INSTRUCTION MANUAL

BMP5 Direct SDK

Revision: 9/17

Campbell Scientific, Inc.
BMP5 SDK End User License Agreement (EULA)

NOTICE OF AGREEMENT: Please carefully read this EULA. By installing or using this software, you are agreeing to comply with the terms and conditions herein. The term "developer" herein refers to anyone using this BMP5 Direct SDK.

By accepting this agreement, you acknowledge and agree that Campbell Scientific may from time-to-time, and without notice, make changes to one or more components of the SDK or make changes to one or more components of other software on which the SDK relies. In no instance will Campbell Scientific be responsible for any costs or liabilities incurred by you or other third parties as a result of these changes.

The core operational files included with this BMP5 Direct SDK (hereinafter referred to as “BMP5 Direct Binaries”) include the files: SimplePB.DLL and coralib3d.dll. Developer may distribute or sell their software including the BMP5 Direct Binaries subject to the terms hereafter set forth.

RELATIONSHIP

Campbell Scientific, Inc. hereby grants a license to use BMP5 Direct Binaries in accordance with the license statement above. No ownership in Campbell Scientific, Inc. patents, copyrights, trade secrets, trademarks, or trade names is transferred by this Agreement. Developer may use these BMP5 Direct Binaries to create as many applications as desired and freely distribute them. Campbell Scientific, Inc. expects no royalties or any other compensation. Developer is responsible for supporting applications created using the BMP5 Direct Binaries.

RESPONSIBILITIES OF DEVELOPER

The Developer agrees:

- To provide a competent programmer familiar with Campbell Scientific, Inc. datalogger programming to write the applications.
- Not to sell or distribute documentation on use of the BMP5 Direct Binaries.
- Not to sell or distribute the applications that are provided as examples in the BMP5 Direct SDK.
- To develop original works. Developers may copy and paste portions of the code into their own applications, but their applications are expected to be unique creations.
- This Agreement does not give Developer the right to sell or distribute any other Campbell Scientific, Inc. Software (e.g., PC200W, VisualWeather, LoggerNet or any of their components, files, documentation, etc.) as part of Developer's application. Distribution of any other Campbell Scientific, Inc. software requires a separate distribution agreement.
- Not to sell or distribute applications that compete directly with any application developed by Campbell Scientific, Inc. or its affiliates.
- Not to use Campbell Scientific’s name, trademarks, or service marks in connection with any program you develop with the SDK. You may not state or infer in any way that Campbell Scientific endorses any program you develop, unless prior written approval is received from Campbell Scientific.

- To assure that each application developed with BMP5 Direct Binaries clearly states the name of the person or entity that developed the application. This information should appear on the first window the user will see.

WARRANTY

There is no written or implied warranty provided with the BMP5 Direct SDK software other than as stated herein. Developer agrees to bear all warranty responsibility of any derivative products distributed by Developer.

TERMINATION

Any license violation or breach of Agreement will result in immediate termination of the developer's rights herein and the return of all BMP5 Direct SDK materials to Campbell Scientific, Inc.

MISCELLANEOUS

Notices required hereunder shall be in writing and shall be given by certified or registered mail, return receipt requested. Such notice shall be deemed given in the case of certified or registered mail on the date of receipt. This Agreement shall be governed and construed in accordance with the laws of the State of Utah, USA. Any dispute resulting from this Agreement will be settled in arbitration.

This Agreement sets forth the entire understanding of the parties and supersedes all prior agreements, arrangements and communications, whether oral or written pertaining to the subject matter hereof. This Agreement shall not be modified or amended except by the mutual written agreement of the parties. The failure of either party to enforce any of the provisions of this Agreement shall not be construed as a waiver of such provisions or of the right of such party thereafter to enforce each and every provision contained herein. If any term, clause, or provision contained in this Agreement is declared or held invalid by a court of competent jurisdiction, such declaration or holding shall not affect the validity of any other term, clause, or provision herein contained. Neither the rights nor the obligations arising under this Agreement are assignable or transferable.

If within 30 days of receiving the BMP5 Direct SDK product developer does not agree to the terms of license, developer shall return all materials without retaining any copies of the product and shall remove any use of the BMP5 Direct Binaries in any applications developed or distributed by Developer. In the absence of such return, CSI shall consider Developer in agreement with the herein, stated license terms and conditions.

Limited Guarantee

The following warranties are in effect for ninety (90) days from the date of shipment of the original purchase. These warranties are not extended by the installation of upgrades or patches offered free of charge.

Campbell Scientific warrants that the installation media on which the software is recorded and the documentation provided with it are free from physical defects in materials and workmanship under normal use. The warranty does not cover any installation media that has been damaged, lost, or abused. You are urged to make a backup copy (as set forth above) to protect your investment. Damaged or lost media is the sole responsibility of the licensee and will not be replaced by Campbell Scientific.

Campbell Scientific warrants that the software itself will perform substantially in accordance with the specifications set forth in the instruction manual when properly installed and used in a manner consistent with the published recommendations, including recommended system requirements. Campbell Scientific does not warrant that the software will meet licensee's requirements for use, or that the software or documentation are error free, or that the operation of the software will be uninterrupted.

Campbell Scientific will either replace or correct any software that does not perform substantially according to the specifications set forth in the instruction manual with a corrected copy of the software or corrective code. In the case of significant error in the installation media or documentation, Campbell Scientific will correct errors without charge by providing new media, addenda, or substitute pages. If Campbell Scientific is unable to replace defective media or documentation, or if it is unable to provide corrected software or corrected documentation within a reasonable time, it will either replace the software with a functionally similar program or refund the purchase price paid for the software.

