INSTRUCTION MANUAL

CSI Software Development Kit Beginner's Guide

Revision: 8/06

CSI Software Development Kit Beginner's Guide Table of Contents

PDF viewers note: These page numbers refer to the printed version of this document. Use the Adobe Acrobat® bookmarks tab for links to specific sections.

Section 1. Introduction

1.1 Background

Campbell Scientific dataloggers process measurements made with a wide variety of sensors and store summaries and statistics of these measurements as data. Dataloggers carry out these operations based on instructions in a datalogger program. Users create these datalogger programs with CSI software such as Short Cut, Edlog and CRBasic Editor. Programs can be sent to dataloggers via a variety of communication channels (e.g. phone modem or RF).

A network of dataloggers may contain anywhere from one to several hundred dataloggers, each with its own set of sensors, communication links, and schedule for making measurements. Management of datalogger networks is an intricate task. Campbell Scientific software such as LoggerNet handles the task of network management by:

- sending user-created programs to a datalogger
- checking computer and datalogger times and synchronizing if necessary
- managing the schedule of data collection from each datalogger
- storing values in the LoggerNet server data cache in tables that correspond to a given datalogger and datalogger program
- keeping log files for all communications, errors, etc.

Campbell Scientific customers have a wide variety of measurement interests—agriculture, weather, water resources, and vehicle testing, to name a few. Users often need applications that can be customized to suit their needs. The LoggerNet software product offers a number of general purpose clients that work with the LoggerNet server to perform the above tasks. However, there are times when users may need to write their own custom applications. Therefore, CSI offers different Software Development Kits that help developers create applications that extend or replace standard Campbell Scientific software products.

1.2 Advantages of Using an SDK

When developers need to write custom applications, SDK controls play an intermediary role (Figure 1-1) in performing desired tasks, such as sending a program to a specific datalogger or retrieving data from a datalogger. One can, in principle, achieve the same results without SDK controls by directly interacting with the LoggerNet server or the dataloggers through a “messaging” process, but this can be a very complex process. SDK controls greatly simplify the communication process and provide a layer of insulation from future changes of the messaging protocol to the datalogger.

