Delta Tau Power PMAC

EPICS Device Support Manual

Issue:	Draft 1.0
Date:	8th February 2013

	NAME	DATE	SIGNATURE
Prepared by	Alan Greer, Observatory Sciences Ltd.	8 February 2013	
Checked by	Philip Taylor, Observatory Sciences Ltd.	13 February 2013	
Released by			

TABLE OF CONTENTS

1 Scope

This document describes the installation and use of EPICS device support code for the Delta Tau Power PMAC motion controller. 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 his or her 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 fiber optic ring, or over an EtherCat network. It can be controlled through an Ethernet interface by opening an SSH connection and this document describes the EPICS device support software written to monitor and control the unit over SSH.

2 Reference Documents

[RD1] IOC Application Developer's Guide (EPICS 3.14.12), *Marty Kraimer et al.*
[RD2] EPICS R3.14 Channel Access Reference Manual, *Jeffrey O. Hill.*
[RD3] Power PMAC Software Reference Manual, *Delta Tau.*
[RD4] asynDriver: Asynchronous Driver Support, *Marty Kraimer, Eric Norum and Mark Rivers.*
[RD5] EPICS Motor Record and related software, *Tim Mooney, Joe Sullivan, Ron Sluiter.*

3 Introduction

The Power PMAC EPICS software module contains a low level SSH driver, asyn driver and device support code for the Delta Tau Power PMAC motion controller. The software has been written to closely match the already existing PMAC EPICS support code where possible and to provide the standard motor record interface as used by many installations.

A small test application is present within the module that 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. The module also contains a test application with EPICS database files and startup scripts to quickly get started with a test EPICS IOC. There are some EDM screens that can be used to interact with the motor records and a very simple set of CSS BOY files to provide a basic interface using CSS.

4 Requirements

This section details the hardware and software requirements for using the Power PMAC EPICS device support code.

4.1 Hardware Requirements

The following hardware is required to run the device support code:

- 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 start-up script.

4.2 Software Requirements

The following software must be installed to run the driver and device support code:

- Linux operating system. The device support code has been written using general asyn, motor and EPICS base function calls and methods and as a result will be fully functional on any system that can compile the required versions of EPICS base and the asynDriver and motor modules. The device support code was developed on Debian 6 (32 bit) but has also been tested on CentOS 5.7 (32 bit) and CentOS 6.3 (32 bit and 64 bit).
- EPICS Base (version 3.14.12 or later)
- EPICS module ‘AsynDriver’ (version 4.17 or later)
- EPICS module ‘motor’ (version 6.7.1 or later).
- MSI extension for EPICS (version 5 or later). Required if building the supplied test application
- EDM to run engineering screens, if they are required.
- CSS BOY to run the opi screens, if they are required.
- libssh2 library (not an EPICS module, see installation instructions below).

5 Installation Under Linux

This section covers installation of the device support code for the Power PMAC on a Linux operating system.

5.1 libssh2 Installation

The lower level driver class provided by the EPICS module requires that an external library be installed on the system to handle SSH encryption and connections. The library libssh2 (http://www.libssh2.org) should be downloaded and installed. The version that the driver has been built against is 2-1.4.3.

Either download using the link on the libssh2 website, or from a command line download using wget:

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

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 that location requires root access.

On the Debian 6 operating system it was first necessary to install some extra packages:

```
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.

5.2 EPICS Base and Modules

The Power PMAC module has been developed against EPICS base version 3.14.12.1, asynDriver version 4.17 and motor version 6.7.1. These modules must be installed before attempting to compile the Power PMAC module.

EPICS base can be downloaded from http://www.aps.anl.gov/epics. The module has also been tested against the Debian EPICS package available from NSLS II (http://epics.nsls2.bnl.gov/debian) which can be installed using the Debian package management system.

```
apt-get install epics-dev
```

The asynDriver EPICS module can be downloaded from http://www.aps.anl.gov/epics/modules/soft/asyn and installation instructions can be found on the website, they are beyond the scope of this document.

The motor EPICS module can be downloaded from http://www.aps.anl.gov/bcda/synApps/motor and installation instructions can be found on the website, they are beyond the scope of this document.

5.3 Power PMAC Module Installation

Assuming the required software described above has been installed, unpack the Power PMAC EPICS module archive file and cd into the top directory.

```
gunzip powerPMAC_V1-0.tar.gz
tar -xvf powerPMAC_V1-0.tar
cd powerPMAC_V1-0
```

In here you will see the following files and directories:

Directory	Description
powerPMACApp	This directory contains the driver and device support code, a template database for control and EDM and BOY screen files.
powerPMACTestApp	This directory contains a test application. A substitutions file sets up a system configured for a Power PMAC with eight axes.
configure	The configuration directory.
iocBoot	The boot directory for the test application. It contains the startup script that defines the IP address of the Power PMAC as well as the username and password. It also sets up the motor record code (see below).
Makefile	Top-level Make file to build all the software
runGUI	This is a bash script to start the EDM application and open the main EDM screen.
req	This file contains the save/restore file that could be used for the motor records. This file is not required for the test application as it does not depend on or use the EPICS save/ restore module.

Before building the application it is necessary to define the installation locations of the EPICS installation, asyn and motor modules, and a link to the libssh2 library. cd into the configure directory and edit the RELEASE file. Update the lines that define the location of the asyn and motor installations to point to wherever these modules are installed on your system. For example

```
ASYN=/usr/lib/epics/asyn-4.18
MOTOR=/usr/lib/epics/motorR6-7-1
```

Also update the line that defines the location of your EPICS installation. For example

```
EPICS_BASE=/usr/lib/epics
```

Save any changes and exit. cd back up to the top level and then into the application source directory.

