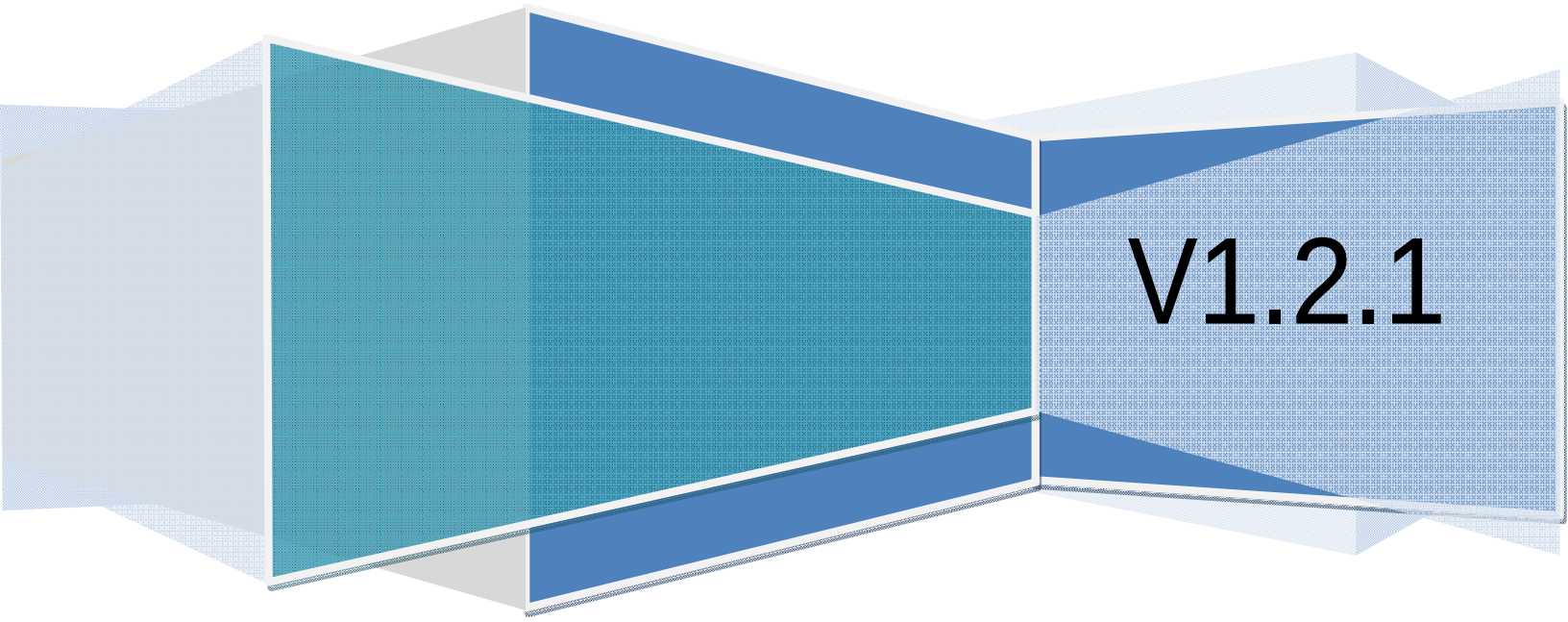
# User's Programming Manual

## Telescope Interface Module

RC Optical Systems Inc.

Flagstaff, AZ



# Table of Contents

## Introduction

The latest version of the TIM Control application (> Version 1.02) implements a programming interface based on Microsoft's Component Object Model (COM). COM is supported by all Microsoft operating systems and allows a simple object oriented way of communicating between binary software components. COM is largely language independent; it is thus possible to talk to the TIM Control Application from within a variety of programming languages. Visual Basic and Visual C# are one of the most popular examples, but Visual C++, Java and JavaScript support COM as well.

The reader might enjoy the following links for more detailed information about COM:

COM Fundamentals from Microsoft's MSDN:
*http://msdn.microsoft.com/en-us/library/ms690156%28v=vs.85%29.aspx*
Calling COM Components from .Net clients
*http://msdn.microsoft.com/library/ms973800.aspx*

This document will not cover the use of COM from within all the various programming languages since this would exceed the purpose of this document. Instead, programming examples will use Microsoft's Scripting Engine (WScript) and COM. WScript offers a simple way to access COM objects and the programmer does not need to worry about compiler specific COM access and memory management. Besides its simplicity, WScript is a good way to debug COM objects and perform simple testing and debugging. Last but not least, ASCOM uses WScript to access its COM objects for testing and enumerating purposes.

For more information about Microsoft's WScript please refer to the MSDN documentation:

WScript User's Guide
*http://msdn.microsoft.com/en-us/library/9bbdkx3k%28v=VS.85%29.aspx*

WScript Language Reference
*http://msdn.microsoft.com/en-us/library/98591fh7%28v=VS.85%29.aspx*

This manual assumes that the reader has been exposed to programming languages especially under the Windows environment. The user should feel comfortable using the command line and should as a minimum have some scripting experience. Understanding Microsoft's .Net and COM is beneficial.

## Using Microsoft's WScript

WScript is an interpreted and object oriented language. A script is evaluated at run-time and does not need to be compiled. WScript supports loose type variables which means that variables do not need to be explicitly declared. WScript supports automatic memory management and offers a comprehensive API access to Microsoft Windows specific functions.