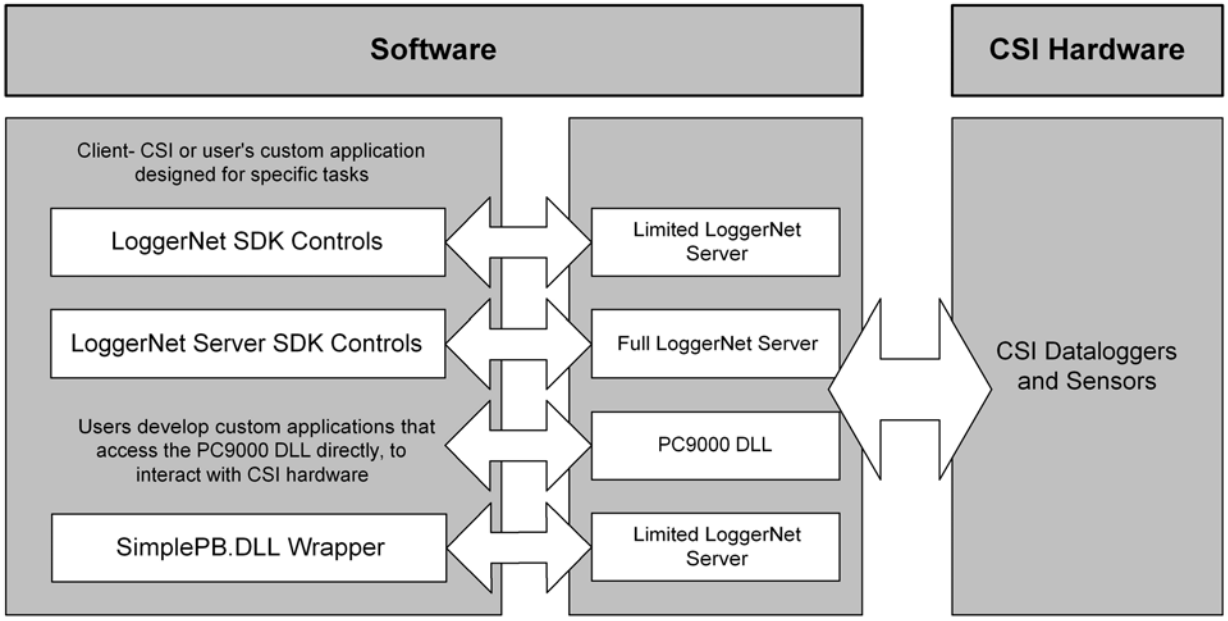


Figure 1-1

SDK products simplify the task of application development tremendously by encapsulating complexities inherent in client-server communications.

1.3 Overview of Current SDK Products

Campbell Scientific offers several different SDK products. Each SDK product provides different advantages depending on the specific needs of the developer.

1.3.1 Current SDK Products

1.3.1.1 LoggerNet SDK

This software development kit contains six ActiveX controls and a limited version of the LoggerNet server DLL. A developer can use this SDK to create applications that remotely access an existing LoggerNet server installation; however, the included LoggerNet server direct DLL (CORALIB3D.DLL) can only be loaded and accessed locally.

Installation of this SDK includes source code of examples written in Delphi 6.0, Visual Basic 6.0, and Visual C++ 8.0 MFC that will start the CORALIB3D.DLL and use the other SDK controls. The install also includes documentation for developing applications using the methods and functions of the included SDK controls. Please note that the CORALIB3D.DLL included in this SDK will only communicate with a single datalogger through a direct connection using RS-232 or an IP Port and will not allow remote client connections.

1.3.1.2 LoggerNet Server SDK

This software development kit contains six ActiveX controls and the LoggerNet server DLL (CORALIB3.DLL). A developer can use this SDK to create applications that can remotely connect to an existing LoggerNet installation or the included LoggerNet server DLL.

Installation of this SDK includes source code of examples in Delphi 6.0, Visual Basic 6.0, and Visual C++ 8.0 MFC that start the LoggerNet server and use the included SDK controls. The install also includes documentation for developing applications that can load and manage the LoggerNet server, edit the network map, and get data via the LoggerNet server. The included LoggerNet server DLL allows remote client connections and includes everything necessary to communicate with dataloggers independent of any other LoggerNet installation.

1.3.1.3 PC9000 SDK

This software development kit contains the PC9000 communication DLL and drivers needed to facilitate connections to CR9000(X) and CR5000 dataloggers. A developer can use this SDK to quickly create applications that will only need to communicate one at a time to either a CR9000(X) or a CR5000 datalogger.

Installation of this SDK includes source code of an example in Visual Basic 6.0 of how to build an executable that can load and manage the PC9000 DLL. Also included in the installation is documentation for developing applications using the SDK.

1.3.1.4 BMP5 Direct SDK

This software development kit is for developers that want a simple way to create an application that communicates with any PakBus datalogger using RS-232 or an intermediate communication that is transparent to RS-232 such as RF400 radios. The application created using the BMP5 Direct SDK will only be able to communicate with one datalogger at a time.

Starting with version 2.0, the BMP5 Direct SDK provides the developer access to a communications engine, CORALIB3D.DLL, through the SimplePB.DLL wrapper. The BMP5 Direct SDK install includes examples with source code along with documentation describing how to use the SimplePB.DLL.

Previous versions of the BMP5 Direct SDK used the PakBusDLL.DLL as the communication engine. The SimplePB.DLL commands remain the same. Only the communication engine used by the SimplePB.DLL has changed in version 2.0 of this SDK.