```
cd ..
cd powerPmacApp/src
```

It is also necessary to edit the Makefile in that source directory to ensure it points to the correct installation location of libssh2. In the Makefile there is a variable ssh2_DIR that should be set to the location of the installed library file. For example

```
ssh2_DIR = /usr/local/lib
```

Finally the host architecture must be defined before the build can commence. From a terminal (assumed to be running the Bash shell) enter the following line:

```
export EPICS_HOST_ARCH=linux-x86
```

There is a test application supplied with the device support code. If this is to be used then the startup script should be altered. cd into the iocBoot/iocpowerPMAC directory from the top level.

```
cd iocBoot/iocpowerPMAC
```

Edit the st.cmd file and update the line that configures the asyn Power PMAC port with the IP address of the Power PMAC unit on your network. For example

```
drvAsynPowerPMACPortConfigure("SSH1","10.2.2.76","root",
"deltatau","0","0","0")
```

Now cd back to the top level and type ‘make’ to build the device support code and the test application. The build should complete with no errors.

6 Running the Test Application

Once the system has successfully compiled the test application should be executed to ensure a connection to the Power PMAC is completed.

6.1 Running the EPICS Database

To run the IOC cd into the iocBoot/iocpowerPMAC directory from the top level directory and execute the st.cmd script.

```
cd iocBoot/iocpowerPMAC
./st.cmd
```

It may be necessary to change the permissions of the bash script to make it executable. Use a chmod command from the command line to achieve this.

