

ASTROHUB AUTOMATION INTERFACE SPECIFICATION

RELEASE 1.206

Introduction

The AstroHub has been designed with application developers and script writers foremost in mind. A complete software interface has been developed and what follows documents the functions available. Installation of the AstroHub software included installation and registration of the astro_hub.ocx so it is immediately available to users.

At the moment there is no automation interface for the Stepper Plug-in since there are multiple devices that can be emulated and all have ASCOM drivers associated with them. A generic stepper interface can be added upon request but using one of the ASCOM drivers will likely suffice for most applications.

Note that there is also an ASCOM telescope hub included that is named astrohub.exe and that the OCX is named astro_hub.ocx (with the underscore).

Feel free to use the AstroHub Yahoo group for questions or requests about this interface and also for bug reports. We actively monitor and participate in this group so this will be the fastest way to get questions answered or problems resolved.

Release Notes

Version 1.205

Initial release. It works fine here.

Objects

There are presently 4 objects in astro_hub.ocx and all have been registered with Windows during the AstroHub installation. A sample script folder was installed for you as well and contains simple scripts to see the basic syntax of instantiated an object and using it. The interface is a fully compliant with Microsoft's Component Object Model.

The Objects included are:

- Mount
- Guider
- Control Port
- TelescopeStatus

MOUNT

Description

The AstroHub implements a "snoop processor" which monitors telescope mount communication and acts on or responds to certain special commands. The Mount Processor can be used for several functions including turning mount power on or off. At the moment, the Mount Processor this is a powerful function that is not being utilized very heavily. It is a simple matter to add to the Mount Processor firmware to implement additional mount functions. Please contact Aquest, Inc. technical support with any requests for added function. The interface below will be extended as more ideas for use are submitted. The actual microprocessor used for this function is running at 20MHz and has a significant amount of program and non-volatile memory available in anticipation of expanding then function provided. There is a lot of spare resource in this microprocessor at the moment and we would like to take better advantage of it. Give us some ideas!

The Mount Processor listens to all mount communications (such as "Go To", pulse guiding, or any other mount communications) and responds to any that have a special prefix designed so only the Mount Processor understands and the mount itself ignores. All other traffic simply passes to and from the Mount Processor transparently.

Properties

<p>CommID Returns the Windows handle to the open serial port allowing use of any Windows API call for the port.</p>	<p>Integer (Read Only)</p>
<p>Description Returns the string "Mount Processor"<CR> used for verifying that the connection to the Mount Processor is made and operational.</p>	<p>WideString (Read Only)</p>
<p>DirectComm Used by the ASCOM Telescope Hub (astrohub.exe) to define whether the hub has the connection to the telescope driver open or not. DirectComm = TRUE indicates that the driver connection to the telescope is closed and the Mount interface should self-open the serial port and handle communications directly. If DirectComm is false, the interface uses an instance of AstroHub.Telescope to communicate via the ASCOM CommandString method of the hub. This is all handled automatically in the interface so that simply accessing a property will handle the communication with Mount Processor using the appropriate path. It is essential that DirectComm be set properly for this functionality to work.</p>	<p>Boolean</p>
<p>DriverVersion Returns the version of astro_hub.ocx file.</p>	<p>WideString (Read Only)</p>
<p>FirmwareVersion Returns the version of the Mount Processor's resident firmware</p>	<p>WideString (Read Only)</p>
<p>HubInterface Used by the ASCOM Telescope Hub (astrohub.exe) to tell the astro_hub.ocx mount interface what its name is so the interface can create an instance of the hub object and communicate via the hub's CommandString method.</p>	<p>WideString</p>

