabilities 11
Chapter 3 – Unpacking, Inspection, System Requirements 13
3.1 Unpacking and Inspection 13
Unpacking and Inspection 13
Discharge Static Electricity 13
Packing List 13
3.2 Environmental and System Requirements 15
Chapter 4 – Installing the Hardware and Software 17
4.1 Installing the TE3200 17
4.2 Installing the Software Drivers for Windows 9x/Me/2000/XP 18
4.3 Installing the Application Software 22
Chapter 5 – Operation and Programming 23
5.1 Overview 23
5.2 TE3200 Graphical User Interface 23
5.3 DLL Configuration File 24
5.4 Constant Declarations 24

Programming with C/C++ Tools 25
Programming with Visual Basic 26
5.5 DLL Functions and Procedures 26
Chapter 6 – Functions Reference 27
6.1 General System Functions 27
te32_Close_all 27
te32_DLLVersion 27
te32_WinDriverVersion 27
te32_BoardPresent 27
te32_Model_Number 27
te32_xWaitFor 28
te32_Reset 28
6.2 TE3200 Control Functions 29
te32_FlashLed 29
te32_OutputRelay 29
te32_ PowerState 29
te32_Set_voltage 29
te32_Get_voltage 30
te32_Set_current 30
te32_Get_current 30
6.3 Measurement Functions 31
te32_Measure_Voltage 31
te32_Measure_Current 31
6.4 Status Functions 32
te32_Status 32
6.5 Calibration and Configuration Functions (EPROM access) 33
te32_EEProm_Read 33
te32_GetConfigValue 33
te32_GetCalibration 34
6.6 DLL Error Codes 35
Device Driver Errors 35
Range Errors - Current 35
Range Errors – Other 35
Misc. Errors 36

EPROM Errors 36
6.7 Visual Basic Applications 37
Programming tips 37
DLL Declaration Module 37
Constant Declaration Module 39
Appendix A – Calibration Tolerances 40
Calibration Tolerances 40
Appendix B – Specifications 41
General 41
Outputs 41
Input Power 41
Protection Circuitry 42
Physical 42

Chapter 1 – General Information

1.1 Overview

This TE3200 User's Manual provides all the information needed to install, program, and use Team's TE3200 dual programmable 0-30VDC/2A power supply. This manual assumes that the user has a general knowledge of personal computers, Windows 9x/Me/2000/NT operating systems and an electronics background. Some knowledge of programming and development tools will permit computer program control of the programmable DC power supply.

The TE3200 series of instruments are dual output power supplies on 3U size PXI card formats. The instruments are message-based devices and are provided with industry standard software drivers. A dual 48VDC input power connects to the TE3200 directly through a front panel mounted connector. This approach avoids the need for either special backplane modifications to route DC power to the module (although available as an option) or for excessively large backplane power supplies.

The TE3200 power supplies are packaged on standard PXI single-slot, 3U card formats. Each instrument is comprised of a Mother PCB containing the PXI interface and communications circuitry. The Mother PCB provides the necessary control circuitry for communications over the PXI backplane as well as providing storage of calibration data and default setting in on-board EEPROM.

1.1.1 OUTPUT STAGES

Designed for applications requiring higher power levels and signal quality, output power is provided by two independent and isolated output stages. The output stages are rated at 0-30 VDC, 2 Amps and are limited to 30 Watts per channel. Output stages operate at >80% efficiency.

The TE3200 power supplies are covered and shielded to eliminate EMI/RFI interference and to provide for controlled airflow. Each stage is joined to a cooling plate sized to provide adequate cooling up to 30W per channel with rated airflow (see PXI specification) even in instances where adjacent PXI card slots are vacant. To provide for maximum flexibility in meeting a broad range of UUT power requirements, output stages may be used in parallel or serial under software control. For example, an instrument configured with two 30VDC output stages may be used as a dual output module at 30W per output, or may be programmed as a single 0-30 VDC/60W supply in parallel mode, or as a single 0-60 VDC/30W supply in serial mode. To cover larger power requirements, additional TE3200 supplies may also be paralleled with some de-rating in performance specifications.

The TE3200 may be purchased with either optional AC-to-48VDC desktop style converters or a separate AC to dual 48VDC output module to allow the unit to operate from worldwide AC mains.

1.2 Firmware Capability

1.2.1 SOFTWARE CALIBRATION

The TE3200 instruments contain internal non-volatile EEPROM memory for storage of calibration and configuration data. Calibration is accomplished via backplane communications without the need for manual adjustments. It is not necessary to remove the instrument from the PXI card cage to perform calibration.

1.3 Graphical User Interface

Software drivers and examples for Visual Basic and Visual C are included on the CD. A graphical user interface is also included as an executable program that provides a quick and easy means of controlling the instrument. No knowledge of specific TE3200 instrument drivers or commands is needed when using the soft-panel (GUI).

1.4 Applicable Specification

The TE3200 instruments have been designed and manufactured to comply with the following standards and specifications:

MIL-STD-28800D Type III, Class 3

MIL-STD-45208 Quality Standards

MIL-STD-461C EMI Part 7 (For commercial test equipment)

VDE

UL

CE

CSA

IEC 435 Safety and Isolation

IEC 380 Safety Requirements

IEEE-488.2

PXI Specification Revision 2.0

1.5 Web Site Information

Team provides both pre-sales and post-sales technical support for all products. Our technical support engineers can help you select hardware and software for your application, understand the specifications, and assist you in selecting a complete system. Call our Technical Support department from 9:00AM to 5:00PM Pacific Standard Time (PST). Call (949) 348-7766 or send email to support@team-solutions.com.

Company Contact Information:

Team Solutions, Inc.

27071 Cabot Road, Suite 124

Laguna Hills, CA 92653

Phone: (949) 348-7766

Fax: (949) 348-7776

Sales: sales@team-solutions.com

Technical Support: support@team-solutions.com

Web site: www.team-solutions.com

Chapter 2 – PXI Description

2.1 Overview

The goal of PXI is to define a technically sound modular instrument standard based on the PCI standard which is open to all manufacturers and is compatible with present industry standards.

The PXI specification details the technical requirements of PXI compatible components such as mainframe, backplane and logic power supplies. The specification also provides for interconnecting and operating different manufacturers' products with the same card chassis. The TE3200 has been designed to be compliant with the current revision of the PXI specification. You may download the latest revision of the PXI specification from http://www.pxisa.org.

2.2 PXI Capabilities

The TE3200 does not implement any of the following PXI capabilities as no J2 connector is installed.

Local Bus - A daisy-chained bus that connects each peripheral slot with its adjacent peripheral slots to the left and right. Thus, the right local bus of a given peripheral slot connects to the left local bus of the adjacent slot, and so on. Each local bus is 13 lines wide and can be used to pass analog signals between modules or to provide a high-speed side-band digital communication path that does not affect the PXI bandwidth.

Star Trigger - The local bus lines for the leftmost peripheral slot of a PXI backplane are used for the star trigger. The star trigger bus implements a dedicated trigger line between the first peripheral slot (adjacent to the system slot) and the other peripheral slots.

Trigger Bus – Up to eight triggers may be passed from one module to another, allowing precisely timed responses to asynchronous external events that are being monitored or controlled.