```
chmod 755 st.cmd
```

Below is an example of the output generated when the system is started on Debian 6.

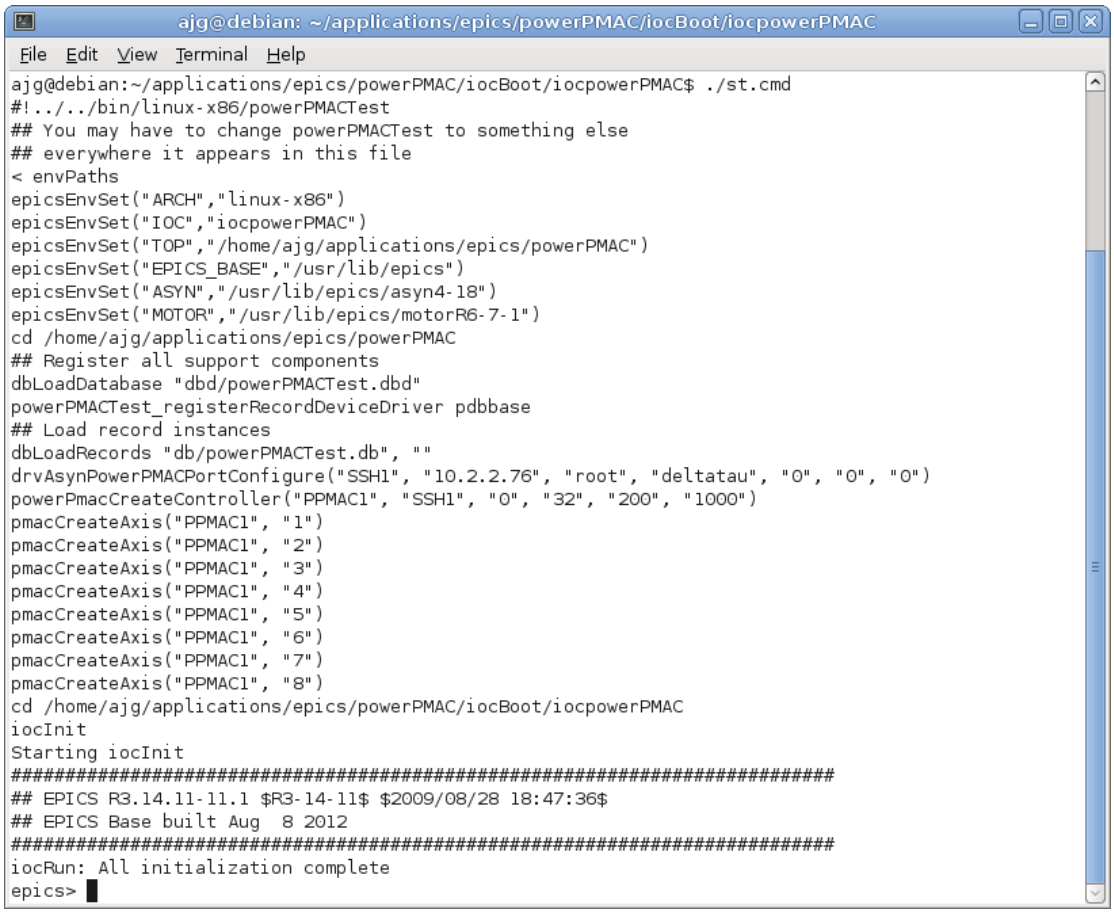

Figure 1 IOC output during start-up.

Once the start-up has completed and the records loaded the Power PMAC can be controlled by setting various records present in the database. The main record name and device port macros are set during the build procedure and can be changed by editing the substitutions file present in the test application database directory.

6.2 Running the EDM Screens

To allow easy testing some EDM engineering screens have been added; these can be started on Linux by executing the script called runGUI present in the top-level directory. The screens can be converted to other formats (http://www.aps.anl.gov/epics/) if required.

Once started the user is presented with the main screen. The test application contains some records to verify the connection status and firmware version of the Power PMAC device as well as the motor records for driving the individual axes.

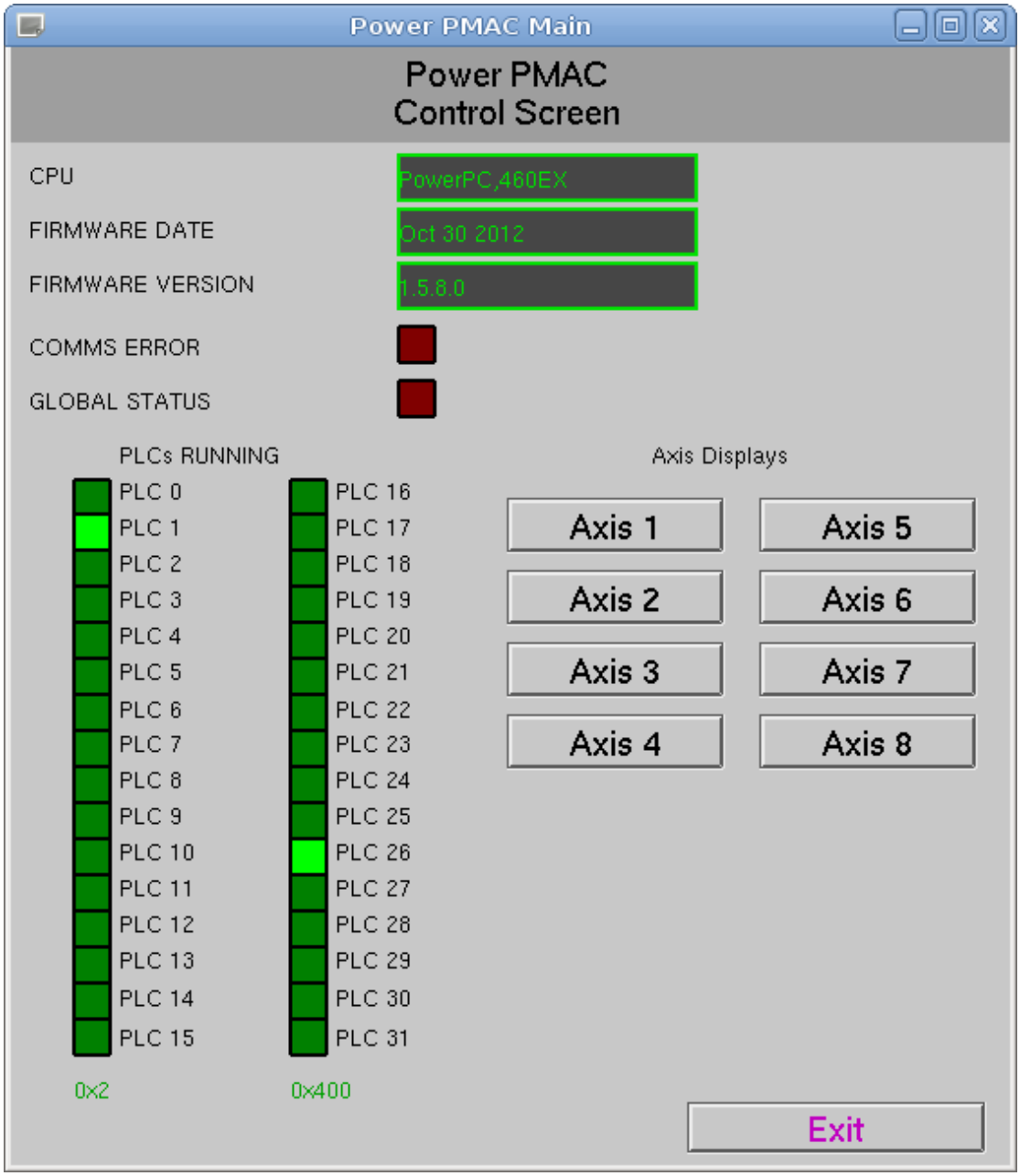

Figure 2 Main EDM screen.

The screen presents the following indicators and buttons:

- CPU. The Power PMAC is queried for its CPU type and the version; the information returned is displayed here.
- Firmware Date. The Power PMAC is queried for the date of the current firmware version; the information returned is displayed here.
- Firmware Version. The Power PMAC is queried for the current firmware version; the information returned is displayed here.
- Comms Error. If communications with the Power PMAC is lost then this indicator will light up. Whilst this indicator is lit no other information

presented regarding the Power PMAC can be considered up to date. If the connection is recovered then this indicator will turn off.

- Global Status. This indicator will light up if global faults are reported by the Power PMAC.
- PLCs Running. These indicators display the currently running PLCs on the Power PMAC device.
- Axis 1 to 8. Click on any of these buttons to open the motor specific screen for the corresponding axis.
- Exit. Click on this to close the screen.

Clicking on the axis buttons opens the motor specific screens for the corresponding axis. The main motor control and status screens are shown below:

Figure 3 Motor axis control screen.

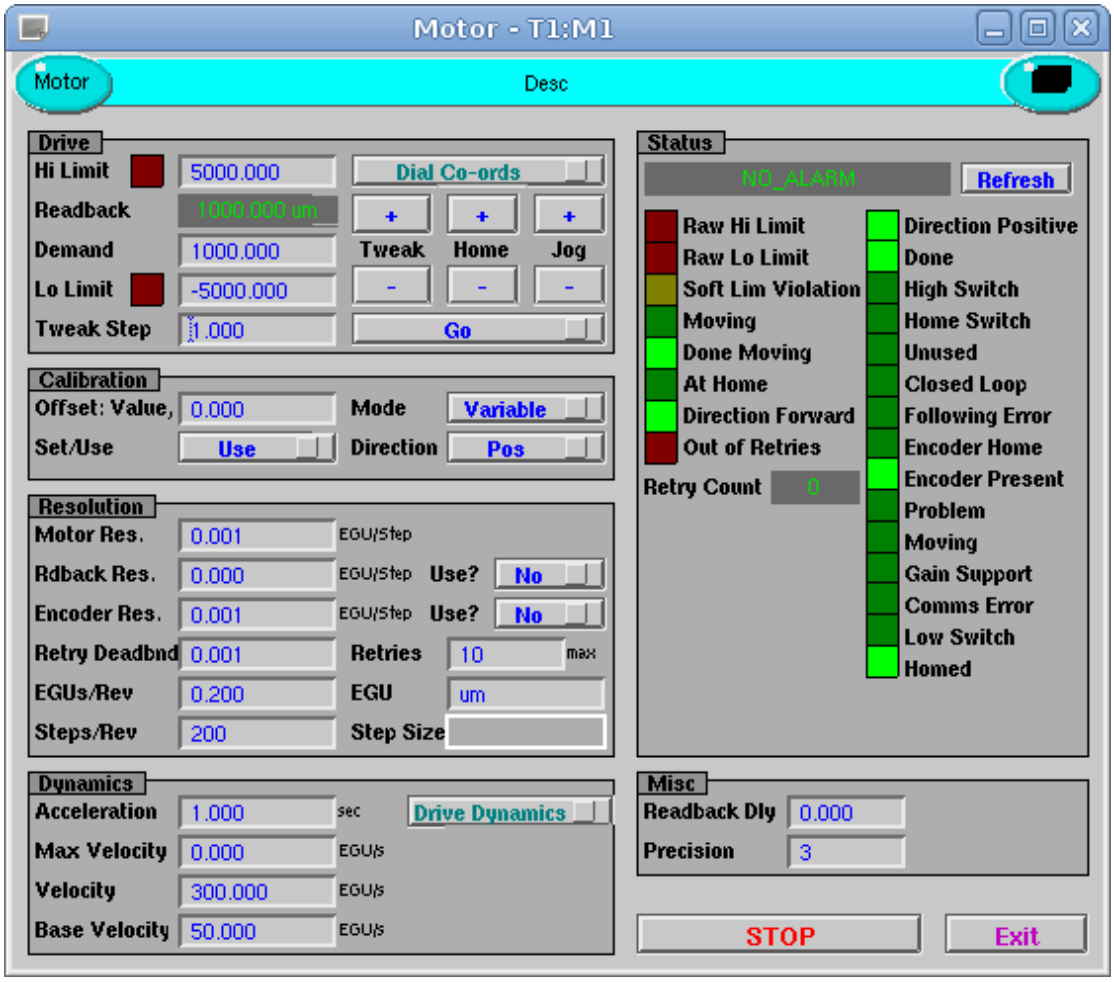

Figure 4 Motor axis status screen.

The motor axis control screen provides complete control of the EPICS motor record for the corresponding axis, including absolute and relative moves, jogging (constant velocity moves) and homing. Many of the motor record process variables can be set from this screen and its sub-screeens. The status screen provides some additional information in a concise format and allows many standard input variables to be set.

6.3 Running the CSS BOY Screens

The module provides some basic control screens that have been developed for use with the CSS/BOY display manager. Note that these screens should be considered as draft screens.