<p>IsConnected Valid only when DirectComm = TRUE and indicates that the interface has the serial port open and can communicate as required.</p>	<p>Boolean (Read Only)</p>
<p>MountPort The AstroHub creates several COM ports and these are arbitrarily assigned by the operating system. Accessing this property will return the COM port that has been assigned to the Mount Processor.</p>	<p>Byte (Read Only)</p>
<p>PowerState Sets or returns the actual state of power applied to the mount. This property takes into consideration the Normally Open (NO) or Normally Closed (NC) mount power relay contact jumper as well as state of the physical relay in terms of whether it is energized or not. In other words, PowerState = TRUE means the mount is physically powered.</p>	<p>Boolean</p>
<p>Relay Sets or returns the state of the relay. Relay = TRUE means that power has been applied to the relay coil and it is energized (picked). Note that this doesn't necessarily mean the mount power is on as the mount power relay contact jumper may be in the Normally Closed (NC) position which would mean the relay has disconnected power from the mount. PowerState is the preferred method of controlling mount power</p>	<p>Boolean</p>
<p>RelayContacts Returns the position of the mount power relay contact jumper as a string of either "NO" (normally open) or "NC" (normally closed). See the Relay and PowerState properties above for the affect of this value.</p>	<p>WideString (Read Only)</p>
<p>StartState The firmware in the Mount Processor can be set to start with the relay energized or not. This persistent property (stored in the Mount Processor EEPROM) sets the state at startup. If StartState = TRUE the Mount Processor will energize (pick) the mount power relay when it starts executing its firmware.</p>	<p>Boolean</p>
<p>MinToShutdown Sets or returns the number of minutes until the mount power is automatically turned off. Setting this property to 0 minutes will force ArmShutdown to FALSE and disable automatic power off</p>	<p>Integer</p>
<p>ArmShutdown Sets or returns the flag to cause the built in timer to count down in minutes starting at the current MinToShutdown value. If ArmShutdown is set to FALSE while MinToShutdown is > 0, the countdown pauses until it is set to TRUE at which time the count will resume where it left off (in whole minutes).</p>	<p>Boolean</p>

Methods

AutoOpen(out Port: Byte; out Connected: Boolean)

The interface automatically determines the COM port the Mount Processor is on. Call this method to open the port and have returned the COM port number (Port), and the success of the operation (Connected).

AutoOpenBlind

Same as AutoOpen except that no parameters need be passed so as to easily be used in scripts. Read the MountPort and isConnected properties to determine the COM port and connection status.

ClosePort

Closes the COM port the Mount Processor is communicating on. Calling this method if the COM port is already closed has no effect.

OpenPort(PortNumber: Byte)

Attempts to manually open the Mount Processor COM port using the PortNumber value. Test the IsConnected property to assess the success of the operation.

Events

OnPowerStateChange(PwrState: Boolean)

This event fires whenever the power state of the mount changes. The event fires if the PowerState is changed directly or if the timed power off function turns power off.

OnShutdownTick(MinutesToGo: Integer)

If ArmShutdown is true, this event fires whenever a minute of the countdown elapses. This is useful for a UI to display a countdown of minutes remaining to power off .

GUIDER

Description

The base AstroHub includes a complete guiding system capable of accepting commands from a computer and driving any type of mount auto-guider input. The automation interface below allows for the guide outputs to be controlled from application software or scripts using high level commands as given. Guiding can also be accomplished using any application software in the typical way of sending guide bytes (where the low order bits denote direction) into the open serial port. In fact, a combination of the two may be used where pulseguide commands can overlay classical guiding as both proceed simultaneously.

Properties

<p>CommID Returns the Windows handle to the open serial port allowing use of any Windows API call for the port.</p>	<p>Integer (Read Only)</p>
<p>Description Returns the string "Guide Processor"<CR> used for verifying that the connection to the Guide Processor is made and operational.</p>	<p>WideString (Read Only)</p>
<p>DriverVersion Returns the version of astro_hub.ocx file.</p>	<p>WideString (Read Only)</p>
<p>FirmwareVersion Returns the version of the Guide Processor's resident firmware</p>	<p>WideString (Read Only)</p>
<p>GuidePort The AstroHub creates several COM ports and these are arbitrarily assigned by the operating system. Accessing this property will return the COM port that has been assigned to the Guider.</p>	<p>Byte (Read Only)</p>
<p>GuiderType The AstroHub will automatically switch its guider output to conform to various auto-guider electrical interfaces. This property is not normally used but is provided if software wishes to determine the type of mount connected. Returned values are: FALSE = "ST-4" compatible (e.g. driving a Meade LX-200) TRUE = Older style Gemini mounts Note: This property will not work if there isn't a ground connection between the AstroHub and the mount. A ground will typically be available if the mount is powered through the AstroHub and/or a serial connection is made between the AstroHub Mount/RS-232 connector and the mount's RS-232 input.</p>	<p>Boolean (Read Only)</p>
<p>IsConnected Indicates that the interface has the serial port open and can communicate as required.</p>	<p>Boolean (Read Only)</p>

Methods

AutoOpen(out Port: Byte; out Connected: Boolean)

The interface automatically determines the COM port the Guide Processor is on. Call this method to open the port and have returned the COM port number (Port), and the success of the operation (Connected).

