Biclops Robotic Camera Positioning Mechanism

Specifications and User's Guide, Rev H, API V9.0

1 Introduction

1.1 Overview

The TRACLabs Biclops robotic sensor positioning mechanism is a multi-axis motion control platform for aiming single or multiple sensors. It is both compact and lightweight, consumes little power, and provides excellent position observability. The load mount is consistent with the standard ¼-20 tripod mount, including an alignment pin common on many digital and video cameras. All axes are under closed-loop digital control, with motion commands and status feedback through either a standard RS-232 or USB 2.0 full speed connection.

Biclops delivers nearly 2 N-m of torque to each axis, with a mechanical upper load limit of 4kg. Peak rotational speeds exceed 170°/sec. while drawing about 14 watts of power for both the motors and the controller and less than 2 watts quiescent. Mounted within the base of Biclops, the controller consists of an embedded servo motor controller and brushed servo amplifiers.

The command interface to the controller is a straightforward binary packet protocol as dictated by the controller chipset. When a command packet (command byte, checksum, and data) is sent it is immediately acknowledged by a status packet (status byte plush checksum). Each axis can be polled independently for position and motion parameters as well as quality of performance indications (e.g., ability of axis to maintain commanded position).

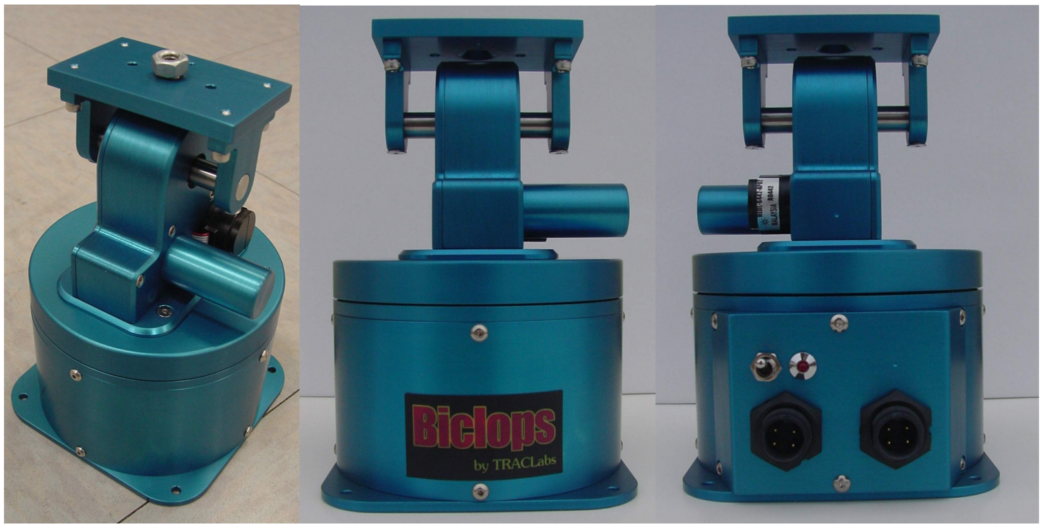


Figure 1 Biclops PT-M from various angles

1.2 General Performance Parameters & Physical Specifications

The following tables highlight the mechanical and electrical features of Biclops.

Overall Biclops Specifications:

Maximum camera/lens mass	4Kg
Camera position	center (-M and –S), 4", 5", 6", 7" (-S only)
mass	1.0 Kg
Overall dimensions	6" (H) x 4" (W) x 4.125" (D)
Mounting hole pattern	four 6-32 THRU holes, on 3.25" square w/ ¼-20 x ¼"DP tapped hole in center of base
Power consumption:	
24v motor power	1.5W quiescent, 10W typical, 20W max

Biclops Robotic Camera Positioning Mechanism

Joint Specifications (no load):

	Pan	*Tilt*	*Roll*
Range of motion	±175°(±180°)	±90°(±60°)	±90°(±60°)
Max. speed	170°/sec	170°/sec	170°/sec
Max. acceleration	3000°/sec^2	3000°/sec^2	3000°/sec^2
Resolution	1.8 arc-min	1.8 arc-min	1.8 arc-min
(Encoder Ratio)	(12000 counts/rev)	(12000 counts/rev)	(12000 counts/rev)

The remainder of this document discusses three separate aspects of the Biclops system: its mechanical configuration and operation, control electronics, and software interface. The mechanical and electronics discussions are intended to familiarize the reader with the inner workings of the mechanism to help understand the system's capabilities and limitations. The software description is where users will focus the majority of their attention for controlling Biclops.