System Reference Clock - The PXI 10 MHz system clock (PXI_CLK10) can be used for synchronization of instruments.

Chapter 3 –Unpacking, Inspection, System Requirements

3.1 Unpacking and Inspection

Unpacking and Inspection

Caution: Static-sensitive instrument! Ground yourself to discharge static.

1. Remove the TE3200 from the static bag by handling only the metal portions.
2. Check the contents of the shipping carton to verify that all of the items found in it match the packing list.
3. Inspect the TE3200 for possible damage. If there is any sign of damage, return the TE3200 immediately. Please refer to the warranty information at the beginning of this manual.

Discharge Static Electricity

To reduce the risk of damaging the TE3200, observe the following precautions:

- Leave the TE3200 in the anti-static bag until you are ready to install it. The anti-static bag protects the power supply from harmful static electricity.
- Save the anti-static bag in case the TE3200 is removed from the computer in the future.
- Carefully unpack and install the TE3200. Do not drop the TE3200 or handle it roughly.

Packing List

The TE3200 is shipped from the factory with the following:

- TE3200 User's manual and software (on CD-ROM)
- TE3200 dual programmable 0 to 30VDC/2A power supply
- Optional: Two 48VDC output desktop power supplies (for 115/220 VAC operation)
- Optional: Front Panel Mating Connector Set (pins, mating connector, and back-shell)
- Optional: Screw-terminal /power input assembly (mates with the TE3200's HD DB15 connector via a standard VGA cable) that provides screw-terminal connections for both desktop power supplies, both outputs, and all sense lines. Provides jumpers for on-boaad sense line connections and single-power supply usage (as long as isolated grounds are not needed between the two outputs).

The TE3200 was carefully tested and inspected for mechanical and electrical defects prior to shipment. The TE3200 should be inspected for any visible damage that may have occurred in transit. If the shipping container is damaged or other damage is apparent, it is recommended to; a) report damage immediately to the carrier and to the factory, b) retain shipping container and, c) photograph any damage to the container or product.

3.2 Environmental and System Requirements

In order for the TE3200 to meet its specifications, the operating environment must be within the following limits:

Temperature	0 to +40 degrees C
Relative Humidity	20 - 95% (non-condensing)

The non-operating temperature specification is from –40 degrees C to +85 degrees C

The TE3200 is shipped from the factory pre-wired for the following front panel input power:

Two separate isolated +48VDC/1.25A/60W inputs are needed for dual output.

See the schematics in the "HD15pinout" directory on the CD-ROM for the connections.

The TE3200 requires the following input power from the PXI backplane:

+5VDC @ 1.0A

+12VDC @ 0.2A

Chapter 4 – Installing the Hardware and Software

4.1 Installing the TE3200

Caution: Turn the chassis power OFF before attempting installation. Do not attempt to insert or remove the power supply with the chassis power on.

---

1. Turn off the power to the PC.
2. Unplug the PXI chassis from the AC power outlet to avoid possible electrical shock.
3. Locate an available PXI slot,
4. Remove the slot cover plate

WARNING: Inserting the TE3200 in a PXI slot with the power to the PXI chassis turned on may damage the board, the PXI chassis, or both.

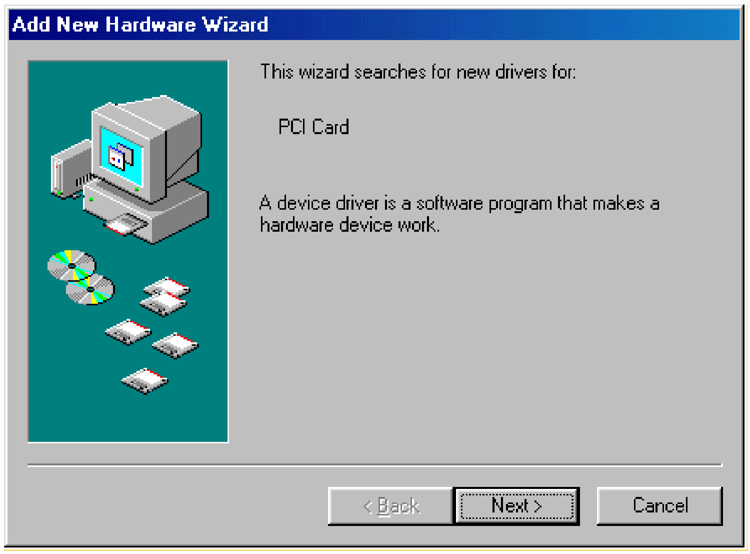
---

5. Insert the TE3200 board into the PXI slot. It should insert with minimal amount of force. Press in firmly yet gently on the top edge of the front bracket while lifting the bottom insertion lever, applying equal force.
6. Secure the TE3200 board by tightening the captive screws at the top and bottom of the front panel.
7. Connect the power cord to the back of the PXI chassis and plug it back into the AC wall outlet.

4.2 Installing the Software Drivers for Windows 9x/Me/2000/XP

When you install the TE3200 for the first time, Windows 9x/Me/2000/XP will automatically detect that you have a new device the first time you power on your PC and request the software drivers for the detected device. Please follow the step-by-step instructions below to properly install the required software drivers.

1. Below is a picture of the dialog box that will notify you that Windows has detected your new hardware.



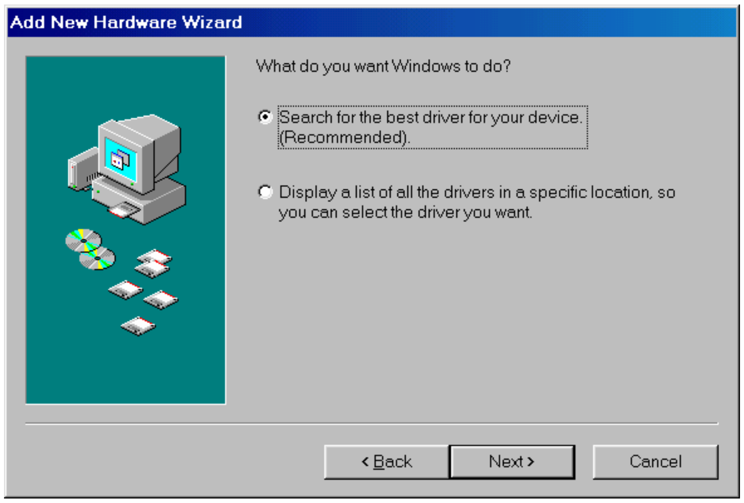
Add New Hardware Wizard

This wizard searches for new drivers for:

PCI Card

A device driver is a software program that makes a hardware device work.

2. Windows will now automatically run the "Add New Hardware Wizard". Select the first option "Search for the best driver…" and click the "Next" button.