AutoOpenBlind

Same as AutoOpen except that no parameters need be passed so as to easily be used in scripts. Read the GuidePort and isConnected properties to determine the COM port and connection status.

ClosePort

Closes the COM port the Guide Processor is communicating on. Calling this method if the COM port is already closed has no effect.

Move(Direction: Byte)

Starts a guide movement in one or two directions at once. The move continues until a Stop or StopAll command is received or a pulsed move on that axis completes. Dual-axis moves are allowed by the adding together of movement direction constants as follows:

N = 8
S = 4
E = 2
W = 1

Example: To move in a NE direction call Move(10);

OpenPort(PortNumber: Byte)

Attempts to manually open the Guide Processor COM port using the PortNumber value. Test the isConnected property to assess the success of the operation.

Pulse(Direction: Byte; PulseTime: Integer; Fire: Boolean)

Causes a pulsed movement to occur on a guider output for the duration specified in PulseTime expressed in milliseconds (range: 10 to 9999 ms).

Back to back commands with different durations are permitted. To start pulses at the same time send all but the last command with Fire = FALSE to "arm" an output and the last command with Fire=TRUE. The "arm" commands will be queued until a "fire" command is received and then all will fire at the same time. If back to back commands with the same direction are sent the first is effectively terminated and the second starts as a new pulse.

Stop(Direction: Byte)

Stops a guide movement in one direction only. Directions are as follows:

N = 8
S = 4
E = 2
W = 1

Note: a pulsed movement that is overlaid on a static move command will cause the static command to end when the pulse terminates.

StopAll

Immediately stop all guide movements in progress regardless of how they were initiated

Events

There are no events associated with the guide interface.

CONTROL PORT

Description

The AstroHub Control Port is a general purpose I/O card of which two may be used at the same time if desired. This object is a high level interface to the functions on the card including events for when certain input conditions occur.

There is also the ability to store and retrieve values from non-volatile memory on the Control Port. This memory is organized into Words (16 bits) and Longs (32 bits) for convenience.

Properties

<p>CommID Returns the Windows handle to the open serial port allowing use of any Windows API call for the port.</p>	<p>Integer (Read Only)</p>
<p>Description Returns the string "Control Processor"<CR> used for verifying that the connection to the Control Processor is made and operational.</p>	<p>WideString (Read Only)</p>
<p>DriverVersion Returns the version of astro_hub.ocx file.</p>	<p>WideString (Read Only)</p>
<p>FirmwareVersion Returns the version of the Control Processor's resident firmware</p>	<p>WideString (Read Only)</p>
<p>In1 Returns the current state of Input 1 on the Control Port TRUE= the input pin is asserted to ~2.5V or above.</p>	<p>Boolean (Read Only)</p>
<p>In2 Returns the current state of Input 2 on the Control Port TRUE= the input pin is asserted to ~2.5V or above.</p>	<p>Boolean (Read Only)</p>
<p>In3 Returns the current state of Input 3 on the Control Port TRUE= the input pin is asserted to ~2.5V or above.</p>	<p>Boolean (Read Only)</p>
<p>In4 Returns the current state of Input 4 on the Control Port TRUE= the input pin is asserted to ~2.5V or above.</p>	<p>Boolean (Read Only)</p>
<p>IsConnected Indicates that the interface has the serial port open and can communicate as required.</p>	<p>Boolean (Read Only)</p>
<p>MountVin Returns the value of the voltage read at mount power input of the AstroHub. Format of returned string is: xx.xV<CR></p>	<p>WideString (Read Only)</p>