2 Mechanical System

Biclops consists of two or three, or four degrees of freedom; namely pan, tilt, and roll (optional), and verge (optional). Each axis is fully decoupled from the others so they may be controlled independently or coordinated to fit the application. The details of these axes are discussed below.

2.1 Pan Axis

Pan makes up the majority Biclops' volume. It houses the mechanism for rotating about the vertical (Z) axis and also contains the control electronics. The pan base provides the mounting flange for attaching Biclops to another surface such as a table top or to a standard tripod mount.

Pan uses a DC brushed motor with a 76:1 zero-backlash gearhead to drive a timing belt transmission with a 14:60 gear ratio, resulting in nearly 2N-m of torque about the axis of rotation. The pan rotor rides on a 1.5"OD thin-section bearing. This bearing is pre-loaded with a gothic-arch race profile which supports moment loads as well as radial and axial loads. This bearing has a 1" bore, used for passing the upper axis wires through to the control PCB. This bore is also used for passing custom sensor cabling from the tilt axis to the back of the Biclops unit.

Feedback for the pan axis is provided through a 500 line encoder (2000 counts/rev) driven off a separate timing belt from the motor drive. With a 6:1 pulley ratio, the axis has a total of 12000 encoder counts/revolution, giving a positional granularity of 0.03 degrees (1.8 arc-min). Since the encoder transmission does not incur any loading (other than the encoder itself), the angular measurement is accurate to within one encoder count. Combining this with factory calibration, this configuration results in an accuracy of ±½ encoder count, or 0.9 arc-min. Since this is the only encoder used for feedback, belt flexibility and residual gearhead backlash are compensated for in the control loop, although these deficiencies can play havoc with controller stability if control parameters are not chosen carefully. Recommendations for choosing these parameters are offered in the software section that follows.The pan axis has a physical stop, limiting the range of motion to ±175°. An optional extended range hard limit allows a full ±180° rotation.

2.2 Tilt Axis

The tilt axis utilizes the same drive design as pan so its torque and speed are equivalent to pan. New with Biclops RevH, the tilt drive mechanism has been turned inside out relative to previous versions so that all of the sensitive components are protectect from environmental dust and moisture. An adapter plate joins tilt to pan while simultaneously sealing off the hole in the pan center needed to route control wiring from the tilt axis.

Range of motion has been extended to ±90°, but a more limited ±60° range is possible with the substitution of a limit screw on the tilt axis. There is also an option to route custom cabling through the tilt axis internally to the back of the unit, thus minimizing problems associated with external cabling that may interfere with device operation.

2.3 Roll Axis

Roll is identical to tilt mechanically and sits between pan and tilt. A mating adapter between roll and tilt orients the tilt axis perpendicular to the roll axis motion. When roll is installed, the pan axis is rotated 90 degrees clockwise so tilt faces forward as it would when only pan and tilt are present. Note that roll performance must be derated relative to the tilt axis due to roll having to support the extra mass of the tilt axis in addition to the customer load.

2.4 Verge Axis

Vergence is sometimes necessary when customers have a pair of sensors that need to be articulated relative to each other, either in a coordinated or independent fashion. Typically, such a requirement could be accommodated with two separate Biclops bases, but it is often only required to vary the vergence angle between the sensors while maintaining an identical tilt angle. To this end, a dedicated vergence mechanism reduces the volume and complexity of the actuator system by providing a dedicated vergence stage.