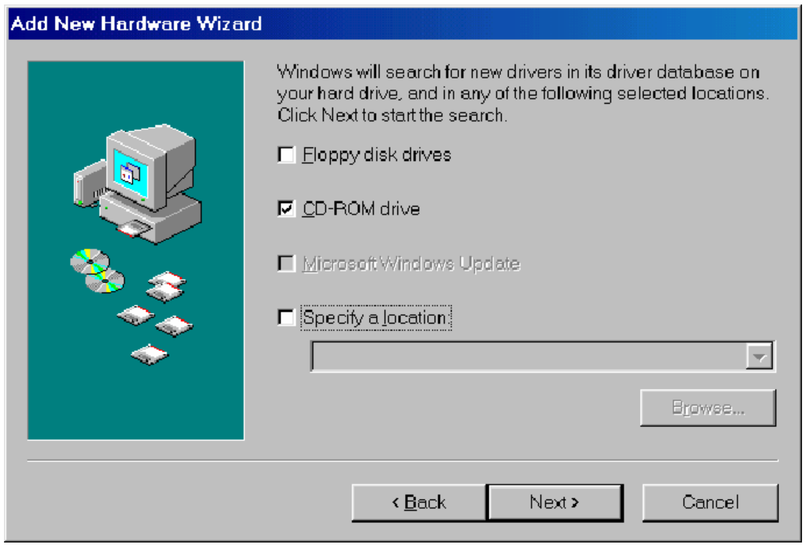
Add New Hardware Wizard

What do you want Windows to do?

[✓ Search for the best driver for your device. (Recommended).]

Display a list of all the drivers in a specific location, so you can select the driver you want.

3. You will now see a box that asks where you would like to search for the driver. Select the “CD-ROM drive” option. Make sure you have the TEAM installation CD in the CD-ROM drive. After you have done this, click on the “Next” button.

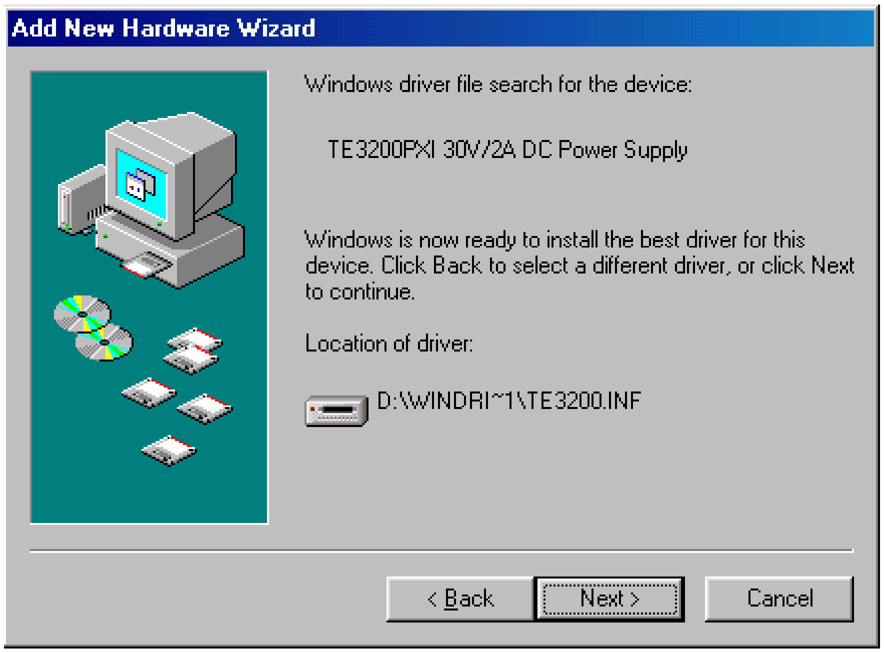


Add New Hardware Wizard

Windows will search for new drivers in its driver database on your hard drive, and in any of the following selected locations. Click Next to start the search.

- [ ] Floppy disk drives
- [✓] CD-ROM drive
- [ ] Microsoft Windows Update
- [ ] Specify a location

4. After Windows locates your new driver on the CD, it will ask if you want to install the driver (TE3200.inf). Select the first option that says, “The Updated Driver” then click on the “Next” button



Add New Hardware Wizard

Windows driver file search for the device:

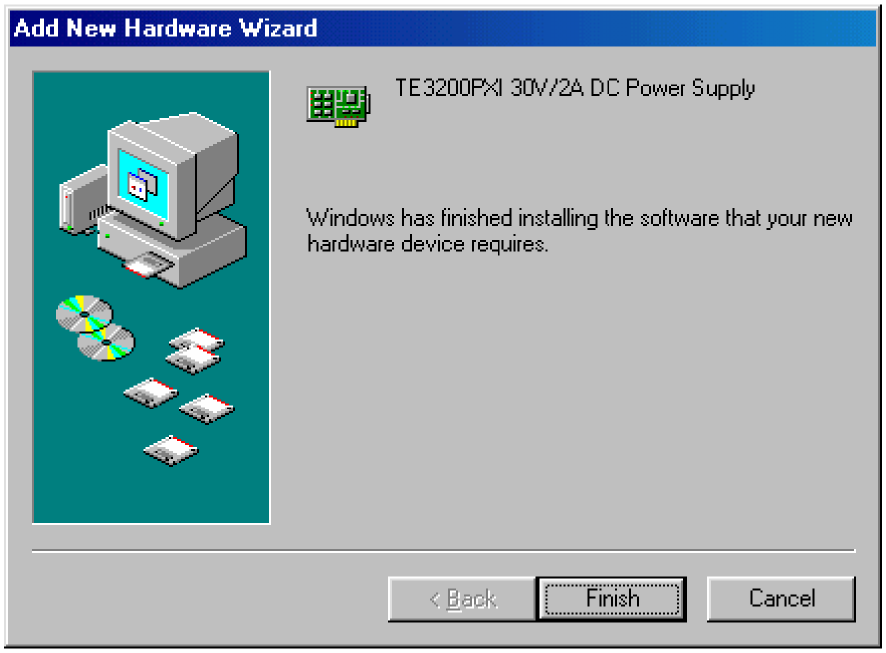
TE3200PXI 30V/2A DC Power Supply

Windows is now ready to install the best driver for this device. Click Back to select a different driver, or click Next to continue.

Location of driver:

D:\WINDRI~1\TE3200.INF

5. Windows will inform you that it is ready to install your new driver. Click on the "Next" button.

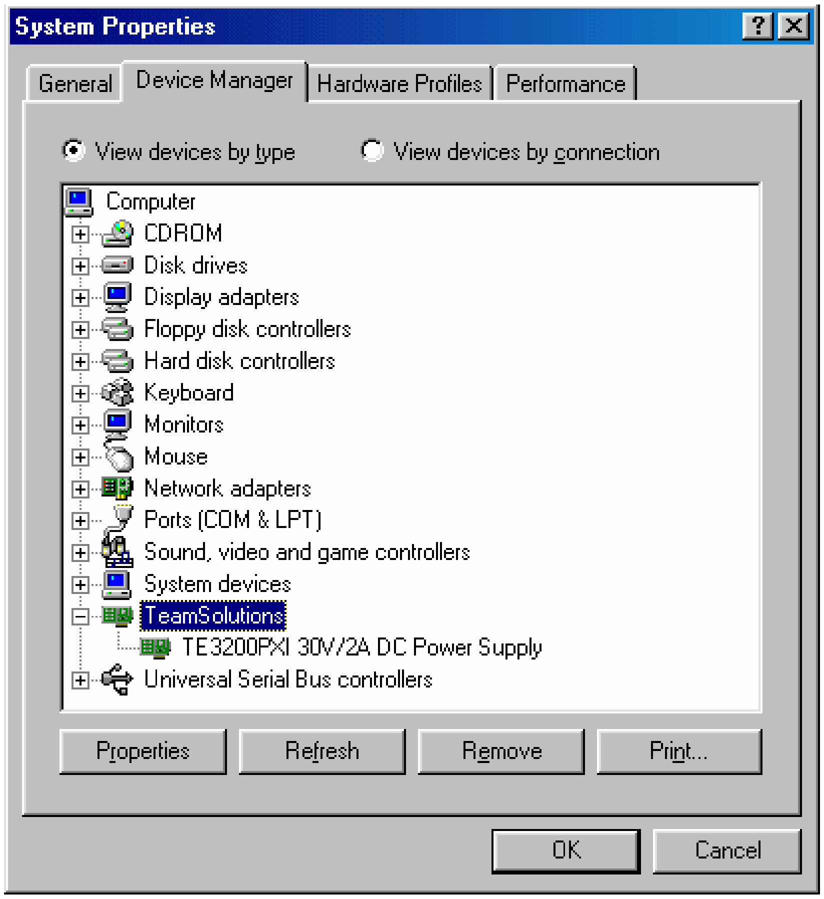


TE3200PXI 30V/2A DC Power Supply

Windows has finished installing the software that your new hardware device requires.

6. Driver installation is complete. Click the "Finish" button.

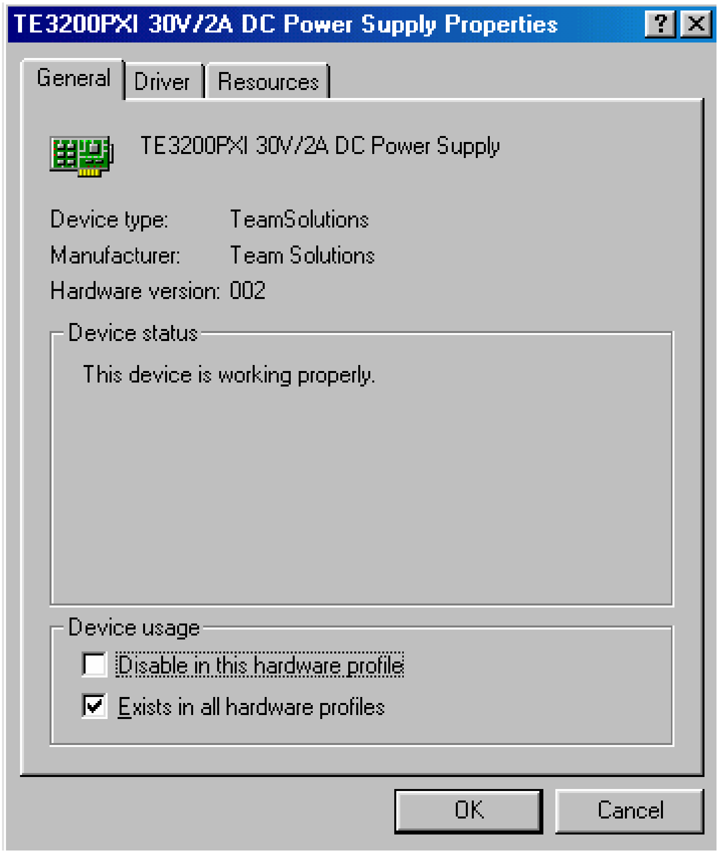
7. To check your installation, right click on the “My Computer” icon on the desktop, select Properties and select the Device Manager tab. You should see the following screen:



● View devices by type ○ View devices by connection

- Computer
  - CDROM
  - Disk drives
  - Display adapters
  - Floppy disk controllers
  - Hard disk controllers
  - Keyboard
  - Monitors
  - Mouse
  - Network adapters
  - Ports (COM & LPT)
  - Sound, video and game controllers
  - System devices
  - TeamSolutions
    - TE3200PXI 30V/2A DC Power Supply
  - Universal Serial Bus controllers

8. Double-click the Team Solutions Icon. You should see the following screen.


TE3200PXI 30V/2A DC Power Supply Properties

TE3200PXI 30V/2A DC Power Supply

Device type: TeamSolutions
Manufacturer: Team Solutions
Hardware version: 002

Device status
This device is working properly.

Device usage
☐ Disable in this hardware profile
☑ Exists in all hardware profiles

4.3 Installing the Application Software

To install the application software, insert the installation disk supplied and run the setup program. Run the setup program for the applicable operating system directory. This is "D:\Win_nt\te3200_win_nt.exe" for Windows NT/2K/XP or "D:\Win_95\te3200_win95.exe" for Windows 9x/Me. Follow the on screen instructions. The installation program will create the required shortcuts on the Windows desktop and a program group in the Start menu.

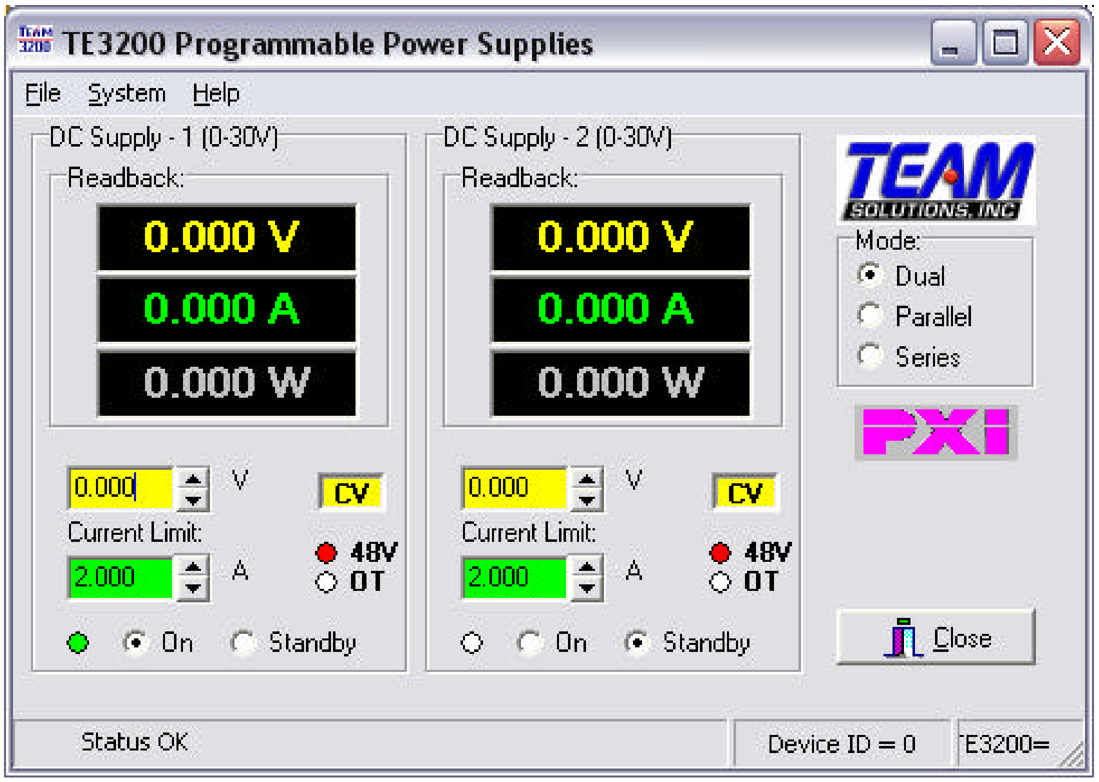
Chapter 5 – Operation and Programming

5.1 Overview

The TE3200 is a PXI/Compact PCI dual programmable 0-30 VDC/2A power supply. The quantity of TE3200 instruments that may be deployed in the same chassis are limited only by the environmental cooling, available slots, and available system resources. All installed units are controlled through a single Windows dynamic link library (TE3200ps.dll). The DLL provides the interface between the hardware and the end-user's application program.