1.3.2 SDK Comparison Table

SDK Name	Installed Items	General Purpose	Advantages
LoggerNet SDK	• Limited LoggerNet Server (coralib3d.dll) • Six ActiveX Control DLLs • Examples • Help File • Documentation	Although custom software can be written to access an existing LoggerNet server remotely, the included communications server can only be loaded and accessed locally. Since this is a limited version of the LoggerNet server, only direct communication using RS-232 or an IP Port is allowed. Additionally, scheduled collection and connections from remote clients are not allowed.	If you are creating a basic application that only requires simple direct communication using RS-232 or an IP Port or if you merely want to create an application that extends the capabilities of an existing LoggerNet software installation, this SDK is less expensive to purchase.
LoggerNet Server SDK	• Full LoggerNet Server (coralib3.dll) • Six ActiveX Control DLLs • Examples • Help File • Documentation	Used to develop applications that can load and manage the LoggerNet server, edit the network map, and get data via the LoggerNet server. The included LoggerNet server DLL allows remote client connections and includes everything necessary to communicate with dataloggers and be independent of any other LoggerNet installation.	The most robust SDK available with the LoggerNet server. Developer's that want unlimited application control, operations, and telecommunications to the datalogger network will want this package.
PC9000 SDK	• PC9000 DLL • Windows Drivers • Examples • Documentation	Used to create applications to communicate directly to CR9000 or CR5000 dataloggers only. This SDK only allows communication with one datalogger at a time and does not rely on the LoggerNet Server nor does it support PakBus communication.	A simple solution for developers that are focused on a very specific API application that connects to CR9000 and CR5000 dataloggers only.
BMP5 Direct SDK	• SimplePB.DLL • Examples • Documentation	Used to develop applications that can communicate with any PakBus datalogger via RS-232 or intermediate communications transparent to RS-232 such as RF400 radios. This SDK can only communicate with one datalogger at a time.	Developers creating applications that only communicate with a single PakBus datalogger may prefer this SDK because it is less complex than the LoggerNet SDK.

Section 2. LoggerNet SDK

2.1 LoggerNet Overview

LoggerNet is developed around client-server architecture. LoggerNet's client-server technology is based on a server that communicates with a network of dataloggers via various communications technologies. The server listens for client requests, accepts the requests, and acknowledges to the client that a request has been received. The server fulfils this request and returns information to the client.

Often a client makes several requests. The server, however, processes only one request at time and in the order it was received. Once a client submits a request to the server, the client is free to do something else, knowing that its request will be processed. In other words, client requests may not get immediate response from the server nor do the clients have to stop doing something else while waiting for an answer. This is called asynchronous communication.

2.2 LoggerNet SDK Overview

This software development kit contains six ActiveX controls and a limited version of the LoggerNet server DLL (CORALIB3D.DLL). A developer can use this SDK to create applications that remotely access an existing LoggerNet Server; however, the included LoggerNet server direct DLL can only be loaded and accessed locally.

Installation of this SDK includes source code of examples written in Delphi 6.0, Visual Basic 6.0, and Visual C++ 8.0 MFC that will start the CORALIB3D.DLL and use the other SDK controls. The install also includes documentation for developing applications using the methods and functions of the included SDK controls. Please note that the CORALIB3D.DLL can only communicate with a datalogger using a direct RS-232 or an IP Port connection and will not accept remote client connections.

2.2.1 LoggerNet SDK Controls

The LoggerNet SDK controls simplify the task of application development tremendously by encapsulating complexities inherent in client-server communications. A summary of these controls and their uses is shown in Table 2-1.

TABLE 2-1. LoggerNet SDK Controls and Uses	
LoggerNet SDK Control	**Uses**
CsiBrokerMap	Display names of dataloggers in the LoggerNet server's network map Display names of tables and columns of data from dataloggers
CsiDataLogger	Establish a connection to a datalogger via the LoggerNet server Send/Receive datalogger programs Check datalogger time, synchronize time with PC Retrieve data from a connected datalogger
CsiDataSource	Monitor data collected from a datalogger
CsiCoraScript	Execute CoraScript commands on the LoggerNet server
CsiLogMonitor	Monitor LoggerNet server transaction and communication log files.
CsiServerDirect	Start and Stop the CORALIB3D.DLL