Although the Biclops electronics and software support driving a vergence stage, TRACLabs does not offer a vergence solution. However, the customer may attach their own vergence mechanism.

2.5 Orientation Conventions

Biclops axis rotation follows the right hand rule convention, with the x-axis pointing forward, the y-axis to the right, and the z-axis point down. Therefore, pan positive rotation is to the right, tilt positive rotation is looking up, and roll positive rotation is to the right. This convention is consistent with JAUS (Joint Architecture for Unmanned Systems).

3 Control Electronics

The Biclops controller board fits entirely within the pan base (see Figure 2). It measures 3.6” in diameter and has the following major components:

- PMD MC2120 (or 2140 for units with roll or verge) Control Processor (CP)
- PMD MC2100 I/O Processor (IP)
- Two (2) ST Micro L298P Quad H-Bridges
- FTDI FT232R USB controller
- A Sipex Maxim MAX232A RS232 to TTL transceiver
- A Texas Instruments PT78000W DC/DC converter
- A 40MHz oscillator
- Two or four axis connectors (Hirose DF11 type and JST PH type)
- Mini USB type B connector and Hirose DF11 connector for RS232
- green power LED

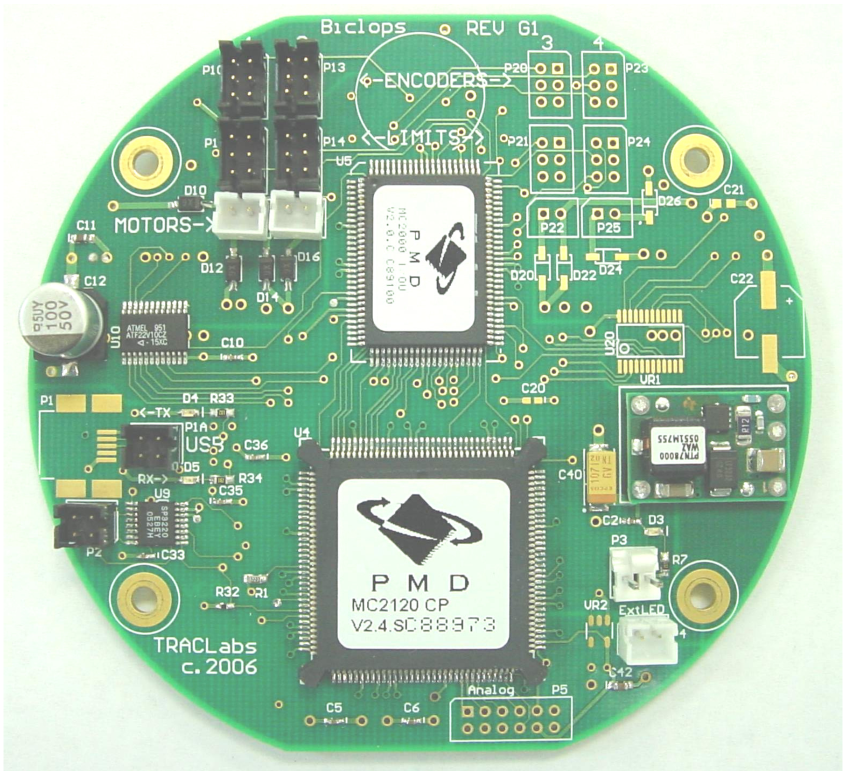


Figure 2 Biclops PMD Controller

Biclops uses the Performance Motion Device's (PMD) MC21x0 Navigator series of motion control chips to move and coordinate the individual axes. The pair of integrated circuits combine to issue control signals to the motors, monitor position feedback, and communicate with an external host computer for command and status information relay.

PMD provides a technical specification that elaborates on the functional and electrical characteristics of the chipset.Complete details about the internal operations of the MC2140 chipset may be found in PMD's Navigator users guide. The Navigator user's guide presents a detailed description of how the chipset operates, motor control operational modes and the different types of interfaces available to communicate with the chipset.

