Delta Tau Power PMAC

Communications Library Manual

Issue:	1.2
Date:	17th March 2015

	NAME	DATE	SIGNATURE
Prepared by	Philip Taylor, Observatory Sciences Ltd.	21 March 2013	Philip B. Taylor
	Andrew Wilson, Observatory Sciences Ltd.	10 September 2014	Andrew Wilson
Checked by	Alastair Borrowman, Observatory Sciences Ltd	10 September 2014	A. J. Borrowman
Released by	Andrew Wilson, Observatory Sciences Ltd.	10 September 2014	Andrew Wilson

Revision History

Revision Number	Authors	Date	Reason for Issue / Description of Changes
1.0	PBT AAW	10 Sep 2014	Initial release
1.1	AAW	17 Feb 2015	Library release 1.1. Describe command line arguments for test applications.
1.2	AAW	17 Mar 2015	Library release 1.2.

TABLE OF CONTENTS

1 Scope

This document describes the installation and use of the Communications software library for the Delta Tau Power PMAC motion controller. It includes a description of the hardware and software environments with which the library can be used.

It does *not* include detailed descriptions of the individual functions provided by the library. These are provided in the form of Doxygen compatible documentation (HTML format files) which is automatically generated from the source code when the software is installed.

2 Reference Documents

[RD1] Power PMAC User's Manual. Delta Tau Document 050-PRPMAC-0U0. Dated March 25, 2014.

[RD2] Power PMAC Software Reference Manual. Delta Tau Document 050-PRPMAC-0S0. Dated May 23, 2014.

3 Introduction

The Power PMAC UMAC CPU is the most powerful and most flexible controller that Delta Tau presently offers. It is intended to be used in a 3U UMAC Rack, which is a modular rack format permitting the user to configure their rack with whatever I/O, servo control cards, MACRO cards, or any other accessory card Delta Tau offers for the UMAC Rack format, to build a totally user-customized system. The Power PMAC UMAC CPU can control up to 256 axes, whether through direct local control, or distributed control over a MACRO fibre optic ring, or over an EtherCat network.
It may be controlled through an Ethernet interface by writing commands and reading replies and status as text strings over an SSH protocol connection.

This document describes a communications software library written to monitor and control a Delta Tau Power PMAC over SSH. The library can be built and used under both Linux and Windows (Visual Studio C++) systems.

The Power PMAC Communications Library software module consists of a library providing a set of functions providing control and status for both the Delta Tau Power PMAC controller itself and individual axes. The software distributed includes a lower level SSH driver library which performs basic I/O to the Power PMAC. The software has been written to be compatible with a variety of computer platforms (OS and development environments).

Two test applications are also provided which can be used to communicate with the Power PMAC from a terminal shell, and can be used to verify the SSH driver is functional and that the Power PMAC is responding as expected.

4 Requirements

This section details the hardware and software requirements for building and using the Power PMAC Communications Library software.

4.1 Hardware Requirements

The following hardware is required to use the communications library:

- Standard PC with an Ethernet port.
- Power PMAC present on the network. For operation with the EPICS device support code the Power PMAC should be configured to have a static IP address and this address must be specified in the initial library call which connects to the hardware.

  See [RD1] for details of how to configure the Power PMAC IP address.

4.2 Software Requirements

The following software must be installed to build and run the communications library:

- libssh2 library. This is a general purpose, low-level library providing support for the SSH communications protocol. It is freely available for most platforms under an Open Source software licence. See installation instructions below. Pre-built libssh2 binaries are provided for use with Windows.

5 Supported Platforms

The communications library has been built from source and tested under the following software environments.

5.1 Linux Operating Systems

The library has been built and tested under the CentOS operating system. The standard Linux build system (make) is used, with a Makefile provided. The version of software utilities and libraries used is shown in the table below.

Note that CentOS is effectively identical to the corresponding Red Hat Enterprise Linux release.

O/S version	Linux Kernel	gcc compiler	make utility	libssh2
CentOS 6.5	2.6.32-431	4.4.7	3.81	1.4.3

5.2 Windows Operating Systems

O/S version	Development Environment	libssh2
Windows 7	Microsoft Visual Studio C++ 2010 (version 10.0)	1.4.3

6 Source Code

Source code is provided for the library. The table below summarizes the files provided that are relevant to both Linux and Windows platforms. Additional platform-specific files are described in the appropriate section below.