The screens were developed using the version of CSS provided by NSLS-II (http://cs-studio.sourceforge.net/nsls2/nsls2.html). The installation and operation of CSS is beyond the scope of this document, please visit the website for further information and installation/operation instructions.

There are two screens provided, one for the Power PMAC device and one for individual axis control.

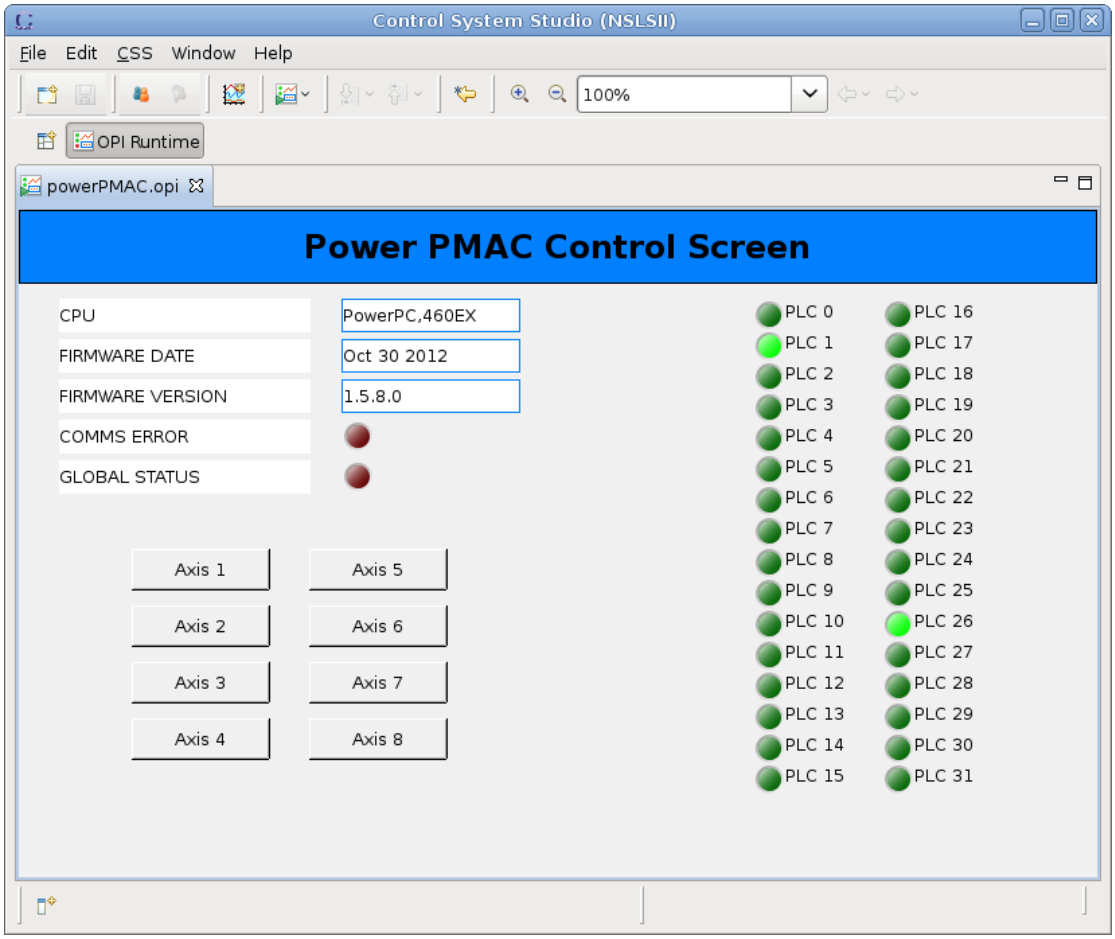

Figure 5 Main CSS BOY screen.

The main Power PMAC screen provided is very similar to that described above for the EDM application, the CPU, firmware date and firmware version are provided as well as indicators for the communications status, global hardware status and individual PLC execution status. Buttons are present to open the individual axis control screens when clicked.

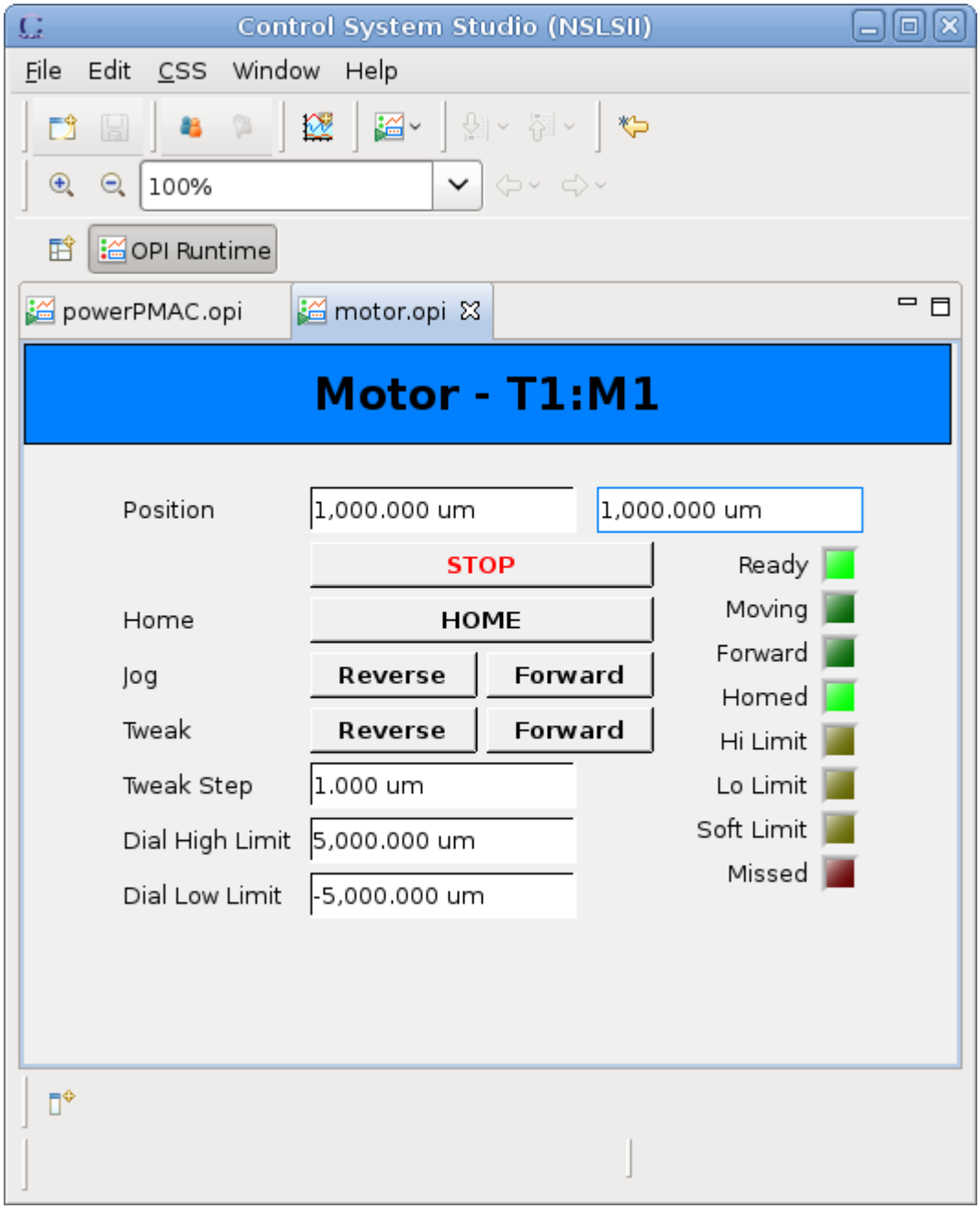

Figure 6 CSS BOY axis control screen.

The individual axis control screen shown above provides simple control and status monitoring of a single axis. The operator can move the axis (absolute or relative moves), or jog the axis in either direction (constant velocity moves). The axis can also be homed and stopped. Software limits can also be set from this screen. The status indicators show whether the axis is stopped and ready or moving, the direction of travel, whether the axis has been homed and the limit status. The missed indicator lights up if an axis has repeatedly attempted to reach a position to within its dead-band limit but failed. For a more detailed explanation see [RD5].

7 Development of the Device Support Code

This section describes in more detail the main files/classes involved in the Power PMAC EPICS module. These files can be located in the source directory of the Power PMAC application.

7.1 SSH Driver Class

Filenames	sshDriver.h sshDriver.cpp
Location	${TOP}/powerPMACApp/src
Methods	SSHDriver setUsername setPassword connectSSH setBlocking flush write read disconnectSSH ~SSHDriver

The SSH driver class provides a wrapper around the libssh2 library and presents a simple interface for establishing connections to the Power PMAC and write/read/flush operations.

Method	SSHDriver
Parameters	Host - Host name or IP of device.
Return	N/A
Description	Constructor for the driver class. Initializes internal variables.

Method	setUsername
Parameters	Username - The username for the SSH connection.
Return	SSHDriverStatus.
Description	Setter for the username.

Method	setPassword
Parameters	Password - The password for the SSH connection.
Return	SSHDriverStatus.
Description	Setter for the password. If this method is not called then authentication using keys is attempted when connecting.

Method	connectSSH
Parameters	None.
Return	SSHDriverStatus.
Description	Attempt to create a connection and authorize the username with the password or by keys. Once a connection has been established a dumb terminal is created.

Method	setBlocking
Parameters	Blocking - 0 for non-blocking, 1 for blocking.
Return	SSHDriverStatus.
Description	This is a private method and will always be set to 0 for the Power PMAC module to avoid blocking read calls.

Method	flush
Parameters	None.
Return	SSHDriverStatus.