To control Biclops, it must be connected to an computer through either a USB or RS-232 cable. A single connector on the back of the unit provides both USB and RS-232 signals, with the custom data cable providing the connection to a standard USB-A connector or DB-9F connector. Upon powerup, Biclops is configured to communicate at 9600 bits-per-second (bps), with one start bit, one stop bit, no parity. Once the initial communication handshaking has occurred between Biclops and the computer, the data rate can be changed to any rate that the PMD chipset is designed to handle, namely 1200, 2400, 9600, 19200, 57600, 115200, 250000, or 416667 bps. Note that the RS-232 interface has a maximum data rate of 115200bps.

The motor amplifiers may be disabled through the setting of software controls as described in the software section. They are dual Full H-bridges capable of supplying up to 2A continuously for each H-bridge, although the Biclops motors only require about 10% of this capacity. Pan and tilt are driven off of one of the chips while roll is driven off of the other. These chips are housed in power SO20 surface mount packages that are designed to dissipate thermal energy through a metal slug beneath the IC into the PCB. Additionally, the amplifiers are mounted on the bottom of the controller PCB, so the IC package is in direct contact with the Biclops base plate, which acts as a giant heat sink providing ample thermal dissipation.

4 Software Interface

The Biclops controller command format is defined by the Navigator chipset. PMD provides a programmers reference (visit http://www.pmdcorp.com), which details the individual commands and their formats. To simplify control of Biclops and to assist the user in quickly integrating Biclops into an application, every Biclops ships with a software API that insulates the user from much of the low-level commands. At the highest level, the Biclops API offers a suite of commonly-used class methods wrapped up in a C++ class, along with lower-level support classes. Since the API is delivered with source code, the user is free to interface with the chipset at any level. Moreover, sample applications are provided to demonstrate the use of the API and allow quick control of Biclops.

At the top level, the Biclops API consists of the Biclops C++ class that defines Biclops specific constants and helper methods for dealing with issues specific to Biclops. The Biclops class controls each axis through instances of the PMDAxisControl class. The PMDAxisControl class provides the bulk of the publicly visible methods for controlling every aspect of an axis.

The higher level classes make no assumptions about the actual hardware interface to Biclops since that detail is handled at a lower level. Therefore, ideally Biclops could be connected to the host computer through a serial port, parallel port, USB, IEEE 1394 or other interface. Biclops currently only has an RS232 or USB connector, so the serial interface is the only valid choice presently.

The Biclops API provides several basic types of operations. Other than the necessary class constructor, there exist methods for initializing the controller and communications, initializing the axes (homing), moving the axes either independently or jointly, and requesting status about each joint.

Interfacing to Biclops is best illustrated by a minimal code example. The following description identifies the salient features of this application and explains their significance in the control of Biclops.

First create an instance of the top Biclops API, vis.

```
Biclops biclops;
```

The next step is to establish a communication channel to the controller, initialize the controller, and define the characteristics of each axis. To provide the greatest amount of flexibility in doing each of these steps, the Biclops API uses a configuration file to represent these aspects. To avoid having to explain all of the file format details, suffice it to say that the file format represents controller characteristics by using token-value pairs. The values loaded from the file may be overridden at almost any point during initialization and control, but its use supports the definition of a default configuration without requiring a significant amount of user code. The user application's only task is to provide a file path in a call to the Biclops class' `Initialize()` method. This method loads the configuration file, opens the communication channel, and verifies communication with the control chipset. Upon successful completion of this method (returns a value of `true`), the application may find the home positions of each axis through the HomeAxes() method. The first parameter of this method lets the application choose the axes to be controlled (represented as a bit mask).

```
if (biclops.Initialize(file)) {
    if (biclops.HomeAxes(axisMask)) {
```

Since homing can take up to a minute to perform, it is usually desireable to implement a task to inform the user that homing is still continuing. In this example, we simply wait for the homing process to complete by not specifying a wait time.