As an alternative to writing custom code, a Windows Graphical User Interface (GUI) is provided which may be used for interactive control of the TE3200. This chapter describes the functions contained in the TE3200ps.dll and the function declarations. Samples are provided for calling the DLL from VB and VC.

5.2 TE3200 Graphical User Interface



The main GUI screen consists of two identical frames, one for each power supply output. If the supplies are operated in either Parallel or Series mode, only the left hand side frame (DC Supply 1) will be visible. In these modes, both supplies are controlled by the DC Supply 1 frame controls. The mode can be selected on the right hand side of the GUI window.
The top three fields with black backgrounds read out the Voltage, Current and Power measurements for each supply as long as the output relay for the supply is ON. To change settings, use the Voltage (Yellow) or Current (Green) spin controls below the measurement readout. You can also type in a value directly and press the "Enter" key or move to a different control to effect a setting change.

The status of each supply is shown directly to the right of the Voltage and Current controls. If the 48V supply feeding the PWM circuit is missing, the 48V indicator will light up and an error message will be visible in the status bar at the bottom. If an over temperature condition occurs, the supply will be shut down and the OT indicator will be on.
To open or close the output relay, use the On or Standby controls at the bottom of each frame.
To exit the program, click on the "Close" button or select File, Exit from the menu bar.

5.3 DLL Configuration File

Each time the DLL is initiated, it looks for a configuration settings file located in the TE3200 application directory. This file is called TE3200ps.ini and contains the following information and formatting:

```
[board_settings]
base0 = 0
Base1 = 8
Base2 = 16
Base3 = 24
```

Values shown here are defaults and will be used if one or more entries or the entire ini file is omitted. Base numbers correspond to board ID's, thus base0=0 indicates that the board with ID 0 will be the first board found. Offset values may be assigned to the installed boards as desired as long as there are no duplicates and the values are 8 apart. For most situations, there is no need to set the base values and the defaults may be used.
A unique board ID number ranging from 0 to *n* identifies each board. Every function or procedure call in the DLL has a board ID parameter. It is always a 32-bit integer type (**long** in VB).

The Windows 32-bit TE3200ps.dll can be used for applications that are developed for Windows 9x/Me/2000/XP.

To use the DLL driver from an application, the DLL must reside in one of the following directories:

- Application directory
- Windows directory (e.g., \Windows or \WinNT)
- Windows system directory (e.g., \Windows\System or WinNT\System32)
- One of the directories specified in the PATH statement

The TE3200ps.dll file may be distributed with the TE3200 power supply and any associated applications.

5.4 Constant Declarations

The following constants may be useful when developing application programs for the TE3200.

```
Define EPROM_OK = $AA55;              // EPROM good value
//============================================================
  // EPROM Address Locations stored scaled to 16 bit integers
//============================================================
define EE_ZERO = $0;
define EE_MODEL = 1;                  // Board model number 9 character length
define EE_SNUMBER = $0B;              // Serial number
define EE_REVISION = $0C;             // Hardware revision
define EE_MFG_MONTH = $0D;            // Manufacturing month
define EE_MFG_YEAR = $0E;             // Manufacturing year
define EE_CAL_DAY = $1D;
define EE_CAL_MONTH = $1E;
define EE_CAL_YEAR = $1F;
define EE_CHECK = $0F;
//=============================================================
// Control State Constants
//=============================================================
define RELAY_CLOSED = 0;
define RELAY_OPEN = 1;
//=============================================================
// Offsets shown for PSU1.
// PSU 2 offset = PSU1 offset + $10      Scaling              Resolution
//=============================================================
define EE_CAL_VGAIN = $10;               // Scaled x 10000
define EE_CAL_VOFFS = $11;               // Scaled x 1000      1 mV
define EE_CAL_IGAIN = $12                // Scaled x 10000
define EE_CAL_IOFFS = $13                // Scaled x 1000      1 mA
define EE_CAL_VMGAIN = $14               // Scaled x 10000
define EE_CAL_VMOFFS = $15               // Scaled x 1000      1 mV
define EE_CAL_IMGAIN = $16               // Scaled x 10000
define EE_CAL_IMOFFS = $17;              // Scaled x 1000      1 mA
define EE_LIM_VOLT = $18;                // Scaled x 100       10 mV
define EE_LIM_CURR = $19;                // Scaled x 100       10 mA
define EE_LIM_POWER = $1A;               // Scaled x 10        100 mW
define EE_LIM_VGAIN = $1B;               // Scaled x 1000000
define EE_LIM_IGAIN = $1C;               // Scaled x 1000000
 //=============================================================
 // Cal Parameter Constants
 //=============================================================
define CAL_VGAIN = 1;
define CAL_VOFFS = 2;
define CAL_IGAIN = 3;
define CAL_IOFFS = 4;
define CAL_VMGAIN = 5;
define CAL_VMOFFS = 6;
define CAL_IMGAIN = 7;
define CAL_IMOFFS = 8;
```

Programming with C/C++ Tools

The following steps are required to use the TE3200 driver with C/C++ development tools:

- Include the TE3200.H header file in the C/C++ source file that uses the TE3200 function. This header file is used for all driver formats. The file contains function prototypes and constant declarations to be used by the compiler for the application.
- For Windows applications, make sure the DLL is installed in the proper directory (see previous sections beginning on page 21, that describe how to use the DLL).
- Add the required .LIB file to the project. This can be the import library TE320032.LIB for 32-bit Microsoft applications, or TE3200.LIB for 32-bit Borland C++ applications. Windows based applications that explicitly load the DLL by calling the Windows **LoadLibrary** API need not include the .LIB file in the project.
- Add code to call the TE3200 as required by the application.
- Build the project.
- Run, test, and debug the application.

Programming with Visual Basic

The TE320032.BAS file contains function declarations for the TE3200PS.DLL driver. The BAS file must be loaded using **Load File** from the Visual Basic File menu before the functions can be used.

5.5 DLL Functions and Procedures

The following functions and procedures are provided by the DLL:

General System functions

- te32_Close_all
- te32_DLLVersion
- te32_WinDriverVersion
- te32_BoardPresent
- te32_Model_Number
- te32_xWaitFor
- te32_Model_Number
- te32_Reset

Power Supply Control functi ons

- te32_FlashLed
- te32_OutputRelay
- te32_PowerSate
- te32_set_voltage
- te32_set_current

Measurement functions

- te32_Measure_Voltage
- te32_Measure_Current

Status functions

- te32_Status

EPROM functions

- te32_EEProm_Read
- te32_GetConfigValue
- te32_GetCalibration

Details on parameters passed and results returned are covered in the following paragraphs.