Description	Attempt to flush the connection. Also performs a read to ensure no bytes are hanging around for the next read.

Method	write
Parameters	Buffer - The ascii buffer to write. BufferSize - The number of bytes to write (size of buffer). BytesWritten - The actual number of bytes that were written. Timeout - The number of milliseconds to wait before timing out.
Return	SSHDriverStatus.
Description	Write a message to the connection. The echoed message is then read back from the connection so that the next read does not contain these extra bytes.

Method	read
Parameters	Buffer - A string buffer to hold the data that has been read. BufferSize - The maximum number of bytes to read (the size of the buffer). BytesRead - The actual number of bytes read. ReadTerm - A terminator to use as a check for the end of the message. Timeout - The number of milliseconds to wait before timing out.
Return	SSHDriverStatus.
Description	Read data from the connected channel. The read method will continue to read data from the channel until either the specified terminator is read or the timeout is reached.

Method	disconnectSSH
Parameters	None.
Return	SSHDriverStatus.
Description	Close the connection.

Method	~SSHDriver
Parameters	None.
Return	N/A
Description	Destructor.

7.2 Asyn SSH Port Driver

Filenames	drvAsynPowerPMACPort.h drvAsynPowerPMACPort.cpp drvAsynPowerPMACPort.dbd
Location	${TOP}/powerPMACApp/src
Functions	closeConnection asynCommonReport cleanup connectIt asynCommonConnect asynCommonDisconnect writeIt readIt flushIt sshCleanup drvAsynPowerPMACPortConfigure

These functions implement an Asyn Octet interface, allowing the higher level motor record device support to connect, lock and communicate with the driver and the Power PMAC hardware. This driver code also handles loss of connection and reconnection, keeping the interface clean for the higher level.

Function	closeConnection
Parameters	pasynUser - Pointer to asyn user structure. ssh - Pointer to SSH controller structure. why - String reason for connection close.
Return	void
Description	Closes the connection. Calls disconnect on the driver class if the object exists. Then destroys and cleans up the driver.

Function	asynCommonReport
Parameters	drvPvt - Void pointer to the SSH controller structure. fp - Pointer to a FILE structure. details - The level of detail to provide in the report.
Return	void
Description	Prints a report to the file specified.

Function	cleanup
Parameters	arg - Void pointer to the SSH controller structure.
Return	void
Description	Locks the asyn port, then disconnects and destroys the driver.

Function	connectIt
Parameters	drvPvt - Void pointer to the SSH controller structure. pasynUser - Pointer to asyn user structure.
Return	asynStatus
Description	Creates a new SSH driver object. Sets the username and password according to supplied parameters in the SSH controller structure and then attempts a connection to the device. If the connection is

	successful then it starts the Power PMAC gpascii communications application ready for execution of motor commands.

Function	asynCommonConnect
Parameters	drvPvt - Void pointer to the SSH controller structure. pasynUser - Pointer to asyn user structure.