Upon successful homing completion, Biclops is ready to be moved. It is usually convenient to get a handle to each axis through the `GetAxis()` method. Using the axis handles, each axis can be manipulated independently. In the example application a simple move of each axis is executed by first getting the motion profile parameters currently set for the axis and then modifying the position of the profile, followed by a call to the axis `Move()` command. This sequence is illustrated with the following code segment:

```
// Get current state of axis profile parameters
PMDAxisControl::Profile panProfile;
panAxis->GetProfile(panProfile);

// Change the profile to move to a new location
panProfile.pos = 100.0/360.0;
panAxis->SetProfile(panProfile);

// Start the move
panAxis->Move();
```

With the above sequence, the pan axis will move to the desired position before returning from the `Move()` call. Each axis may be controlled independently in this way. Alternatively, the Biclops class provides some helper methods that allows multi-axis motion to be coordinated. An overloaded version of the `Move` method allows multiple axes to initiate their moves simultaneously. Another method, `MoveCoordinated()`, not only starts each axis at the same time, but ensures that all the axes end motion at the same time. In the example application, only the first version of `Move` is shown, viz,

```
biclops.Move(axisMask);
```

A complete example of the introductory concepts discussed above may be found in the file `BiclopsBareBones.cpp`.

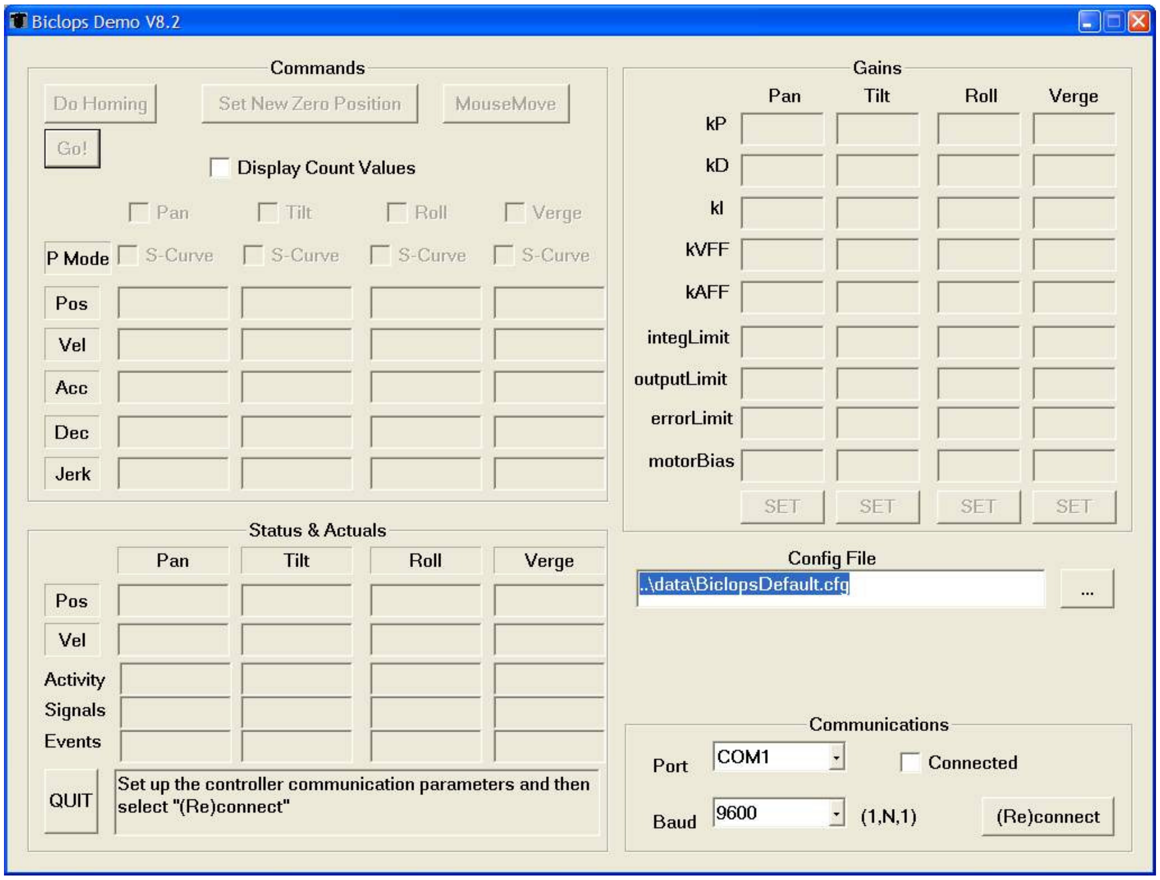
Biclops also comes with a Microsoft Windows-based GUI application for exercising some of the major Biclops features and gives the user the quickest and most intuitive access to controlling Biclops out of the box. It may be used to connect to Biclops (Reconnect), home one or more of the axes (Do Homing), and move the axes with the mouse (MouseMove). The figure below shows a snapshot of the interface. You can also adjust the gains profile parameters to tune the Biclops control parameters to suit the current load. This application is not meant to be a full-fledged GUI and so does not exercise many of the API's features.