All warranties of merchantability and fitness for a particular purpose are disclaimed and excluded. Campbell Scientific shall not in any case be liable for special, incidental, consequential, indirect, or other similar damages even if Campbell Scientific has been advised of the possibility of such damages. Campbell Scientific is not responsible for any costs incurred as a result of lost profits or revenue, loss of use of the software, loss of data, cost of re-creating lost data, the cost of any substitute program, telecommunication access costs, claims by any party other than licensee, or for other similar costs.

This warranty does not cover any software that has been altered or changed in any way by anyone other than Campbell Scientific. Campbell Scientific is not responsible for problems caused by computer hardware, computer operating systems, or the use of Campbell Scientific's software with non-Campbell Scientific software.

Licensee's sole and exclusive remedy is set forth in this limited warranty. Campbell Scientific's aggregate liability arising from or relating to this agreement or the software or documentation (regardless of the form of action; e.g., contract, tort, computer malpractice, fraud and/or otherwise) is limited to the purchase price paid by the licensee.

PLEASE READ FIRST

About this manual

Please note that this manual was originally produced by Campbell Scientific Inc. primarily for the North American market. Some spellings, weights and measures may reflect this origin.

Some useful conversion factors:

Area: 1 in^2 (square inch) = 645 mm^2

Length: 1 in. (inch) = 25.4 mm
1 ft (foot) = 304.8 mm
1 yard = 0.914 m
1 mile = 1.609 km

Mass: 1 oz. (ounce) = 28.35 g
1 lb (pound weight) = 0.454 kg

Pressure: 1 psi (lb/in^2) = 68.95 mb

Volume: 1 UK pint = 568.3 ml
1 UK gallon = 4.546 litres
1 US gallon = 3.785 litres

In addition, while most of the information in the manual is correct for all countries, certain information is specific to the North American market and so may not be applicable to European users.

Differences include the U.S standard external power supply details where some information (for example the AC transformer input voltage) will not be applicable for British/European use. *Please note, however, that when a power supply adapter is ordered it will be suitable for use in your country.*

Reference to some radio transmitters, digital cell phones and aerials may also not be applicable according to your locality.

Some brackets, shields and enclosure options, including wiring, are not sold as standard items in the European market; in some cases alternatives are offered. Details of the alternatives will be covered in separate manuals.

Part numbers prefixed with a "#" symbol are special order parts for use with non-EU variants or for special installations. Please quote the full part number with the # when ordering.

Recycling information

At the end of this product's life it should not be put in commercial or domestic refuse but sent for recycling. Any batteries contained within the product or used during the products life should be removed from the product and also be sent to an appropriate recycling facility.

Campbell Scientific Ltd can advise on the recycling of the equipment and in some cases arrange collection and the correct disposal of it, although charges may apply for some items or territories.

For further advice or support, please contact Campbell Scientific Ltd, or your local agent.

Campbell Scientific Ltd, 80 Hathern Road, Shepshed, Loughborough, LE12 9GX, UK
Tel: +44 (0) 1509 601141 Fax: +44 (0) 1509 601091
Email: support@campbellsci.co.uk
www.campbellsci.co.uk

Table of Contents

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

1. BMP5 Direct SDK Overview 1

1.1 General Notes on BMP5 Direct SDK Usage 1
1.2 Datalogger Program Table Structure 2
1.3 Developing Applications Using the .NET Framework 2

2. SimplePB.dll Reference 2

2.1 OpenPort() 2
2.2 ClosePort() 3
2.3 OpenIPPort() 3
2.4 CloseIPPort() 3
2.5 GetClock() 4
2.6 SetClock() 4
2.7 GetValue() 5
2.8 SetValue() 6
2.9 GetData() 7
2.10 GetDataHeader() 8
2.11 GetCommaData() 9
2.12 File_Send() 10
2.13 GetAddress() 11
2.14 GetStatus() 12
2.15 GetTableNames() 13
2.16 GetDLLVersion() 13
2.17 GetLastResults() 14
2.18 FileControl() 14
2.19 SetSecurity() 15
2.20 GetTableRecordsCount() 15

Appendix

A. Sample Program Table Structure A-1

A.1 CR200 Datalogger Program Tables A-2
A.1.1 CR200 Datalogger Program A-3
A.2 WeatherHawk Weather Station Tables A-4

CRBasic Example

A-1. CR200 Datalogger Program A-3

BMP5 Direct SDK

1. BMP5 Direct SDK Overview

The BMP5 Direct Software Development Kit (SDK) comprises a simple call-level API (SimplePB.dll) wrapper for the included coralib3d.dll communications server. Client applications developed using the SDK will execute calls to the C-type functions exposed by the SimplePB.dll to effect datalogger communications via the coralib3d.dll.

The SDK components and example applications are installed by default in *C:\Campbellsci\BMP5DirectSDK*. The SDK does not require registration on the host PC. However, the SimplePB.dll wrapper and the coralib3d.dll communications server must be installed into the same folder as the client application's executable.

NOTE

If you have been using version 4.3 or earlier of the BMP5Direct SDK on your machine, you may wish to uninstall, remove, or relocate the files located in the C:\Campbellsci\BMP5DirectSDK\Examples folder before installing this version. This will help avoid confusion about code locations after installation.