2.3 Software Requirements

2.3.1 Required Campbell Scientific, Inc. Software

The core CSI product necessary for applications developed using the LoggerNet SDK is the LoggerNet server DLL. This SDK includes a limited version of this DLL as the LoggerNet server direct DLL (CORALIB3D.DLL). An application can also be created with the included SDK controls to use an existing installation of LoggerNet software. CSI software products that provide the necessary server DLL in an existing software installation include:

- LoggerNet version 1.1 or higher
- PC400 version 1.0 or higher
- Visual Weather 1.0 or higher

NOTE Only the products mentioned above can be used with the SDK. The SDK controls cannot communicate through PC208W for example.

2.3.2 Development Tool Requirements

The SDK ActiveX controls were developed in Visual C++ 8.0 MFC environment. They are best suited for the Visual C++ 8.0 MFC and Visual Basic 6.0 environments. They can also be used in a Delphi (Object Pascal) environment. Examples are included for the Visual Basic 6.0, Visual C++ 8.0 MFC and Delphi 6.0 development environments.

2.4 How to Install SDK Controls on a Computer

SDK controls are ActiveX DLLs (Dynamically Linked Libraries). DLLs contain code written in one of several computer languages (e.g., Visual C++ MFC in this case). DLLs are dynamically linked, meaning they are stored separately from the main application and accessed by the main application whenever needed.

2.4.1 How to Register SDK Controls on a PC

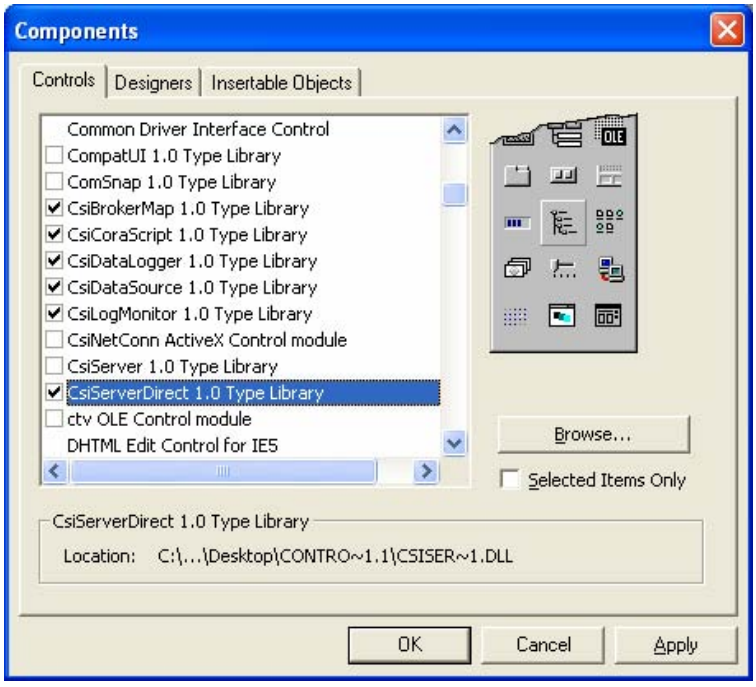
You will find six DLLs in the SDK: CsiBrokerMap.dll, CsiDatalogger.dll, CsiDataSource.dll, CsiCoraScript.dll, CsiLogMonitor.dll, and CsiServerDirect.dll, each corresponding to the SDK control bearing the same name as listed in Table 2-1. The SDK also includes a LoggerNet server direct DLL, coralib3d.dll, which doesn't need to be registered but needs to reside in the directory of the created application, within the declared PATH environmental variable, or in the Windows system directory.