Chapter 6 –Functions Reference

6.1 General System Functions

This chapter contains all of the TE3200 driver functions. Each function lists a syntax example followed by a short description of the function, parameters, and type.

Te32_Close_all

procedure te32_Close_all;
This function closes all device handles still open to any TE3200 board. ***This function must be called upon exiting your application program***. If you fail to do so, the board cannot be accessed again as the drivers lock access to the board while board handles are open.
If you are using Visual Basic, place this call in the Form, QueryUnload event handler.
This function has no parameters and returns a void. (no function result.)

te32_DLLVersion

function te32_DLLVersion(iBrdID:integer; Var strRevision:Pchar):integer;
This function returns a null terminated string containing the version of the TE3200ps.dll. This version string is passed by reference and is the second parameter in the function list. The board ID must be passed by value as the first parameter. The function result returns a 32-bit integer (int32), which contains the DLL version multiplied by 100. Thus, version 1.00 would be returned at 100.
Future versions of the DLL may contain additional functions. By having the ability to determine the DLL version, your application program can be designed to handle new features when available.

Te32_WinDriverVersion

function te32_WinDriverVersion(iBrdID:integer; Var strRevision:Pchar):integer;
This function returns a null terminated string containing the version of the PCI card device driver. This version string is passed by reference and is the second parameter in the function list. The board ID must be passed by value as the first parameter. The function result returns a 32-bit integer (int32), which contains the WinDriver version multiplied by 100. Thus, version 1.00 would be returned at 100.

Te32_BoardPresent

function te32_BoardPresent(iBrdID:integer):integer;
This function returns 0 if the board with the board ID passed as the parameter is not present. If the board is present, a “-1” is returned instead. The function type is a 32-bit integer (int32).
This function must be used any time a new device is being accessed to make sure a handle to the board is available. If a board is not present and access to that board is attempted using any of the other DLL functions (except the te32_DLLVersion call), a run-time error will occur.

Te32_Model_Number

procedure te32_Model_Number(iBrdID:integer):integer;
This function returns the model number for the board ID specified as the first and only parameter. The board model for a TE3200 is 3200 hex or 12800 decimal.

Te32_xWaitFor

procedure te32_xWaitFor(period:integer);
This procedure generates a time delay specified in milliseconds. The resolution and accuracy of this function is about 1 millisecond for times under 2 seconds. Times above 2 seconds (2000 msec) will result in relinquishing control to other windows applications, which may affect accuracy. The time to delay for is specified as a 32-bit integer (int32). The function result for this call is void.

Te32_Reset

procedure te32_Reset(iBrdID:integer);
This procedure generates a time delay specified in milliseconds. The resolution and accuracy of this function is about 1 millisecond for times under 2 seconds. Times above 2 seconds (2000 msec) will result in relinquishing control to other windows applications, which may affect accuracy. The time delay period is specified as a 32-bit integer (int32). The function result for this call is void.

6.2 Control Functions

te32_FlashLed

procedure te32_FlashLed(iBrdID:integer; iIndex:integer);
This function causes one of the LED's on the PSU's front panel to flash momentarily. The iIndex parameter (Int32) determines which of the two output module's LED's will flash.
This function has no parameters and returns a void. (no function result.)

te32_OutputRelay

procedure te32_OutputRelay(iBrdID:integer; iIndex:integer; iValue:integer);
This function controls the state of the output relay and PWM power for the selected output module (iIndex). The iValue passed determines whether the output relay is opened or closed. The state of the output relay and TE3200 power is returned as part of the Status query (see te32_Status, paragraph 6.4).
Note: To prevent damage to the output relay, the TE3200 power is disengaged before the output relay is opened and re-engaged before the output relay is closed. This process requires approximately 250 msec to complete.
Valid values for iValue are:

RELAY_CLOSED = 0;
RELAY_OPEN = 1;

te32_ PowerState

procedure te32_PowerState(iBrdID:integer; iIndex:integer; iValue:integer);
This function controls the state of the PWM power for the selected PSU output (iIndex). The iValue passed determines whether the PWM power relay (48 Volt Relay State) is applied or not. The state of the PWM power is returned as part of the Status query (48 Volt Relay State), see paragraph 0.
Note: This process requires approximately 50 msec to complete to allow for relay de-bounce.
Valid values for iValue are:

RELAY_CLOSED = 0;
RELAY_OPEN = 1;

te32_Set_voltage

function te32_Set_Voltage(iBrdID:integer; iIndex:integer; sVoltage:double):integer;
This function programs the output voltage in VDC for the selected output channel (1 or 2). The desired output voltage is passed as a double precision floating point number.
If a problem occurs during this query, the function result may be one of the following error values:

<u>Error Codes:</u>

-200	// V set below lower limit
-201	// V set above upper limit
-202	// V set requested exceeds total power available. Note that V * I must be < Max. Power.
-203	// V set exceeds DAC upper limit

te32_Get_voltage

function te32_Get_Voltage(iBrdID:integer; iIndex:integer; var sVoltage:double):integer;
This function returns the programmed voltage value for the requested output module number. The programmed value is passed by reference.
If a problem occurs during this query, the function result may be one of the following error values:

<u>Error Codes:</u>

-220	// // Output module number out of range
-230	// Board not present

te32_Set_current

function te32_Set_Voltage(iBrdID:integer; iIndex:integer; sVoltage:double):integer;
This function programs the output current limit in ADC for the selected output channel (1 or 2). The desired current limit is passed as a double precision floating point number.
If a problem occurs during this query, the function result may be one of the following error values:

<u>Error Codes:</u>

-210	// I set below lower limit
-211	// I set above upper limit
-202	// V set requested exceeds total power available. Note that V * I must be < Max. Power
-213	// I set exceeds DAC upper limit

te32_Get_current

function te32_Get_Current(iBrdID:integer; iIndex:integer; var sCurrent:double):integer;
This function returns the programmed current limit value for the requested output module number. The programmed value is passed by reference.
If a problem occurs during this query, the function result may be one of the following error values:

<u>Error Codes:</u>

-220	// // Output module number out of range
-230	// Board not present

6.3 Measurement Functions

te32_Measure_Voltage

function te32_Measure_Voltage(iBrdID:integer; iIndex:integer; iAverage:integer; var sVoltage:double):integer;

This function call measures the output voltage for the select power supply. If no measurement data is available, the function result will be non-zero. The iAverage parameter can range from 1 to 255 and sets the number of measurements to be averaged before returning the average measurement result.

If a problem occurs during this query, the function result may be one of the following error values:

Error Codes:

-240	// Measurement Average requested less than 1

te32_Measure_Current

function te32_Measure_Current(iBrdID:integer; iIndex:integer; iAverage:integer; var sCurrent:double):integer;

This function call measures the output current for the select power supply. If no measurement data is available, the function result will be non-zero. The iAverage parameter can range from 1 to 255 and sets the number of measurements to be averaged before returning the average measurement result.