Return	asynStatus
Description	Calls connectIt.

Function	asynCommonDisconnect
Parameters	drvPvt - Void pointer to the SSH controller structure. pasynUser - Pointer to asyn user structure.
Return	asynStatus
Description	Calls closeConnection.

Function	writeIt
Parameters	drvPvt - Void pointer to the SSH controller structure. pasynUser - Pointer to asyn user structure. data - ASCII string to write. numchars - Number of characters to write. nbytesTransferred - Number of characters (bytes) that have actually been written.
Return	asynStatus
Description	First checks if a connection exists. If not it calls connectIt. Then a call to the SSH driver write method is made.

Function	readIt
Parameters	drvPvt - Void pointer to the SSH controller structure. pasynUser - Pointer to asyn user structure. data - Buffer ready to accept the response string. maxchars - The maximum size of data (buffer size) that can be received. nbytesTransfered - Number of bytes that have actually been read. gotEom - Was an end of message character returned?
Return	asynStatus
Description	First checks if a connection exists. If not it calls connectIt. Then a call to the SSH driver read method is made and the returned bytes placed into the data buffer.

Function	flushIt
Parameters	drvPvt - Void pointer to the SSH controller structure. pasynUser - Pointer to asyn user structure.
Return	asynStatus
Description	Calls the flush method on the SSH driver.

Function	sshCleanup
Parameters	ssh - Pointer to the SSH controller structure.
Return	void
Description	Frees all memory allocated to the SSH controller structure.

Function	drvAsynPowerPMACPortConfigure
Parameters	portName - Name of the asyn port assigned to this driver. hostName - Name (or IP) of the device to connect to. userName - Username to use when establishing an SSH connection. password - Password to use when establishing an SSH connection. priority - The priority of the driver. 0 indicates epicsThreadPriorityMedium. noAutoConnect - 0 indicates asyn automatically connects the port. noProcessEos - 0.
Return	int
Description	This function is (indirectly) called from the EPICS IOC startup script entry, and allocates the SSH controller structure. It then populates the structure with the necessary information and registers the relevant functions with the asyn interface layers. Functions are registered with the common interface and the octet interface.

7.3 Power PMAC Controller Class

Filenames	powerPmacController.h powerPmacController.cpp
Location	${TOP}/powerPMACApp/src
Functions	powerPmacController ~powerPmacController writeInt32 writeFloat64 report getAxis poll pmacSetAxisScale pmacSetOpenLoopEncoderAxis lowLevelWriteRead lowLevelPortConnect debugFlow getGlobalStatus

This class extends the asynMotorController class, providing device support for the motor record. It is connected to the lower lever SSH port driver by specifying the relevant port name when created from the startup script. This class connects to the driver using the asyn OctetSyncIO interface, thus ensuring the port is locked while the write/read is taking place.

Method	powerPmacController