2.4.2 How to Access SDK Controls in Visual Basic 6.0

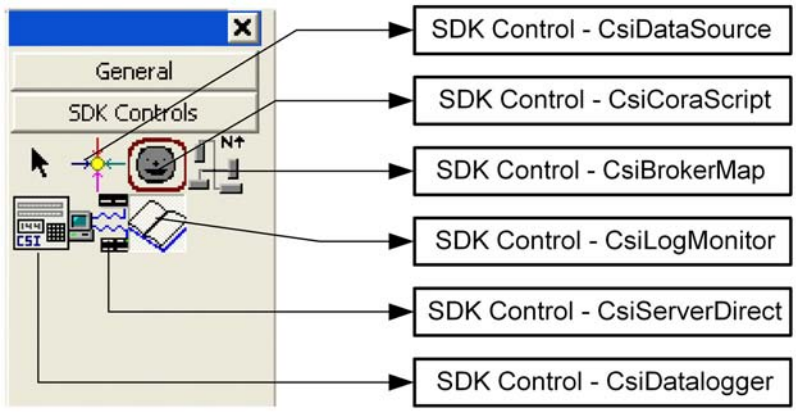
1. Start the Visual Basic environment (Windows | Start | Microsoft Visual Basic 6.0 | Microsoft Visual Basic 6.0).

2. File | New Project | Standard.EXE → Opens a form.

3. VB Main Menu | Project | Components. Select the CSI SDK controls as shown and click OK.



4. VB Main Menu | View | Toolbox → SDK control icons appear.



You are now ready to start developing a client application using the SDK controls.

Section 3. LoggerNet Server SDK

3.1 LoggerNet Overview

LoggerNet is developed around client-server architecture. LoggerNet's client-server technology is based on a server that communicates with a network of dataloggers via various communications technologies. The server listens for client requests, accepts the requests, and acknowledges to the client that a request has been received. The server fulfils this request and returns information to the client.

Often a client makes several requests. The server, however, processes only one request at time and in the order it was received. Once a client submits a request to the server, the client is free to do something else, knowing that its request will be processed. In other words, client requests may not get immediate response from the server nor do the clients have to stop doing something else while waiting for an answer. This is called asynchronous communication.

3.2 LoggerNet Server SDK Overview

This software development kit contains six ActiveX controls and the LoggerNet server DLL (CORALIB3.DLL). A developer can use this SDK to create applications that remotely access an existing LoggerNet software installation or the included LoggerNet server DLL.

Installation of this SDK includes source code of examples in Delphi 6.0, Visual Basic 6.0, and Visual C++ 8.0 MFC that start the LoggerNet server and use the included SDK controls. The install also includes documentation for developing applications that can load and manage the LoggerNet server, edit the network map, and get data via the LoggerNet server. The included LoggerNet server DLL allows remote client connections and includes everything necessary to communicate with dataloggers and be independent of any other LoggerNet software installation.

3.2.1. LoggerNet Server SDK Controls

The LoggerNet Server SDK controls simplify the task of application development by encapsulating complexities inherent in the client-server communication protocols. A summary of these controls and their uses is shown in Table 3-1.

TABLE 3-1. LoggerNet Server SDK Controls and Uses	
LoggerNet Server SDK Control Name	**Uses**
CsiBrokerMap	Display names of dataloggers in the LoggerNet server's network map Display names of tables and columns of data from dataloggers
CsiDataLogger	Establish connections to dataloggers via the LoggerNet server Send/Receive datalogger programs Check datalogger time, synchronize time with PC Retrieve data from a connected datalogger
CsiDataSource	Monitor data collected from a datalogger
CsiCoraScript	Execute CoraScript commands on the LoggerNet server
CsiLogMonitor	Monitor LoggerNet server transaction and communication log messages
CsiServer	Start and Stop the CORALIB3.DLL

3.3 Software Requirements

3.3.1 Required Campbell Scientific, Inc. Software

The core CSI product necessary for developing applications with the LoggerNet SDK is the LoggerNet server DLL. The LoggerNet server DLL is included with the LoggerNet Server SDK. An application can also be created to use an existing installation of the LoggerNet software. CSI software products that contain the necessary DLL in an existing installation and can interact with the LoggerNet Server SDK controls include:

- LoggerNet version 1.1 or higher
- PC400 version 1.0 or higher
- Visual Weather 1.0 or higher

NOTE Only the products mentioned above can be used with the SDK. The SDK controls cannot communicate through PC208W for example.