This version uses a folder structure in this form:
\Examples\C#
\Examples\MFC-VS2015
\Examples\VB.NET

Older versions use a folder structure like this:
\Examples\C#\SmplPB_CS
\Examples\MFC
\Examples\VBNET

1.1 General Notes on BMP5 Direct SDK Usage

The SDK supports only PakBus® datalogger communication via a serial port (COM) link or a TCP/IP socket connection. PakBus packet routing is **not** supported. Only a single, directly connected (leaf node) PakBus datalogger is accessible at any one time.

The "dialing" of communication devices such as a dial-up phone modem or an RF500M modem is **not** supported. However, a connection via a transparent bridging device such as an RF450 or an RF401 radio is possible.

A successful call to the OpenPort() or OpenIPPort() function will start the CORALIB3D communications server (hereafter, referred to as "the Server"). The application should stop the Server by calling either the CloseIPPort() or ClosePort() function before exiting.

Both the Server and the SimplePB.dll wrapper write log files to *C:\Campbellsci\SimplePB\Ver#\logs*; where "Ver#" is the version number of the SimplePB.dll. These files can provide useful information about the Server's behavior when troubleshooting connection issues. Refer to Appendix E of the *LoggerNet Instruction Manual* for information regarding log files.

Once a connection is established, additional functions can be called to accomplish the desired task. For example: send and manage datalogger programs, check or set the datalogger clock, query the datalogger for data table information, get/set table values, and collect table records.

1.2 Datalogger Program Table Structure

The application developer must understand the table structure of the program running in the datalogger because table and field names and numbers are used as arguments for many of the functions exposed by the SimplePB.dll. The GetTableNames() function can be used to obtain a list of tables and their associated numbers. Refer to Appendix A, *Sample Program Table Structure (p. A-1)*, for information regarding the table structure of PakBus dataloggers.

1.3 Developing Applications Using the .NET Framework

From the perspective of the .NET Framework, the SimplePB.dll is unmanaged code; not unlike the native functions of the Windows® API. Therefore, the platform invoke (P/Invoke) services provided by the common language runtime (CLR) can be utilized to directly access the SimplePB.dll functions.

Fundamentally, the implementation involves attaching a "DllImport" attribute (requires the System.Runtime.InteropServices namespace) to a static or shared declaration of the external function. The DllImport attribute notifies the CLR of the name of the DLL to load and the exposed function to call. An example of using the OpenPort() function is shown in the C# code snippet below:

```
[DllImportAttribute("SimplePB.dll", EntryPoint = "OpenPort", CallingConvention =
CallingConvention.StdCall)]
public static extern int OpenPort(int comPortNumber, int baudRate);
```

Attention should be paid to the marshalling of parameter data types. Particularly, the "Strings" in the managed code and the "char" arrays in the unmanaged functions. The SimplePB.dll functions expect the "char" arrays to be null-terminated and UTF8 encoded.

The recommended method for accommodating the C-type pointers used by many of the SimplePB.dll functions is to marshal the parameter as a System.IntPtr type. In the case of pointer to a pointer types (char**), pass the IntPtr by reference (ref or ByRef). Optionally, the "unsafe" keyword in C# allows for the direct use of pointer types.

Best practice is to encapsulate or "wrap" the SimplePB.dll function calls into a shared class and expose them to application code via public functions. This approach is implemented in both the C# and VB.NET example applications provided with the SDK.

2. SimplePB.dll Reference

The following C-style functions are exposed by the SimplePB.dll.

2.1 OpenPort()

Opens a COM port (serial port) on the host computer using the specified COM port and baud rate.

Syntax

int _stdcall OpenPort (int com_port_no, int baud)

Parameters

com_port_no: COM port to open.

baud: Baud rate to be used by the COM port.

Return Codes

0 = Successful.
–1 = Port failed to open or is already open.

2.2 ClosePort()

Closes the currently open COM port or IP port connection.

Syntax

int _stdcall ClosePort()

Return Codes

0 = Successful.
–1 = Port failed to close or was not open.

2.3 OpenIPPort()

Opens a TCP socket connection with a network device using the specified IP address and port number. An appropriate device would be a cell modem, serial server, or datalogger. IPv4 and IPv6 addresses or fully qualified domain names are supported.

Syntax

int _stdcall OpenIPPort (char const *ip_address, int tcp_port)

Parameters

ip_address: Pointer to the memory location of a char array defining the IP address to be used. Must be a null-terminated array of UTF8 encoded bytes.

tcp_port: Port number that will be used when communicating with the datalogger.

Return Codes

0 = Successful.
–1 = IP port failed to open or is already open.

2.4 CloseIPPort()

Closes the currently open IP port (synonymous with ClosePort()).

Syntax

int _stdcall CloseIPPort()

Return Codes

0 = Successful.
–1 = IP port failed to close or was not open.

2.5 GetClock()

Queries the datalogger for its current date and time.

Syntax

int _stdcall GetClock (int pakbus_address, int device_type, char **return_data, int *return_data_len)

Parameters

pakbus_address: PakBus® address of the datalogger.

device_type: Type of datalogger:
1 = CR200
2 = CR10XPB, CR23XPB, CR510PB
3 = CR1000
4 = CR3000
5 = CR800 Series
9 = CR6 Series
13 = CR300 Series
14 = CR1000X Series

return_data: Pointer to a pointer to the memory location of a char array containing the data returned from the datalogger.

return_data_len: Pointer to the memory location containing the length of the char array returned from the DLL.