To use this GUI, first verify that the configuration file you want to use is entered correctly in the config file edit box. The ellipsis box next to that field lets the user browse to the correct file.

With the correct file entered, clicking on the (Re)Connect button will cause the application to execute the Biclops::Initialize method with the given configuration file. Upon successful completion of this process, the active axes will be displayed in the Commands and Status sections. If it is desired to change baud parameters at any time, simply select a different baud in the Baud edit box and click on the (Re)Connect button.

Before any axis can be moved, it must be homed. To do this, ensure that the axes to home have their checkboxes active and then click on the DoHoming button. The status window will provide progress on the homing process and inform the user of the result.

At this point, any of the edit fields may be modified to change position of any active axis, or modify the control parameters or desired peak velocity or acceleration. Alternatively, clicking on the MouseMove button brings up a window that allows the user to drag the mouse (with the left mouse button depressed) to move both pan and tilt. Dragging with the right mouse button move tilt and roll.



Set up the controller communication parameters and then select "(Re)connect"

Config File: ..\data\BiclopsDefault.cfg

Port: COM1

Baud: 9600 (1,N,1)

Figure 3 Biclops GUI demo application

4.1 Biclops homing and calibration

Biclops uses incremental encoders to measure position. Consequently, upon power up, absolute position is not know until after a sequence of motions discovers signals at known absolute postions. Prior to revision H, axis homing required the axis to seek both mechanical limits and derive the zero position as the center between those limits. The process is highly repeatable and accurate, but requires the axis to move slowly to minimize mechanism wear and damage while due to hard limit collision. Unfortunately, this mode of homing can be time-consuming, taking a minute or more just to home pan and tilt.

Biclops revision H incorporates a home sensor and an encoder with index to help reduce homing time and avoid the requirement of driving the axis into its hard limits. Using these new signals, homing consists of searching for the zero-aligned index pulse that is coincident with the home sensor, resulting in complete pan/tilt homing in as little as 6 seconds.

Encoder index alignment to the axis zero position occurs during assembly. This process essentially consists of using a fixture to lock the axis in its zero position while the encoder is rotated to the index position and then secured to the axis-encoder transmission with a set screw. Ideally, this assembly process results in an index pulse that is aligned to within ±½ encoder pulse of the zero position. However, in practice, the calibration fixtures may be off by varying amounts due to inaccuracies in the fixture.

To eliminate this calibration error, we recommend that the customer execute a one-time post-assembly procedure that runs each axis through a series of motions to measure the difference between the true zero position and the index pulse. The resulting measurements can then be incorporated into the configuration file as a homing offset.

The application, `BiclopsRevHHomingCalibration`, establishes communication with Biclops and then executes the default hard-limit-seeking homing procedure to determine true zero. It then moves the axis around the zero position looking for the nearest index pulse, displaying a summary of index offsets at the end. The user can then edit the configuration file, substituting the homing offset entry (`homeOffset` in the file) for each axis with the new value.