If a problem occurs during this query, the function result may be one of the following error values:

Error Codes:

-240	// Measurement Average requested less than 1

6.4 Status Functions

te32_Status

function te32_Status(iBrdID:integer; iIndex:integer):integer;
This function call returns the status for the requested output module number, iIndex (1 or 2). The status bits can be decoded using the table below:

Bit number	**Weight**	**Description**	
b31-b5		Reserved	
b4	16	48 Volt Relay State:	0 = Closed, 1 = open
b3	8	Output Relay State:	0 = Closed, 1 = open
b2	4	Fault	0 = Good, 1 = Fault
b1	2	48 Volt	0 = 48 V OK, 1 = 48 V Bad
b0	1	Mode	0 = CC mode, 1 = CV mode

6.5 Calibration and Configuration Functions (EPROM access)

Every TE3200 contains non-volatile memory used to store the configuration and calibration values for each output module. The following two functions may be used to query EPROM configuration and/or calibration data.

Te32_EEProm_Read

function te32_EEProm_Read(iBrdID:integer; iAddress:integer; var iData:integer):integer;
This function may be used to read from any EPROM location. The EPROM address offsets for relevant data fields are shown in the Constant declaration section. Note that while the data type of the variable returned (iData) is a 32-bit integer, each EPROM location only stores 16 bits so the results should be interpreted as a signed short integer or character depending on the location read. For configuration value (limits), use the te_GetConfigValue function instead as it returns a floating-point number. For calibration coefficients, use the te_GetCalibration function instead as it returns a floating-point number.
If a problem occurs during this function call, the function result may be one of the following error values:

Error Codes:

-250 // EPROM read error

te32_GetConfigValue

function te32_GetConfigValue(iBrdID, iIndex, iParam:integer; var dbValue:double):integer;
This function may query a specific configuration (hardware limit) value for the specified board and output module number (iIndex). The configuration value returned depends on the iParam value passed. The following values are valid for iParam:

Parameter value	**Limit Value Returned**
1	Max Voltage
2	Max Current
3	Max. Power

If a problem occurs during this function call, the function result may be one of the following error values:

Error Codes:

-220 // Output module number out of range

-222 // Parameter value out of range

te32_GetCalibration

te32_GetCalibration(iBrdID, iIndex, iParam:integer; var dbCalData:double):integer;
This function returns the selected Cal coefficient for the specified board and output module (iIndex).in the dbCalData value. The cal coefficient value returned is determined by the value of the iParam parameter per the table shown below.

Parameter value	**Calibration Coefficient Value Returned**	
1	CAL_VGAIN.	Voltage output gain
2	CAL_VOFFS	Voltage output offset
3	CAL_IGAIN	Current output gain
4	CAL_IOFFS	Current output offset
5	CAL_VMGAIN	Voltage measurement gain
6	CAL_VMOFFS	Voltage measurement offset
7	CAL_IMGAIN	Current measurement gain
8	CAL_IMOFFS	Current measurement offset

If a problem occurs during this function call, the function result may be one of the following error values:

Error Codes:

-220	// Output module number out of range
-222	// Cal parameter requested out of range

6.6 DLL Error Codes

All DLL function calls return 32-bit integer error results. If no error occurs during the call, the error result will be zero. Any negative result indicates an error. Note that procedure calls have a VOID result and return no error codes.

Below is a list of all error codes by category and their descriptions. The application programmer is responsible for error checking.

Error Code	Description
Device Driver Errors	
-100	PXI board handler error
Range Errors – Voltage	
-200	V set below lower limit. Requested voltage setting is less than zero.
-201	V set above upper limit. Requested voltage setting is exceeds maximum voltage of output module.
-202	V set exceeds power limit. Requested voltage setting would result in power above limit. Try lowering current limit first.
-203	V DAC value out of range. Programmed value exceeds hardware limit.
Range Errors –	
-210	I set below lower limit. Requested current setting is less than zero.
-211	I set above upper limit. Requested current setting is exceeds maximum current limit.
-212	I set exceeds power limit. Requested current setting would result in power above the limit. Try lowering voltage first.
-213	I DAC value out of range. Programmed value exceeds hardware limit.
Range Errors – Other	
-220	Output module number out of range. Index range is 1 to 2.
-221	Cal parameter requested out of range. Unknown calibration parameter index specified.
-222	Limit parameter requested out of range. Unknown limit parameter index specified.

Error Code	Description
Misc. Errors	
-230	Board not present. No board found at board index location.
-240	Measurement average requested less than 1. Minimum value allowed is 1.
EPROM Errors	
-250	EPROM read error. Could not read from EPROM at requested address.
-251	EPROM write error. EPROM is write-protected.

6.7 Visual Basic Applications

The TE3200ps.dll may be called from a Visual Basic 6 application program. The required function declarations are provided in “DLL Declaration Module”, below. A sample VB6 application GUI (TE3200_Sample1.vbp) is included on the CD (D:\SampleCode\ Te32VB6_Sample.zip).

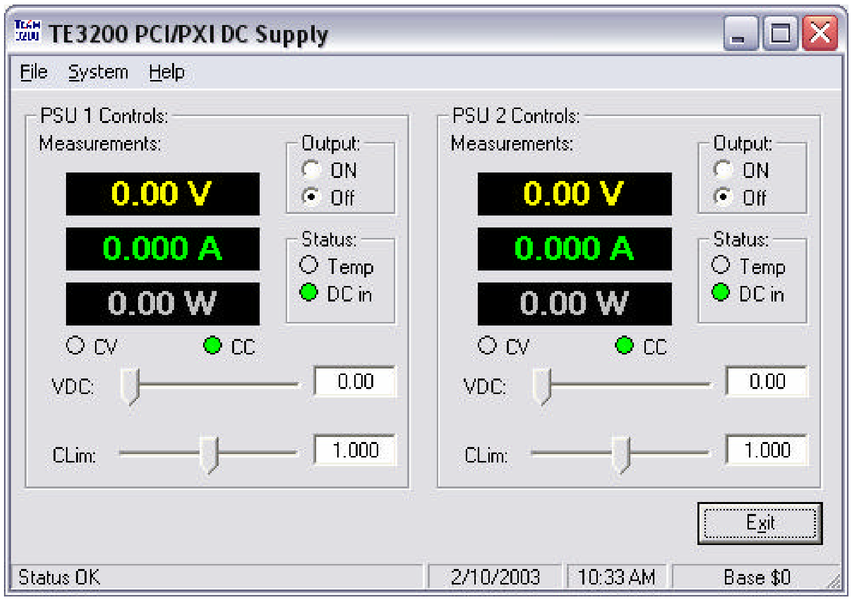


Figure 1: TE3200_Sample1 VB6 Project

Programming tips

When developing a VB Application, make sure to check for the presence of each of up to four boards using the te32_BoardPresent function call.

```
‘Initialize all boards
Dim iBrdID As Long
  For iBrdID = 0 To MAX_BOARDS
    If te32_BoardPresent(iBrdID) = BOARD_PRESENT Then
      te32_Initialize_DDS iBrdID
    End If
  Next iBrdID
```

Make sure to put the te32_Close_all procedure call in the QueryUnload event handler of your main application form.