3.3.2 Development Tool Requirements

The SDK ActiveX controls were developed in Visual C++ 8.0 MFC environment. They are best suited for the Visual C++ 8.0 MFC and Visual Basic 6.0 environments. They can also be used in a Delphi (Object Pascal)

environment. Examples are included for the Visual Basic 6.0, Visual C++ 8.0 MFC and Delphi 6.0 development environments.

3.4 How to Install SDK Controls on a Computer

SDK controls are ActiveX DLLs (Dynamically Linked Libraries). DLLs contain code written in one of several computer languages (e.g., Visual C++ MFC in this case). DLLs are dynamically linked, meaning they are stored separately from the main application and accessed by the main application whenever needed.

3.4.1 How to Register SDK Controls on a PC

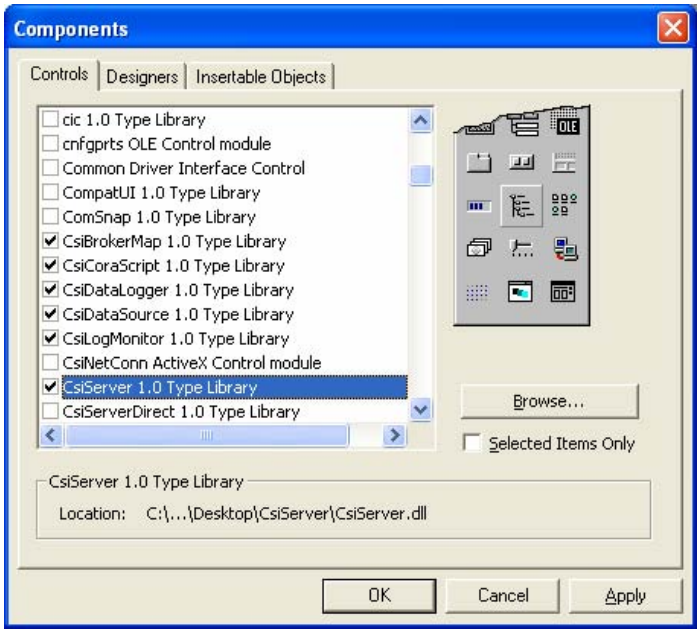
You will find six DLLs in the SDK: CsiBrokerMap.dll, CsiDatalogger.dll, CsiIDataSource.dll, CsiCoraScript.dll, CsiLogMonitor.dll, and CsiServer.dll, each corresponding to the SDK control bearing the same name as listed in Table 3-1. The SDK also includes the LoggerNet server DLL, CORALIB3.DLL, which doesn't need to be registered but needs to reside in the folder containing your created application, the declared PATH environmental variable, or in the Windows system folder.

3.4.2 How to Access SDK Controls in Visual Basic 6.0

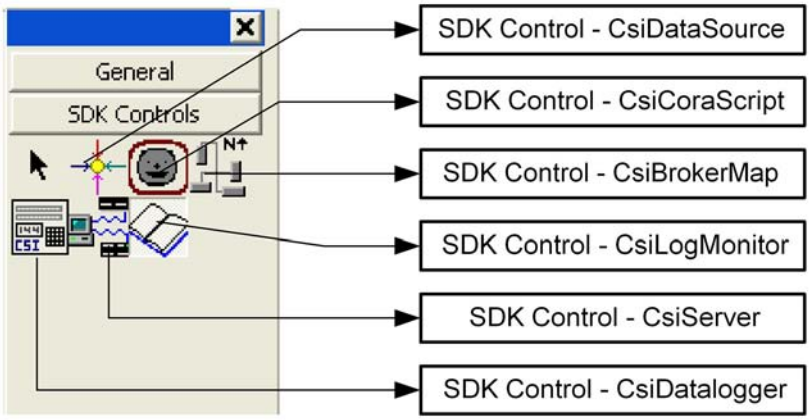
1. Start the Visual Basic environment (Windows | Start | Microsoft Visual Basic 6.0 | Microsoft Visual Basic 6.0).