Return Codes

0 = Successful.
–1 = Communication timed out.
–2 = Port is not open.

Example of data returned by function call

14:12:35 04/16/2004

2.6 SetClock()

Sets the date and time of the datalogger to match the host computer's clock.

Syntax

int _stdcall SetClock (int pakbus_address, int device_type, char **return_data, int *return_data_len)

Parameters

pakbus_address: PakBus address of the datalogger.

device_type: Type of datalogger:
1 = CR200
2 = CR10XPB, CR23XPB, CR510PB
3 = CR1000
4 = CR3000
5 = CR800 Series
9 = CR6 Series
13 = CR300 Series
14 = CR1000X Series

return_data: Pointer to a pointer to the memory location of a char array containing the data returned from the datalogger.

return_data_len: Pointer to the memory location containing the length of the char array returned from the DLL.

Return Codes

0 = Successful.
–1 = Communication timed out.
–2 = Port is not open.

Example of data returned by function call

14:22:51 04/16/2004 (Old Time Old Date)
14:22:27 04/16/2004 (New Time New Date)

2.7 GetValue()

Queries the datalogger for a value or an array of values from the specified table and field.

Syntax

int _stdcall GetValue (int pakbus_address, int device_type, int swath, char const *table_name, char const *field_name, char **return_data, int *return_data_len)

Parameters

pakbus_address: PakBus address of the datalogger.

device_type: Type of datalogger:
1 = CR200
2 = CR10XPB, CR23XPB, CR510PB
3 = CR1000
4 = CR3000
5 = CR800 Series
9 = CR6 Series
13 = CR300 Series
14 = CR1000X Series

swath: The number of values to collect starting at the location specified in the field_name parameter. The requested swath must be within the bounds of an indexed array or an error will occur.

table_name: Pointer to the memory location of a char array defining the name of the table in which the value(s) exist. Must be a null-terminated array of UTF8 encoded bytes.

field_name: Pointer to the memory location of a char array defining the field in which the value(s) exist. Field_name may specify an array element (example: "Temp(3)"). Must be a null-terminated array of UTF8 encoded bytes.

return_data: Pointer to a pointer to the memory location of a char array containing the data returned from the datalogger.

return_data_len: Pointer to the memory location containing the length of the char array returned from the DLL.

Return Codes

0 = Successful.
–1 = Communication timed out.
–2 = Port is not open.

Example of data returned by function call

12.753,111.9,1.239 (Swath of 3 values from fields)

2.8 SetValue()

Set the value of the specified field in the specified datalogger table.

Syntax

int _stdcall SetValue (int pakbus_address, int device_type, char const *table_name, char const *field_name, char const *value)

Parameters

pakbus_address: PakBus address of the datalogger.

device_type: Type of datalogger:
1 = CR200
2 = CR10XPB, CR23XPB, CR510PB
3 = CR1000
4 = CR3000
5 = CR800 Series
9 = CR6 Series
13 = CR300 Series
14 = CR1000X Series

table_name: Pointer to the memory location of a char array defining the name of the table in which the field will be set. Must be a null-terminated array of UTF8 encoded bytes.

field_name: Pointer to the memory location of a char array defining the field that will be set with the new value. Must be a null-terminated array of UTF8 encoded bytes.

value: Pointer to the memory location of a char array defining the value used to set the field. Must be a null-terminated array of UTF8 encoded bytes.

Return Codes

0 = Successful.
–1 = Communication timed out.
–2 = Port is not open.

2.9 GetData()

Queries the datalogger for records and returns each record formatted as a list of fieldname:value pairs. A return code of '1' indicates that additional records remain to be transferred. The function call should be iterated until the return code is '0'.

Syntax

int _stdcall GetData (int pakbus_address, int device_type, int table_no, int record_no, char **return_data, int *return_data_len)

Parameters

pakbus_address: PakBus address of the datalogger.

device_type: Type of datalogger:
1 = CR200
2 = CR10XPB, CR23XPB, CR510PB
3 = CR1000
4 = CR3000
5 = CR800 Series
9 = CR6 Series
13 = CR300 Series
14 = CR1000X Series

table_no: The number for the table from which to collect data.

record_no: The record number where data collection will start. All records following this record number will be included in the collection. Therefore, if the record number is set to 0, all records in the table will be collected. In addition, if the record number specified does not exist in the datalogger, all existing records from the oldest to the newest will be returned. However, if the record number is set to a negative number, only the most recent record in the table will be collected. There is not a way to specify and collect a single record from a table using this command unless that record is the most recent record in the table.

return_data: Pointer to a pointer to the memory location of a char array containing the data returned from the datalogger.

return_data_len: Pointer to the memory location containing the length of the char array returned from the DLL.

Return Codes

0 = Complete.
1 = Successful but more data to collect.
–1 = Communication timed out.
–2 = Port is not open.
–3 = Invalid table number.