Parameters	portName - String name of this port, used by the motor record or other records to connect to this device. lowLevelPortName - String name of the low level asyn driver port to connect to.

	lowLevelPortAddress - Address of the low level driver port. numAxes - Number of axes to create for this controller. movingPollPeriod - Time in milliseconds between status polls whilst an axis is moving. idlePollPeriod - Time in milliseconds between status polls whilst no axes are moving.
Return	N/A
Description	Constructor for the class. Initializes internal variables, calls lowLevelPortConnect and starts the poller for status updates.

Method	~powerPmacController
Parameters	None.
Return	N/A
Description	Destructor.

Method	writeInt32
Parameters	pasynUser - Pointer to asyn user structure. value - Integer value to set.
Return	asynStatus
Description	Write an integer parameter into the parameter set.

Method	writeFloat64
Parameters	pasynUser - Pointer to asyn user structure. value - 64 bit float value to set.
Return	asynStatus
Description	Write a float parameter into the parameter set. Checks to see if the parameter is a low or high soft limit and if it is then writes the limit down to the hardware.

Method	report
Parameters	fp - Pointer to a FILE structure. level - The level of detail to provide in the report.
Return	void
Description	Prints a report to the file specified. Also prints a status line to the terminal.

Method	getAxis
Parameters	pasynUser - Pointer to asyn user structure. or axisNo - Integer axis number.
Return	A pointer to the axis object.
Description	These methods return a pointer to the corresponding axis object for this controller.

Method	poll
Parameters	None.
Return	asynStatus
Description	Polls the controller for status, rather than individual axes. Calls getGlobalStatus.

Method	pmacSetAxisScale
Parameters	axis - Integer axis number. scale - Integer scale
Return	asynStatus
Description	Sets an internal scale factor for the axis that increases the resolution in the motor record.

Method	pmacSetOpenLoopEncoderAxis
Parameters	axis - Integer axis number to setup with a different encoder. encoder_axis - Integer axis number to use as the encoder for this axis.
Return	asynStatus
Description	This method allows the axis number for the encoder read-back of a different axis to be set. It ensures the different axis number will be used for the encoder read-back. Both axes must have been created.

Method	lowLevelWriteRead
Parameters	command - String command to write. response - String response from controller.
Return	asynStatus
Description	Wrapper for asynOctetSyncIO write/read functions.

Method	lowLevelPortConnect
Parameters	port - String name of the driver port to connect to. addr - Address of the driver port to connect to. pasynUser - Pointer to asyn user structure. inputEos - String input EOS. Not used. outputEos - String output EOS. Not used.
Return	int
Description	Establishes communications with the device. Once connected some general information is requested and stored as parameters: CPU type, firmware version and date.

Method	debugFlow
Parameters	message - String debug message to print.
Return	void