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
‘==============================================================================
‘Close Device handles to TE3200 Board(s)
‘==============================================================================
  te32_Close_all
End Sub
```

DLL Declaration Module

```
Attribute VB_Name = “TE3200_DLL_Declarations”
‘==============================================================================
‘ Module:        TE3200 DLL declarations
‘ Copyright:     © 2002 Team Solutions, Inc.
‘ Date:          02/19/2002
‘ Revised:       02/19/2002
```

```
'==============================================================================
'String Passing:
'===============
'String arguments should always be passed using the ByVal keyword to ensure
'they are passed as null terminated strings. Otherwise a string descriptor
'is passed which the te3200ps.dll does not know how to handle as it does
'not know VB.
'Integers
'========
'Pass integers by value using the ByVal keyword
'Arrays
'======
'Arrays cannot be passed using the ByVal keyword and should always be passed by
reference.
'==============================================================================
'te32_Close_all
Declare Sub te32_Close_all Lib "te3200ps.dll" ()
'te32_DLLVersion
Declare Function te32_DLLVersion Lib "te3200ps.dll" (ByVal BrdID As Long, ByRef strRevision As
String) As Long
'te32_WinDriverVersion
Declare Function te32_WinDriverVersion Lib "te3200ps.dll" (ByVal BrdID As Long, ByRef
strRevision As String) As Long
'te32_BoardPresent
Declare Function te32_BoardPresent Lib "te3200ps.dll" (ByVal iBrdID As Long) As Long
'te32_xWaitFor
Declare Sub te32_xWaitFor Lib "te3200ps.dll" (ByVal period As Long)
'te32_Model_Number
Declare Function te32_Model_Number Lib "te3200ps.dll" (ByVal iBrdID As Long) As Long
'te32_Reset
Declare Sub te32_Reset Lib "te3200ps.dll" (ByVal iBrdID As Long)
'PSU Programming Functions
'te32_Output_Power
Declare Sub te32_OutputRelay Lib "te3200ps.dll" (ByVal iBrdID As Long, ByVal iIndex As Long,
ByVal iValue As Long)

Declare Sub te32_FlashLed Lib "te3200ps.dll" (ByVal iBrdID As Long, ByVal iIndex As Long)
Declare Function te32_Set_Voltage Lib "te3200ps.dll" (ByVal iBrdID As Long, ByVal iIndex
As Long, _
                                                      ByVal sVoltage As Double) As Long
Declare Function te32_Set_Current Lib "te3200ps.dll" (ByVal iBrdID As Long, ByVal iIndex
As Long, _
                                                      ByVal sCurrent As Double) As Long
Declare Function te32_Get_Voltage Lib "te3200ps.dll" (ByVal iBrdID As Long, ByVal iIndex
As Long, _
                                                      ByRef sVoltage As Double) As Long
Declare Function te32_Get_Current Lib "te3200ps.dll" (ByVal iBrdID As Long, ByVal iIndex
As Long, _
                                                      ByRef sCurrent As Double) As Long
Declare Function te32_Measure_Voltage Lib "te3200ps.dll" (ByVal iBrdID As Long, ByVal
iIndex As Long, _
                                                          ByVal iAverage As Long, ByRef
sVoltage As Double) As Long
Declare Function te32_Measure_Current Lib "te3200ps.dll" (ByVal iBrdID As Long, ByVal
iIndex As Long, _
                                                          ByVal iAverage As Long, ByRef
sCurrent As Double) As Long
Declare Function te32_Status Lib "te3200ps.dll" (ByVal iBrdID As Long, ByVal iIndex As
Long) As Long
'PSU Eprom Storage Routines
Declare Function te32_EEProm_Read Lib "te3200ps.dll" (ByVal iBrdID As Long, ByVal
iAddress As Long, ByRef iData As Long) As Long
Declare Function te32_GetConfigValue Lib "te3200ps.dll" (ByVal iBrdID As Long, ByVal
iIndex As Long, _
                                                         ByVal iParam As Long, ByRef
dbValue As Double) As Long
'Calibration Routines
Declare Function te32_GetCalibration Lib "te3200ps.dll" (ByVal iBrdID As Long, ByVal
iIndex As Long, _
                                                  ByVal iParam As Long, ByRef dbCalData As
Double) As Long
```

Constant Declaration Module

```
Attribute VB_Name = “TE3200_Constants”
‘================================================================
‘ TE3200 CONSTANTS
‘================================================================
‘ Module:       TE3200 Constant declarations
‘ Copyright:    © 2002 Team Solutions, Inc.
‘ Date:         02/19/2002
‘ Revised:      02/19/2002
‘================================================================
  ‘TE3200 GUI specific constants
  Global Const PROG_VERSION = “1.0 – 02/19/2002”     ‘Program Version
  Global Const PROG_NAME = “TE3200 Dual DC Power Supply”
  Global Const MAX_BOARDS = 3
  Global Const BOARD_PRESENT = -1

  ‘Operating modes
  Global Const MODE_DUAL = 0
  Global Const MODE_PARALLEL = 1
  Global Const MODE_SERIES = 2
```

Appendix A – Calibration Tolerances

Calibration Tolerances

CAL FACTOR	DESCRIPTION	TOLERANCE
VOFS	Voltage calibration offset	0.1% of FS
VGAIN	Voltage calibration gain	
IOFS	Current calibration offset	0.1% of FS
IGAIN	Current calibration gain	
VRBOFS	Voltage read-back calibration offset	0.1% of FS
VRBGAIN	Voltage read-back calibration gain	
IRBOFS	Current read-back calibration offset	0.1% of FS
IRBGAIN	Current read-back calibration gain	
VTRPOFS	Over-voltage trip calibration offset	1% of FS
VTRPGAIN	Over-voltage trip calibration gain	
ITRPOFS	Over-current trip calibration offset	1% of FS
ITRPGAIN	Over-current trip calibration gain	

Appendix B – Specifications

General

Type	Programmable output
Voltage	0-30V
Power	30W
Max Current	2A
Programming Accuracy	14mV
Load Regulation	150mV for full load

Outputs

Modes of Operation:	Constant Voltage / Constant Current
Current Regulation:	Current Limit
Programming Resolution:	12 bits
Read-back Resolution:	12 bits
Remote/Local Sense:	Up to 2 VDC may be dropped across sense loads. This drop reduces the voltage available at the load.

Input Power

Configuration:	2 separate isolated DC Sources
Power:	48VDC/1.25A/60W
Efficiency:	>80% (depends on programmed output voltage and current)

Protection Circuitry

Over Voltage:	100% of full scale.
Current Limit:	Programmable from 0 to FS. Output will automatically switch to constant current mode when limit is reached.
Current Trip:	100% of full scale (fixed).
Over Temperature:	Automatically disables output if maximum allowable temperature is exceeded. (80°C)
UUT Discharge:	Active when output is down-programmed. Circuit will discharge at a constant rate.
Short Circuit:	Outputs are protected in the event of a short across output terminals.
Isolation Relays:	Independently controllable for each output. Circuitry is provided to protect relays from current switching.
Remote Inhibit:	Provides for more remote hardware shutdown of outputs via front panel interlock pins.

Physical

Format:	PXI single-slot
Size:	Single-width 3U PXI standard
Approximate Weight:	12 oz.
Operating Temperature:	0 to +40 °C
Non-operating temperature:	-40 °C to +85 °C
Humidity:	20% - 95% non-condensing
Cooling:	PXI/Compact PCI Chassis airflow