Example of data returned by function call

```
"2004-04-16 14:18:03",1          (Time stamp, Record number)
1,OSversion,v03A                 (Field number, Field name, Field value)
2,OSDate,06-Jan-04
3,ProgName,BATT.CR2
4,ProgSig,54451
5,CalOffset,2.625
6,PakBusAddress,1
7,RfInstalled,424
8,RfNetAddr,0
9,RfAddress,0
10,RfHopSeq,0
11,RfPwrMode,RF1_Sec
12,Rf_ForceOn,0
13,RfSignalLevel,0
14,RfRxPakBusCnt,0
15,VarOutOfBounds,0
16,SkipScan,0
17,TrapCode,0
18,WatchDogCnt,0
19,ResetTables,0
20,BattVoltage,12.3943
```

2.10 GetDataHeader()

Returns the TOA5 file header for the specified table.

Syntax

```
int _stdcall GetDataHeader ( int pakbus_address, int device_type, int table_no,
char **return_data, int *return_data_len )
```

Parameters

pakbus_address: PakBus address of the datalogger.

device_type: Type of datalogger:
1 = CR200
2 = CR10XPB, CR23XPB, CR510PB
3 = CR1000
4 = CR3000
5 = CR800 Series
9 = CR6 Series
13 = CR300 Series
14 = CR1000X Series

table_no: The number of the table for which the header will be generated.

return_data: Pointer to a pointer to the memory location of a char array containing the header returned by the DLL.

return_data_len: Pointer to the memory location containing the length of the char array returned from the DLL.

Return Codes

0 = Successful.
1 = Successful but more data to collect.
–1 = Communication timed out.
–2 = Port is not open.
–3 = Invalid table number.

Example of data returned by function call

"TIMESTAMP","RECORD", OSVersion, OSDate, OSSignature

2.11 GetCommaData()

Queries the datalogger for records and returns each record in a TOA5 comma-separated format. A return code of '1' indicates that additional records remain to be transferred. The function call should be iterated until the return code is '0'.

Syntax

int _stdcall GetData (int pakbus_address, int device_type, int table_no, int record_no, char **return_data, int *return_data_len)

Parameters

pakbus_address: PakBus address of the datalogger.

device_type: Type of datalogger:
1 = CR200
2 = CR10XPB, CR23XPB, CR510PB
3 = CR1000
4 = CR3000
5 = CR800 Series
9 = CR6 Series
13 = CR300 Series
14 = CR1000X Series

table_no: The number for the table from which to collect data.

record_no: The record number where data collection will start. All records following this record number will be included in the collection. Therefore, if the record number is set to 0, all records in the table will be collected. In addition, if the record number specified does not exist in the datalogger, all existing records from the oldest to the newest will be returned. However, if the record number is set to a negative number, only the most recent record in the table will be collected. There is not a way to specify and collect a single record from a table using this command unless that record is the most recent record in the table.

return_data: Pointer to a pointer to the memory location of a char array containing the data returned from the datalogger.

return_data_len: Pointer to the memory location containing the length of the char array returned from the DLL.

Return Codes

0 = Complete.
1 = Successful but more data to collect.
–1 = Communication timed out.
–2 = Port is not open.
–3 = Invalid table number.

Example of data returned by function call

"2005-09-08 14:13:47",1,"CR1000.Std.05","050624",47178

2.12 File_Send()

Sends the specified program to the datalogger. A return code of '1' indicates that a fragment of the file has been successfully transferred, but additional fragments remain. The array pointed to by 'return_data' will contain a string indicating the current progress of the file transfer. The function call should be iterated until the return code is '0'. Once the operation is complete, 'return_data' will point to an array containing the compile results.

Sending a .CR2 file to a CR200 will cause the Server to attempt to invoke the CR200 compiler located at *C:\Campbellsci\Lib\CR200Compilers*. If the compiler is not installed, an error will be returned.

Syntax

int _stdcall File_Send (int pakbus_address, int device_type, char const *file_name, char **return_data, int *return_data_len)

Parameters

pakbus_address: PakBus address of the datalogger.

device_type: Type of datalogger:
1 = CR200
2 = CR10XPB, CR23XPB, CR510PB
3 = CR1000
4 = CR3000
5 = CR800 Series
9 = CR6 Series
13 = CR300 Series
14 = CR1000X Series

file_name: Pointer to the memory location of a char array defining the path and file name of the program file to be sent to the datalogger. Must be a null-terminated array of UTF8 encoded bytes.

return_data: Pointer to a pointer to the memory location of a char array containing the data returned from the DLL.

return_data_len: Pointer to the memory location containing the length of the char array returned from the DLL.

Return Codes

0 = Complete.
1 = Successful but more data to transfer.
–1 = Communication timed out.
–2 = Port is not open.
–3 = Cannot open source file.
–4 = File name is too long.
–5 = Datalogger timed out.
–6 = File offset does not match.
–7 = Datalogger reported an error.
–8 = File control error.
–9 = Cannot get program status.

Example of data returned from a CR1000

OS Version: CR1000.Std.05
OS Signature: 19128
Serial Number: 1031
PowerUp Progr: CPU:Program.cr1
Compile Status: Datalogger Program Running
Program Name: CPU:Program.cr1
Program Sig.: 32083
Compile Result: Compiled in SequentialMode.

2.13 GetAddress()

Queries the open port for a connected PakBus device; if found, the PakBus address is returned. If multiple PakBus devices are connected, only the first to respond is reported.

Syntax

int _stdcall GetAddress (int device_type, char **return_data, int *return_data_len)

Parameters

device_type: Type of datalogger:
1 = CR200
2 = CR10XPB, CR23XPB, CR510PB
3 = CR1000
4 = CR3000
5 = CR800 Series
9 = CR6 Series
13 = CR300 Series
14 = CR1000X Series

return_data: Pointer to a pointer to the memory location of a char array containing the data returned from the DLL.

return_data_len: Pointer to the memory location containing the length of the char array returned from the DLL.