<p>Out1 Sets or returns the state of Output 1 on the Control Port. Note: since the outputs are open collector outputs or "ground closures" (active low), setting the Outx property to TRUE= closed or ~0V on the output pin.</p>	Boolean
<p>Out2 Sets or returns the state of Output 2 on the Control Port. Note: since the outputs are open collector outputs or "ground closures" (active low), setting an Outx property to TRUE= closed or ~0V on the output pin.</p>	Boolean
<p>Out3 Sets or returns the state of Output 3 on the Control Port. Note: since the outputs are open collector outputs or "ground closures" (active low), setting an Outx property to TRUE= closed or ~0V on the output pin.</p>	Boolean
<p>Out4 Sets or returns the state of Output 4 on the Control Port. Note: since the outputs are open collector outputs or "ground closures" (active low), setting an Outx property to TRUE= closed or ~0V on the output pin.</p>	Boolean
<p>PollRate Sets or returns the rate at which the interface is to internally poll the four inputs and fire an event if any of them changes. The poll rate can be set in milliseconds in the range of 100ms to 9999ms. Setting PollRate=0 disables polling</p>	Integer
<p>RawVin Returns the value of the voltage read at the main power input of the AstroHub. Format of returned string is: xx.xV<CR> Meaningless if the AstroHub has been jumpered so main power is connected internally to the mount power input.</p>	WideString (Read Only)
<p>Relay Sets or returns the state of SPDT relay on the Control Port card. TRUE= the relay is energized (picked) so the Normally Open contact is connected to the Common.</p>	Boolean
<p>SocketNumber Returns the physical socket number (1 or 2) that the control port card is plugged into. This is useful if two control ports are installed in the same AstroHub.</p>	Byte (Read Only)

Methods

<p>ClosePort</p> <p>Closes the COM port the Control Processor is communicating on. Calling this method if the COM port is already closed has no effect.</p>
<p>GetInputs(var Inputs: Byte)</p> <p>Returns the state of all 4 input pins at once. Input 1 is the LSB</p>
<p>GetLongMemory(Location: Byte; var Value: Integer)</p> <p>Reads and returns the contents of user non-volatile memory location. Location is a value from 0 to 39. The result is a 4 byte long integer directly readable into Visual Basic, etc.</p>
<p>GetWordMemory(Location: Byte; var Value: Smallint)</p> <p>Reads and returns the contents of user non-volatile memory location. Location is a value from 0 to 39. The result is a 2 byte small integer directly readable into Visual Basic, etc.</p>
<p>OpenPort(PortNumber: Byte)</p> <p>Attempts to manually open the Control Processor COM port using the PortNumber value. Test the IsConnected property to assess the success of the operation.</p>
<p>OutputPulse(Output: Byte; TimeVal: Integer; Fire: Boolean)</p> <p>Causes a pulse to occur on an output pin for the duration specified in TimeVal expressed in milliseconds (range: 10 to 9999 ms). This command controls one output pin at a time and if the value of Output is not 1..5 the result is unpredictable. Output 5 is not connected to a physical pin but is a "virtual" output allowing use of the microprocessor for accurate timing of events.</p> <p>Back to back commands with different durations are permitted. To start pulses at the same time send all but the last command with Fire = FALSE to "arm" an output and the last command with Fire=TRUE. The "arm" commands will be queued until a "fire" command is received and then all will fire at the same time.</p> <p>When a pulse generated with this command is complete, an OnPulseOutputComplete event will be fired returning the value of the output whose pulse has just completed.</p>
<p>OutputPulseAsync(Output: Byte; TimeVal: Integer; Fire: Boolean)</p> <p>Same as OutPulse except no event is generated when the pulse is complete.</p>
<p>SetOutputs(Outputs: Byte)</p> <p>Sets the state of all 4 output pins at once. Output 1 is the LSB. See polarity information in the Outx property descriptions.</p>
<p>SetLongMemory(Location: Byte; Value: Integer)</p> <p>Writes a value into user a non-volatile memory location. Location is a value from 0 to 39. The value to be written is a 4 byte long integer that can come directly from Visual Basic, etc.</p>
<p>SetWordMemory(Location: Byte; Value: Smallint)</p> <p>Writes a value into user a non-volatile memory location. Location is a value from 0 to 39. The value to be written is a 2 byte small integer that can come directly from Visual Basic, etc.</p>

Events

OnFastInputChange(Inputs: Byte)

Inputs 1 and 2 are connected to the COM port associated with the specific Control Port card being used. Specifically, Input 1 is connected to the CTS signal and Input 2 is connected to DSR. Normal Windows API calls could be used to detect the change of these signals (using CommEvent calls) but use of this event eliminates the needs to do so.

Use of CommEvent is embedded in a thread in the Control Port interface code. All that is required is to catch this event and read the input value to determine whether CTS and/or DSR changed state:

Input Values:

- 0: Neither CTS or DSR asserted
- 1: Only CTS asserted
- 2: Only DSR asserted
- 3: Both CTS and DSR asserted

OnInputChange(Inputs: Byte)

This event fires if any input pin changes state. To use this function catch the event and read the value of the Inputs byte. Input 1 is the LSB.