2. File | New Project | Standard.EXE → Opens a form.

3. VB Main Menu | Project | Components. Select the CSI SDK controls as shown and click OK.



4. VB Main Menu | View | Toolbox → SDK control icons appear.



You are now ready to start developing a client application using the SDK controls.

Section 4. PC9000 SDK

4.1 PC9000 SDK Overview

The PC9000 SDK allows software developers to create custom datalogger application programs using such programming languages as Visual Basic or Delphi. The PC9000 SDK includes the latest PC9000 DLL that provides a full and varied set of datalogger control and data collection functions.

The PC9000 SDK is tailored toward direct, high-speed, real-time communications between the PC and a single datalogger. The SDK currently supports communications with CR9000, CR9000X, and CR5000 dataloggers only.

4.1.1 PC9000 SDK Configurations

The included PC9000.DLL is a standard call-level DLL meaning that at this time it does not support OLE automation and does not need to be registered in the system registry. The SDK also includes associated drivers and currently supports the following CSI datalogger configurations:

- CR9000 connection via a BLC100 PC Bus Link Card (TLink) for Windows 95/98/ME only
- CR9000 connection via a PLA100 Parallel Port Link Adapter (LPT)
- CR9000 connection via a TL925 RS-232 Interface (COM)
- CR9000 connection via a NL105 Network Link Interface (NET)
- CR9000X or CR5000 connection via a 9-pin RS-232 serial cable (COM)
- CR5000 connection via a NL100/NL105 Network Link Interface (NET)

4.2 PC9000 DLL Usage

By default, the PC9000.DLL will be installed in the C:\Campbellsci\PC9000SDK\DLL\ directory. Before distributing a created software package, you must do one of the following to ensure the DLL can be found by your application:

- Place the DLL in the application directory.
- Create a new PATH environmental variable describing the location of the PC9000.DLL during install.
- Place the DLL in the "Windows System" directory during install. The Windows System directory is normally C:\WINDOWS\SYSTEM for Windows 95/98/Me systems, C:\WINNT\SYSTEM32 for Windows NT 4.0 and Windows 2000 systems, or C:\WINDOWS\SYSTEM32 for Windows XP systems.

If desired, PC9000.DLL can alternately be located in other directories; for example, some developers prefer to keep a copy of all key DLL files in the same subdirectory as the applications program that they create. By locating a DLL in the same directory as a custom program, multiple DLL versions can reside on the same machine at the same time. For custom, PC9000.DLL-based applications, this will not be of any concern unless the computer in question also runs a copy of PC9000 Version 5.X software. In that case, if the PC9000 V5.X installation uses an older version of PC9000.DLL than the one provided with this software development kit, and there is some specific concern about PC9000 DLL compatibility, then keeping a separate copy of the DLL in the custom application directory may be warranted.

When invoking a DLL function from within a Visual Basic program for the first time, Windows will look for the DLL in the local subdirectory, and then in the Windows System directory and in any other subdirectories included in the PATH environment variable. If it cannot locate the DLL in any of these directories, VB will raise an Error 53 on the line of code where the first DLL function is invoked.

4.3 Declaring and Calling PC9000 SDK DLL Functions

Declaring and using PC9000.DLL functions within a Visual Basic program is no different than using standard Windows API functions. For general assistance in calling API functions from within Visual Basic, consult the applicable Microsoft Visual Basic and/or MSDN documentation. The required declare statements all appear together in the last section of the PC9000 SDK manual.

For simplicity, the declare statements are often placed in a code (.BAS) module of the programmer's choice. They are then available for use in all form, code, and class modules within the application. If the DLL functionality is to be encapsulated within a VB form or class module, declare statements can be located there, but the "Private" keyword will need to be added at the beginning of each Declare statement, else the code will not compile. In that case, the functions will only be usable from routines within the form/class module.