Return Codes

0 = Successful.
–1 = Communication timed out.
–2 = Port is not open.

Example of data returned by function call

PakBusAddress=1;

2.14 GetStatus()

Queries the datalogger for a summary of its current status.

Syntax

int _stdcall GetStatus (int pakbus_address, int device_type, char **return_data, int *return_data_len)

Parameters

pakbus_address: PakBus address of the datalogger.

device_type: Type of datalogger:
1 = CR200
2 = CR10XPB, CR23XPB, CR510PB
3 = CR1000
4 = CR3000
5 = CR800 Series
9 = CR6 Series
13 = CR300 Series
14 = CR1000X Series

return_data: Pointer to a pointer to the memory location of a char array containing the data returned from the datalogger.

return_data_len: Pointer to the memory location containing the length of the char array returned from the DLL.

Return Codes

0 = Successful.
–1 = Communication timed out.
–2 = Port is not open.

Example of data returned from a CR200

OS Version: v03A
OS Signature: 43529
Serial Number:
PowerUp Progr:
Compile Status: Datalogger Program Running
Program Name: BATT.CR2
Program Sig.: 54451
Compile Result: Program Running
Batt=12.38V

2.15 GetTableNames()

Query the datalogger for its table names and numbers.

Syntax

int _stdcall GetTableNames (int pakbus_address, int device_type, char **return_data, int *return_data_len)

Parameters

pakbus_address: PakBus address of the datalogger.

device_type: Type of datalogger:
1 = CR200
2 = CR10XPB, CR23XPB, CR510PB
3 = CR1000
4 = CR3000
5 = CR800 Series
9 = CR6 Series
13 = CR300 Series
14 = CR1000X Series

return_data: Pointer to a pointer to the memory location of a char array containing the data returned from the datalogger.

return_data_len: Pointer to the memory location containing the length of the char array returned from the DLL.

Return Codes

0 = Successful.
–1 = Communication timed out.
–2 = Cannot read table definitions from the datalogger.

Example of data returned by function call

1 Status
2 DataTable1
3 DataTable2
4 Public

2.16 GetDLLVersion()

Gets the version of the SimplePB.dll being used.

Syntax

int _stdcall GetDLLVersion (char **return_data, int *return_data_len)

Parameters

return_data: Pointer to a pointer to the memory location of a char array containing the data returned from the datalogger.

return_data_len: Pointer to the memory location containing the length of the char array returned from the DLL.

Return Codes

0 = Successful.

Example of data returned by function call

SimplePB.dll Version 2.0 / 2,2,3,0

2.17 GetLastResults()

Retrieves the return_data results from memory for the previous function as a String. This function is useful for developers that don't want to manage memory pointers. A new BSTR is allocated each time this function is called.

Syntax

BSTR _stdcall GetLastResults ()

2.18 FileControl()

Used to control compilation and execution of the datalogger program and do file management.

Syntax

int _stdcall FileControl (int pakbus_address, int device_type, char const *file_name, int command)

Parameters

pakbus_address: PakBus address of the datalogger.

device_type: Type of datalogger:
1 = CR200
2 = CR10XPB, CR23XPB, CR510PB
3 = CR1000
4 = CR3000
5 = CR800 Series
9 = CR6 Series
13 = CR300 Series
14 = CR1000X Series

file_name: Pointer to the memory location of a char array defining the path and file name of the device or file subject to the specified command.

command: Specifies the action to be executed upon the specified device or file:
1 = Compile and run; marks the program as "run on power up"
2 = Run on power up
3 = Make hidden
4 = Delete file
5 = Format device
6 = Compile and run (preserve data if no table changed)
7 = Stop running program
8 = Stop running program and delete associated files
9 = Make the specified file the operating system
10 = Compile and run but do not change the "run on power up" program
11 = Pause execution of the running program

12 = Resume execution of the running program
13 = Stop the running program, delete associated files, and mark as run now and on power up
14 = Stop the running program, delete associated files, and mark as run now but not on power up

Return Codes

0 = Successful.
–1 = Communication timed out.
–2 = Port is not open.

2.19 SetSecurity()

Sets the security code that will be used to communicate with the datalogger.

Syntax

int _stdcall SetSecurity (int security_code)

Parameter

Security_code: Security code to use.

Return Codes

0 = Success.
–1 = Failure.

2.20 GetTableRecordsCount()

Queries the datalogger to determine the number of records that are available for collection from the specified table.

Syntax

int _stdcall GetTableRecordsCount (int pakbus_address, int device_type, int table_no, unsigned long *records_count)

Parameters

pakbus_address: The PakBus address of the datalogger.

Device_type: Type of datalogger:
1 = CR200
2 = CR10XPB, CR23XPB, CR510PB
3 = CR1000
4 = CR3000
5 = CR800 Series
9 = CR6 Series
13 = CR300 Series
14 = CR1000X Series

table_no: Number of the table from which to get the records count.

records_count: Pointer to the memory location where the records count value will be written.

Return Codes

0 = Successful.
1 = Successful but more data to collect.
–1 = Communication timed out.
–2 = Port is not open.
–3 = Invalid table number.