OnPulseOutputComplete(Output: Byte)

See the OutputPulse method above. This is the event that fires when any single output pin's pulse has completed. If more than one output pin has been pulsed, an event will be fired when each one completes.

TELESCOPESTATUS

Description

ASCOM provides significant function for astronomy systems including the ability to use a software "hub" and connect several clients to the same telescope driver. Since every telescope driver or hub is required to return its current coordinates and other parameters whenever asked, it is possible for hubs to have their methods reentered and problems occur (like random lock-ups). To protect against this condition in the astrohub.exe ASCOM software hub, provision has been included to broadcast the relevant information twice a second to be passively received by any client that is listening and needing the information. The information is broadcast to all open windows through the Windows messaging system.

The TelescopeStatus object provides an interface so user applications and scripts do not need to deal with Windows messaging and can ask the interface for any of the information at any time without the risk of lock-ups or crashes of the hub. The interface software has been designed to not be effected if it is reentered. All a using application and script needs to do is read the properties below at any time. The values returned will be current to within 0.5 seconds.

Properties

Altitude Returns the mount's current altitude	Double (Read Only)
Azimuth Returns the mount's current azimuth	Double (Read Only)
Declination Returns the mount's current declination	Double (Read Only)
RightAscension Returns the mount's current right ascension	Double (Read Only)
SiderealTime Returns the mount's sidereal time value	Double (Read Only)
IsSlewing Returns TRUE if the mount is slewing. See the ASCOM documentation for the definition of slewing. It does not mean the mount is tracking or being moved at guide rates.	Boolean (Read Only)
IsConnected Indicates that the telescope driver the hub is connected to has opened the serial port to the mount.	Boolean (Read Only)
IsParked TRUE if the mount is parked	Boolean (Read Only)
IsTracking TRUE if the mount is tracking	Boolean (Read Only)

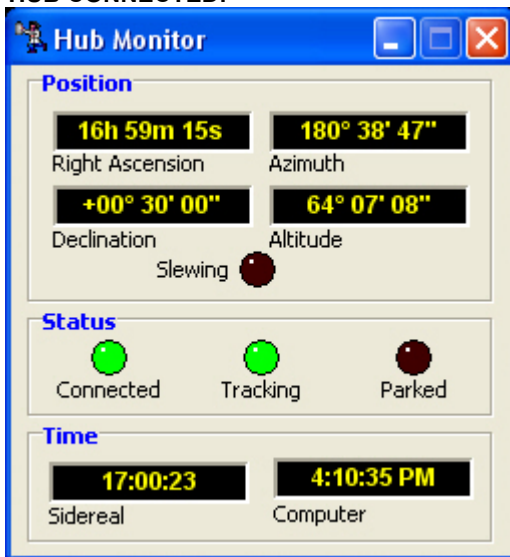
<p>StatusAge</p> <p>Indicates the number of seconds since the last message was received by the interface from the telescope hub. Normally this value will be 0 or 1. Anything higher means the telescope hub is no longer broadcasting messages. For example, it may have been shut down.</p>	<p>Integer (Read Only)</p>
--	--------------------------------

Methods

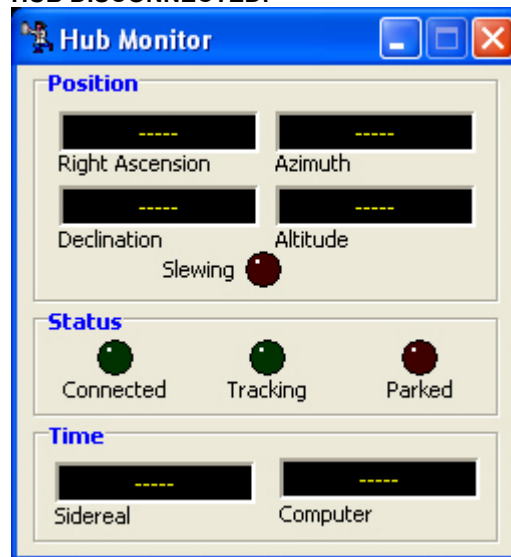
ShowMonitor

A monitor form with all the above properties on it can be displayed by a script or application software without any UI code being developed. Calling this method will pop up the following form which self updates the values as long as the telescope hub is running: Dashes will be shown otherwise.

HUB CONNECTED:



HUB DISCONNECTED:



HideMonitor

Hides the monitor.

Events

There are no events associated with the TelescopeStatus interface.