Filename	**Description**
powerPMACControl	Header file (.h) and C++ code (.cpp). This implements the main PowerPMACControl class which provides the library functions to communicate with the Power PMAC.
libssh2Driver	Header file (.h) and C++ code (.cpp). Provides a set of functions to connect, read and write using the SSH protocol. This is a wrapper for the low-level libssh2 library.
powerPMACShell	C++ code (.cpp). This application provides a simple command shell for a Power PMAC. This enables any command to be sent to a Power PMAC from a remote terminal.
testPowerPMACcontrolLib	C++ code (.cpp). This command-line test application enables testing of any of the functions in the Power PMAC control library. Simple terminal based menus provide a way to select the function to be used, with interactive input prompted for required parameters.
argParser	Header file (.h) and C++ code (.cpp). Provides common functions enabling the test programs to accept command line arguments for the Power PMAC connection parameters as described in Section 9.
Doxyfile	Configuration file, used with the Doxygen utility to automatically create documentation.
multi_thread_test	Header file (.h) and C++ code (.cpp). A utility for testing multithreading performance of the library on Linux operating systems, which starts several concurrent threads executing library functions and prints the outcome of those functions.
isConnected_test	C++ code (.cpp). A small test program which calls powerPMACcontrol_isConnected() at 1 Hz in order to test the effect on this function of disrupting the SSH connection.

7 Linux Installation

This section covers building and installation of the communications library for the Power PMAC on a Linux operating system.

7.1 libssh2 Install from Source

The lower level driver class provided (called libssh2Driver) requires an external library to be installed on the system to handle SSH encryption and connections. The Open Source software library libssh2 (http://www.libssh2.org) should be downloaded and installed. The version of libssh2 that was used during development and testing is 1.4.3 and it is recommended that this version be used. The following section illustrates building libssh2 version 1.4.3 from source.

First download a compressed tar archive (tar.gz) file containing the source code, using a download link provided on the libssh2 website, or from a command line download using wget:

```
$ wget http://www.libssh2.org/download/libssh2-1.4.3.tar.gz
```

Uncompress and unpack the archive to a suitable location, configure, build and install:

```
$ gunzip libssh2-1.4.3.tar.gz
$ tar -xvf libssh2-1.4.3.tar
$ cd libssh2-1.4.3
$./configure
$ make
$ su -
# make install
```

The libraries are by default installed in /usr/local/lib. Note that installation in this default location requires root access.

It may be necessary to install some extra packages (OpenSSL and Libgcrypt libraries). On Debian this can be done as follows:

```
$ apt-get install build-essential
$ apt-get install openssl
$ apt-get install libgcrypt-dev
```

Note that a libssh2 Debian package already exists but the current package version is not compatible with this module.

7.1.1 libssh2 Install using Linux Package Manager

The libssh2 library could be installed directly on the Linux PC using the appropriate package manager utility (yum on CentOS or dpkg on Debian). This is *not recommended* as the installed libssh2 will be the version provided by the package repository and this will probably be incompatible with the Power PMAC communications library.

7.2 Communications Library Build and Install

The released package is provided as either a Zip or Gzip archive file. First create a suitable top-level directory and place the archive file in this directory. After extracting the archive, the files will be placed in a directory named powerPMAC_ssh_1.0 within the current directory. Either extract the Zip archive using

```
$ unzip powerPMAC_ssh_1.0.zip
```

or uncompress and unpack the Gzip archive using

```
$ gunzip powerPMAC_ssh_1.0.tar.gz
$ tar -xvf powerPMAC_ssh_1.0.tar
```

Enter the directory containing the extracted files:

```
$ cd powerPMAC_ssh_1.0
```

In the main directory you will see the files as described in the table in Section 6 above. In addition, a Makefile is provided to enable building under Linux using the standard 'make' utility.

Before building the application it is necessary to check the locations of the libssh2 include file and library. The supplied Makefile assumes the defaults, which are /usr/local/include and /usr/local/lib. If libssh2 has been installed in a different location, then the Makefile will need to be edited to reflect this.

Simple type 'make' from the command line to build the Power PMAC communication library and the two test applications. The library and test application executables will be created in the main directory. Two library files will be created: the static (.a file) and shared library (.so file).

If you wish to install the library and include files into a different (system) area, then issue the command 'make install'. The default install location is /usr/local, so the command will have to be issued as the superuser. This will install the library (.a and .so) and .h files in the lib and include directory below that defined as INSTALL_DIR in the Makefile. Edit the Makefile definition if you wish to install to a different location to the specified default (/usr/local).

8 Windows Installation

This section covers building and installation of the communications library for the Power PMAC on a Windows 7 operating system, using Visual Studio C++ 2010.

8.1 Files provided

The released package is provided as a Zip archive file. The files should be extracted from this archive and placed into a suitable directory.
In the top-level directory you will see the files as described in the table in Section 6 above. The following folders are additionally provided to enable building under Windows using Visual Studio C++.

Folder name	**Description**
lilbssh2	DLLs and include files for libssh2. See Section 8.2 below.
msvc	Contains Visual Studio project files for Visual Studio 2010 in the subfolder named 10.0.

8.2 Using libssh2

The Power PMAC communications library uses the libssh2 library to provide connections using the SSH protocol. Building libssh2 from source using Visual Studio can be a complex procedure and is not described here. Only a limited set of libssh2 pre-built Windows binaries are available on the Internet, and the required version (1.4.3) is not currently provided.