API-style DLL functions don't raise errors to Visual Basic. The success or failure of each function must be determined within the Visual Basic program by evaluating the return codes, as documented in the PC9000 SDK reference. On the other hand, if fundamental errors occur in linking the DLL function to the application, the Visual Basic runtime engine will not be able to properly call the DLL function in the first place. In such cases, runtime errors will be raised to the application program by the runtime engine itself.

More detailed information regarding DLL usage, function arguments and return codes can be found in the PC9000 SDK manual.

Section 5. BMP5 Direct SDK

5.1 BMP5 Direct SDK Overview

The BMP5 Direct software development kit is for developers that want a simple way to create an application that communicates with any PakBus datalogger using only RS-232 or an intermediate communication that is transparent to RS-232 such as RF400 radios. The application created using the BMP5 Direct SDK will only be able to communicate with one datalogger at a time.

The BMP5 Direct SDK provides the developer access to the communications engine, CORALIB3D.DLL, through the SimplePB.DLL wrapper. The BMP5 Direct SDK install includes examples with source code along with documentation describing how to use the SimplePB.DLL.

5.2 SimplePB Wrapper

The SimplePB.DLL wrapper provides the developer an easy interface through a communications engine to a single PakBus datalogger. The following basic commands are available in the SimplePB.DLL:

- OpenPort() – used to open a COM port
- ClosePort ()– used to close a COM port
- OpenIPPort() – used to open an IP port
- CloseIPPort()– used to close an IP port
- GetClock() – used to query a datalogger for its date and time
- SetClock() – used to set a datalogger's date and time
- GetValue() – used to query a datalogger for a value or group of values
- SetValue() – used to set a field value in a datalogger
- GetData() – used to query data from a datalogger table
- GetDataHeader() – used to get just the header information for a specific table in the datalogger
- GetCommaData() – used to get just the CSV data from a specific table in the datalogger
- File_Send() – used to send a program file to a datalogger
- GetAddress() – used to query the PakBus address of a datalogger
- GetStatus() – used to query the current status of a datalogger

- GetTableNames() – used to query a datalogger for current table names and numbers
- GetDLLVersion() – used to return the version of SimplePB.DLL currently in use
- GetLastResults() – used to display the results of a previous command that currently exist in memory as a string so memory pointers do not need to be managed.

Campbell Scientific Companies

Campbell Scientific, Inc. (CSI)
815 West 1800 North
Logan, Utah 84321
UNITED STATES
www.campbellsci.com
info@campbellsci.com

Campbell Scientific Africa Pty. Ltd. (CSAf)
PO Box 2450
Somerset West 7129
SOUTH AFRICA
www.csafrica.co.za
cleroux@csafrica.co.za

Campbell Scientific Australia Pty. Ltd. (CSA)
PO Box 444
Thuringowa Central
QLD 4812 AUSTRALIA
www.campbellsci.com.au
info@campbellsci.com.au

Campbell Scientific do Brazil Ltda. (CSB)
Rua Luisa Crapsi Orsi, 15 Butantã
CEP: 005543-000 São Paulo SP BRAZIL
www.campbellsci.com.br
suporte@campbellsci.com.br

Campbell Scientific Canada Corp. (CSC)
11564 - 149th Street NW
Edmonton, Alberta T5M 1W7
CANADA
www.campbellsci.ca
dataloggers@campbellsci.ca

Campbell Scientific Ltd. (CSL)
Campbell Park
80 Hathern Road
Shepshed, Loughborough LE12 9GX
UNITED KINGDOM
www.campbellsci.co.uk
sales@campbellsci.co.uk

Campbell Scientific Ltd. (France)
Miniparc du Verger - Bat. H
1, rue de Terre Neuve - Les Ulis
91967 COURTABOEUF CEDEX
FRANCE
www.campbellsci.fr
campbell.scientific@wanadoo.fr

Campbell Scientific Spain, S. L.
Psg. Font 14, local 8
08013 Barcelona
SPAIN
www.campbellsci.es
info@campbellsci.es

Please visit www.campbellsci.com to obtain contact information for your local US or International representative.