Appendix A. Sample Program Table Structure

The table structure of a PakBus® datalogger is given in the example below. This example shows a datalogger with two user defined tables plus the **Status** table and **Public** or **Inlocs** table. The second table in the example below contains three records and the third table contains four records. Both the **Status** table and **Public** or **Inlocs** table will always return the most recent records and will not contain any historical data records.

The first table is the **Status** table, which shows the status of the datalogger. The **Public** or **Inlocs** table contains all public variables or input locations. All other tables found in the datalogger are created and defined by the user in the datalogger program. The tables in a PakBus datalogger will always contain a record number and timestamp followed by the data fields.

Table 1 – Status

Record No	Time Stamp	Data Field 1	Data Field 2	Data Field 3-19	Data Field 20

Table 2 – User Defined

RN 0	Time Stamp	Data Field 1	Data Field 2
RN 1	Time Stamp	Data Field 1	Data Field 2
RN 2	Time Stamp	Data Field 1	Data Field 2

Table 3 – User Defined

RN 0	Time Stamp	Data Field 1	Data Field 2	Data Field 3	Data Field 4	Data Field 5
RN 1	Time Stamp	Data Field 1	Data Field 2	Data Field 3	Data Field 4	Data Field 5
RN 2	Time Stamp	Data Field 1	Data Field 2	Data Field 3	Data Field 4	Data Field 5
RN 3	Time Stamp	Data Field 1	Data Field 2	Data Field 3	Data Field 4	Data Field 5

Table 4 – Public or Inlocs

Record No	Time Stamp	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6

A.1 CR200 Datalogger Program Tables

The following tables show the table structure from a program installed in a CR200 datalogger. This program measures and stores the minimum battery voltage and the minimum and maximum temperature over a 60-minute interval. When communicating with a datalogger using the BMP5 Direct SDK, knowing the table structure of the running program is necessary for some commands.

Table Number 1 – Status

Field Number	Field Name	Units	Notes:
Field 1	OSVersion		Operating system version
Field 2	OSDate		Date of operating system
Field 3	ProgName		The name of the program running in the datalogger
Field 4	ProgSig		The signature of the running program
Field 5	CalOffset		
Field 6	PakBusAddress		The PakBus address of the datalogger (1-4094)
Field 7	RfInstalled		Radio detected
Field 8	RfNetAddr		Valid addresses are 0-63
Field 9	RfAddress		Valid addresses are 0-1023
Field 10	RfHopSeq		Valid numbers are 0-6
Field 11	RfPwrMode		RF1_Sec
Field 12	Rf_ForceOn		
Field 13	RfSignalLevel		RF signal strength should be above 40
Field 14	RfRxPakBusCnt		
Field 15	VarOutOfBounds		
Field 16	SkipScan		Program didn’t complete before the next execution interval
Field 17	TrapCode		
Field 18	WatchDogCnt		Number of watchdog errors
Field 19	ResetTables		Clears all stored data
Field 20	BattVoltage	Volts	Current battery voltage

Table Number 2 – Hourly: The **Hourly** table contains the minimum battery voltage and the minimum and maximum temperature over a 60-minute interval.

Field Number	Field Name	Units	Notes:
Field 1	Battery_Min	Volts	
Field 2	Battery_Time	Time	
Field 2	Temp_Min	Deg C	
Field 3	Temp_Max	Deg C	

Table Number 3 – Public: The **Public** table contains only the most recent "real-time" record for the variable described in the datalogger program.

Field Number	Field Name	Units	Notes:
Field 1	Batt_Volt	Volts	
Field 2	Temp	Deg C	

A.1.1 CR200 Datalogger Program

CRBasic Example A-1. CR200 Datalogger Program

```
'CR200 Series
'Declare Variables and Units
Public Batt_Volt, Temp

Units Batt_Volt=Volts
Units Temp=Deg C

'Define Data Tables
DataTable(Hourly,True,-1)
  DataInterval(0,60,Min)
  Minimum(1,Batt_Volt,False,True)
  FieldNames("Battery_MIN,Battery_Time")
  Maximum(1,Temp,False,False)
  Minimum(1,Temp,False,False)
EndTable
'Main Program
BeginProg
  Scan(10,Sec)
    'Default Datalogger Battery Voltage measurement Batt_Volt:
    Battery(Batt_Volt)
    '109 Temperature Probe measurement Temp:
    Therm109(Temp,1,1,1,1.0,0.0)
    'Call Data Tables and Store Data
    CallTable(Hourly)
  NextScan
EndProg
```

A.2 WeatherHawk Weather Station Tables

The following tables show the table structure from a default WeatherHawk® weather station program installed in a CR200 datalogger. When communicating with a datalogger using the BMP5 Direct SDK, knowing the table structure of the running program is necessary for some commands

Table Number 1 – Status

Field Number	Field Name	Units	Notes:
Field 1	OSVersion		Operating system version
Field 2	OSDate		Date of operating system
Field 3	ProgName		The name of the program running in the datalogger
Field 4	ProgSig		The signature of the running program
Field 5	CalOffset		
Field 6	PakBusAddress		The PakBus address of the datalogger (1-4094)
Field 7	RfInstalled		Radio detected
Field 8	RfNetAddr		Valid addresses are 0-63
Field 9	RfAddress		Valid addresses are 0-1023
Field 10	RfHopSeq		Valid numbers are 0-6
Field 11	RfPwrMode		RF1_Sec
Field 12	Rf_ForceOn		
Field 13	RfSignalLevel		RF signal strength should be above 40
Field 14	RfRxPakBusCnt		
Field 15	VarOutOfBounds		
Field 16	SkipScan		Program didn’t complete before the next execution interval
Field 17	TrapCode		
Field 18	WatchDogCnt		Number of watchdog errors
Field 19	ResetTables		Clears all stored data
Field 20	BattVoltage	Volts	Current battery voltage

Table Number 2 – SiteVal: The **SiteVal** table contains values that are stored for calculations by the WeatherHawk program. Data is only stored when field "SaveSite" in the **Public** table is set to one.