To avoid having to build libssh2 locally, prebuilt DLLs and include file for libssh2 V1.4.3 are provided for the convenience of Windows users. The files are provided in the folder libssh2. Sub-folders are:

- dll: DLLs and import library (.lib) files for libssh2. Note that 3 DLLs are required to use libssh2 (libssh2, libeay32, zlib1)
- include: libssh2 header file (libssh2.h)

These files are used by Visual Studio during builds. The compiler and linker will search for these files in the location in which they are supplied, so if they are not moved no additional configuration is required.

However these files may be moved to a different location if desired: for example it might be convenient if the DLL and library files are located in a folder on the Windows standard search path for DLLs (e.g. the Windows system directory). In this case, Windows environment variables are used to define the location of these files, as follows:

SSH2_INCLUDE	The folder containing the file libssh2.h.
SSH2_LIB	The folder containing the DLL and associated import library (.lib) files.

If the libssh2 files are moved, these environment variables should be defined appropriately before Visual Studio is started. Instructions for this operation are given in the Appendix.

8.3 Communications Library Build and Install

Start Visual Studio C++ and select the desired 'solution' file, called libPowerPMACcontrol. This will be found in the folder .\msvc\10.0. The solution file contains 3 projects:

1. Project libPowerPMACcontrol_Shared. This project builds the Power PMAC communications library as a DLL shared library. The output binary file is libPowerPMACcontrol.dll.
2. Project PowerPMACShell. This project builds the Power PMAC command shell test application. The output executable file is PowerPMACShell.exe.
3. Project testPowerPMACcontrolLib. This project builds the Power PMAC communications library test application. The output executable file is testPowerPMACcontrolLib.exe.

Using Visual Studio C++, each project is built by selecting the "Build" option from the pull-down menu. The Visual Studio Output pane should show successful builds. The DLL should be built first, before the executables.

9 Running the Test Applications

Once the system has successfully built, the test applications can be used to ensure a connection to the Power PMAC can be made and that commands are correctly executed.

The test application executables are located in the main directory on Linux, or on Windows in the folder .\bin\msvc-10.0.

Under Windows, running the test applications requires the DLLs to be in your DLL search path. The DLLs are: PowerPMACcontrolLib, libssh2, libeay32 and zlib1. This may require modifying the Path environment variable to add the paths to the DLL location folder(s). Please see the Appendix for a guide to editing environment variables.

The test applications are as follows:

1. PowerPMACShell. This is a simple console application which provides a command prompt at which any Power PMAC command can be sent as a string. The reply received (if any) from the power PMAC is displayed. Type "quit" to exit.
2. testPowerPMACcontrolLib. This is a simple console application which allows all functions in the Power PMAC communications library to be tested. Two terminal-based menus are displayed, selected at the initial prompt: 1 for Controller-oriented functions and 2 for axis/Co-ordinate system oriented functions. The desired function from that menu is selected by entering the appropriate number. Any required parameters are prompted for.

   Hit <Enter> to re-display the menu after a function test. Exit the program by entering your selection as 0.

Both of these applications accept the following command line arguments:

Argument	**Description**	**Default value**
-ip *IP_ADDRESS*	Attempt to connect to the Power PMAC at IP address *IP_ADDRESS*.	192.168.0.48
-user *USERNAME*	Log in to the Power PMAC with user name *USERNAME*.	root
-passw *PASSWORD*	Login password for user *USERNAME* on the Power PMAC.	deltatau
-port *SSH_PORT*	Attempt to connect to port *PORT* on the Power PMAC.	22
-nominus2	When this argument is used, communication with the Power PMAC will be started with the command "gpascii". Otherwise the command "gpascii -2" will be used.	"gpascii -2" will be sent

10 Detailed Library Documentation

Detailed documentation for the library, including a description of all methods in the class PowerPMACcontrol, can be automatically generated by running the Doxygen utility (under Linux or Windows). The generated documentation files, in html format, will be created in the folder called html. The file index.html contains the main page.

Appendix: Editing Windows Environment Variables

This section describes how to add an environment variable in Windows 7, e.g. variable named Path.

1. To open the Environment Variables dialog:
    a. Click on the Start menu, right-click on Computer and select Properties. The System window opens.
    b. From the menu on the left-hand side, select Advanced System Settings. This opens the System Properties dialog at the Advanced tab.
    c. From the bottom of this dialog, click the button labelled Environment Variables to open the Environment Variables dialog.
    d. This dialog contains two lists. Changes will be made in the upper list, User variables for <your username>.
2. If the required variable is already present in this list, append the new path to any that it already contains.
    a. Select the variable from the list and click 'Edit…'.
    b. Append the desired path to the Variable value: separate it from any existing paths using a semicolon. For example, to add the path C:\newpath\ to an environment variable already containing the path C:\existingpath\, the variable value field should read C:\existingpath\;C:\newpath\
    c. Click OK.
3. If the required variable is not already present in this list, click 'New…' and type the name in Variable name and the path in Variable value. Click OK.
4. Click OK in the Environment Variables dialog to save the changes. The changes will take effect in Visual Studio the next time it is launched.