Description	This method will print to screen the debug message if the debugFlag_ private member variable has been set to 1. The message is also sent to the asynPrint function.

Method	getGlobalStatus
Parameters	None.
Return	epicsUInt32
Description	Retrieves and stores the global status of the controller. This includes reading the device global status words, checking the connection and retrieving the running status of each PLC program on the device.

7.4 Power PMAC Axis Class

Filenames	powerPmacAxis.h powerPmacAxis.cpp
Location	${TOP}/powerPMACApp/src
Functions	powerPmacAxis ~powerPmacAxis move moveVelocity home stop poll setPosition getAxisStatus getAxisInitialStatus

This class extends the asynMotorAxis class, providing device support for the motor record. It is connected to the lower lever SSH port driver through the powerPmacController class which also manages the creation of these axis objects. This class provides useful motor methods (move, stop, etc).

Method	powerPmacAxis
Parameters	pC - Pointer to the powerPmacController object that this axis resides on. axisNo - The axis number of this axis.
Return	N/A
Description	Constructor. Initializes all internal variables. Forces a poller wakeup.

Method	~powerPmacAxis
Parameters	None.
Return	N/A
Description	Destructor.

Method	move
Parameters	position - The desired position of the axis. relative - Should the axis move relative to its current position or to an absolute position. min_velocity - Not used. max_velocity - The velocity at which the axis is moved. acceleration - The acceleration of the axis for the duration of the move.
Return	asynStatus
Description	Moves the axis either relative to its current position or to an absolute target position. Sets up the velocity and acceleration if necessary.

Method	moveVelocity
Parameters	min_velocity - Not used. max_velocity - The velocity at which the axis is moved. acceleration - The acceleration of the axis for the duration of the

	move.
Return	asynStatus
Description	Jogs the axis in either direction. Sets up the velocity and acceleration if necessary.

Method	home
Parameters	min_velocity - Not used. max_velocity - Not used. acceleration - Not used. forwards - Not used.
Return	asynStatus
Description	Issues the home command to the device for the axis ("HM").

Method	stop
Parameters	acceleration - Not used.
Return	asynStatus
Description	Stops the axis by issuing a "J/".

Method	poll
Parameters	moving - The moving state of the axis.
Return	asynStatus
Description	Polls an axis for status. Calls getAxisStatus.

Method	setPosition
Parameters	position - Position to set axis to.
Return	asynStatus
Description	Not used.

Method	getAxisStatus
Parameters	moving - The moving state of the axis set from this method.
Return	asynStatus
Description	Reads the status for this axis from the device. The current position, following error, enabled state and status words are read, parsed and the relevant parameters set accordingly.

Method	getAxisInitialStatus
Parameters	None.
Return	asynStatus
Description	Reads the initial status of the axis, including the high and low software limits, and the PID gain parameters.