Field Number	Field Name	Units	Notes:
Field 1	Altitude_m	Meter	
Field 2	Latitude	Degrees	
Field 3	Longitude	Degrees	
Field 4	BPoffset_KPa	KPa	
Field 5	Int_Timer	Minutes	

Table Number 3 – Data1: This table contains data output at the Int_timer rate from the **Public** table. For example, if Int_timer = 15 min, this table contains 15 min data.

Field Number	Field Name	Units	Notes:
Field 1	BatVolt_V	Volts	
Field 2	BatVolt_V_Min	Volts	
Field 3	AirTemp_C_Avg	Celsius	
Field 4	RH_Avg	Percent	
Field 5	WindSpeed_ms_Avg	m/s	
Field 6	Solar_Avg	W/m^2	
Field 7	ETo	mm	
Field 8	AirTemp_C_Min	Celsius	
Field 9	AirTemp_C_TMn	Time	Example: 2004-01-25 13:49:50
Field 10	Max_AirTemp	Celsius	
Field 11	AirTemp_C_C_Tmx	Time	Example: 2004-01-25 13:49:50
Field 12	WindSpeed_ms_WVc(1)	m/s	Average wind speed
Field 13	WindSpeed_ms_WVc(2)	Degrees	Unit vector wind direction
Field 14	WindSpeed_ms_Max	m/s	
Field 15	Baromete_KPa	KPa	
Field 16	RainYearly_mm	mm	

Table Number 4 – Data2: This table contains daily data values.

Field Number	Field Name	Units	Notes:
Field 1	BatVolt_V_Min	Volts	
Field 2	AirTemp_C_Max	Celsius	
Field 3	AirTemp_C_Min	Celsius	
Field 4	WindSpeed_ms_Max	m/s	
Field 5	RainYearly_mm	mm	
Field 6	DailyETo_mm	mm	

Table Number 5 – Public: The **Public** table contains only the most recent "real-time" record.

Field Number	Field Name	Units	Notes:
Field 1	SaveSite		Set to one to save values to SiteVal table
Field 2	Latitude	Degrees	Decimal format: 41 deg 45 min = 14.75
Field 3	Longitude	Degrees	Decimal format: 41 deg 45 min = 14.75
Field 4	Altitude_m	Metre	
Field 5	Bpoffset_KPa	KPa	
Field 6	Int_timer	Minutes	
Field 7	RainReset		Set to 1 to reset RainYearly_mm variable
Field 8	BatVolt_V	Volts	
Field 9	Solar	W/m^2	
Field 10	AirTemp_C	Celsius	
Field 11	RH	Percent	
Field 12	Barometer_KPa	KPa	Sea level adjustment barometric pressure
Field 13	WindSpeed_ms	m/s	
Field 14	WindDirect_deg	Degrees	
Field 15	RainYearly_mm	mm	Running sum of rainfall
Field 16	DailyETo_mm	mm	Running sum of Eto (resets at midnight)

Campbell Scientific Companies

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

Campbell Scientific Canada Corp.
14532 – 131 Avenue NW
Edmonton AB T5L 4X4
CANADA
www.campbellsci.ca • dataloggers@campbellsci.ca

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

Campbell Scientific Centro Caribe S.A.
300 N Cementerio, Edificio Breller
Santo Domingo, Heredia 40305
COSTA RICA
www.campbellsci.cc • info@campbellsci.cc

Campbell Scientific Southeast Asia Co., Ltd.
877/22 Nirvana@Work, Rama 9 Road
Suan Luang Subdistrict, Suan Luang District
Bangkok 10250
THAILAND
www.campbellsci.asia • info@campbellsci.asia

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

Campbell Scientific Australia Pty. Ltd.
PO Box 8108
Garbutt Post Shop QLD 4814
AUSTRALIA
www.campbellsci.com.au • info@campbellsci.com.au

Campbell Scientific Ltd.
3 Avenue de la Division Leclerc
92160 ANTONY
FRANCE
www.campbellsci.fr • info@campbellsci.fr

Campbell Scientific (Beijing) Co., Ltd.
8B16, Floor 8 Tower B, Hanwei Plaza
7 Guanghua Road
Chaoyang, Beijing 100004
P.R. CHINA
www.campbellsci.com • info@campbellsci.com.cn

Campbell Scientific Ltda.
Fahrenheitstraße 13
28359 Bremen
GERMANY
www.campbellsci.de • info@campbellsci.de

Campbell Scientific do Brasil Ltda.
Rua Apinagés, nbr. 2018 ─ Perdizes
CEP: 01258-00 ─ São Paulo ─ SP
BRASIL
www.campbellsci.com.br • vendas@campbellsci.com.br

Campbell Scientific Spain, S. L.
Avda. Pompeu Fabra 7-9, local 1
08024 Barcelona
SPAIN
www.campbellsci.es • info@campbellsci.es

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