### Running Scripts

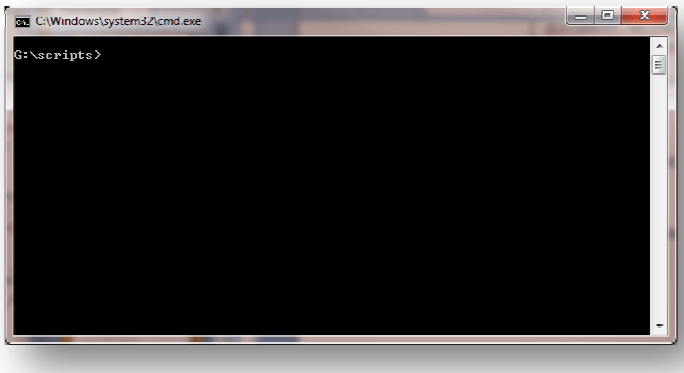
Let's start with a simple program that converts a temperature from Celsius to Fahrenheit:

```
function celsiusToFahrenheit(c)
{
	var f;
	f = c * 9.0/5 + 32;
	return f;
}

var celsius = 18;
var fahrenheit = celsiusToFahrenheit(celsius);
WScript.Echo(celsius + " degree Celsius is\r\n" + fahrenheit
+ " degree Fahrenheit");
```

Create a new document example1.js and save it into a folder that can be easily accessed via the command line (e.g. c:\scripts).

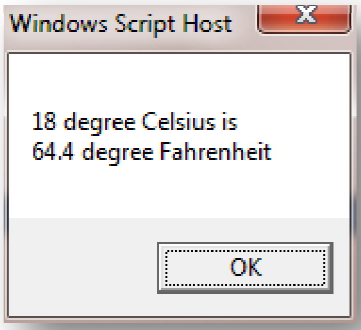
Open the command line (cmd.exe) and change into the folder where the jscript file was saved.




Simply execute the script by typing the filename into the command window.

```
example1.js [ENTER]
```

The script will show a little dialog with the output:




## Accessing a simple COM object

One of the strengths of Windows Scripting Host is its simple way of accessing COM objects. The following example shows how to access the Microsoft's Excel COM object to create a new workbook and enter a value into the first cell. The example requires that Microsoft Excel is installed.

```
// obtain the COM object for Excel
var objExcel = new ActiveXObject("Excel.Application");
var ExcelSheet = new ActiveXObject("Excel.Sheet");

// Make Excel visible.
ExcelSheet.Application.Visible = true;

// Place something in the first sheet
ExcelSheet.ActiveSheet.Cells(1,1).Value = "18";

// Insert formula to calculate Fahrenheit in cell 2
ExcelSheet.ActiveSheet.Cells(2,1).Value = "=$A$1*9/5+32";

// Save the sheet into our folder.
ExcelSheet.SaveAs("C:\\scripts\\TEST.XLS");

// Close Excel
ExcelSheet.Application.Quit();

[example2.js]
```

The call to `new ActiveXObject` instantiates a new COM object in this case of the application Excel. The COM object allows the programmer to place values into an Excel worksheet and save the sheet as text.xls.

# Controlling the Telescope Interface Module (TIM).

## Obtaining the TIM COM Object

The TIM Control application exposes the COM object TIMApi. The user can instantiate this COM object to control various functions of the TIM. A new application window will open for each instance of the COM object. This is useful in order to control more than one TIM unit attached to the same computer. If the user whishes to use the same COM object from various programs, it might be beneficial to use the ASCOM interface instead. Please refer to www.ascom-standards.org for further information.

```
// obtain TIM COM object
var F = new ActiveXObject("timcontrolapp.TIMApi");
var stdout = WScript.StdOut;
var stdin = WScript.StdIn;

stdout.WriteLine("Setting instance default");
// set default instance
F.setInstance("default");
// start sthe connection dialog
F.setupDialog();

stdout.WriteLine("Connecting to system");
F.System_connect();

stdout.WriteLine("Hit Enter to spin fan");
var str = stdin.ReadLine();
F.PWM_setFanPwm(200);

stdout.WriteLine("Hit Enter to exit");
var str = stdin.ReadLine();
F.PWM_setFanPwm(0);

F = null;
```

The first line creates the new COM object "timcontrolapp". The operating system will start a new instance of the TIM Control application which is set visible per default. Once connected, the programmer can use the API to control the TIM unit or interact with the GUI. The user program must be able to handle the user input via the GUI!

## Homing the Focuser / Rotator

The following example script shows how to home Focuser using the COM API. Although the code is for the focuser, the rotator can be homed in similar fashion. Use the Focuser_isHomed() / Rotator_isHomed() methods to determine when the Focuser/Rotator has been homed.

```
// obtain TIM COM object
var F = new ActiveXObject("timcontrolapp.TIMApi");
var stdout = WScript.StdOut;
var stdin = WScript.StdIn;

stdout.WriteLine("Setting instance default");
// set default instance
F.setInstance("default");
// issue connect and wait until connect
F.System_connect();
while (!F.System_isConnected())
{
	WScript.Sleep(500);
}
stdout.WriteLine("Connected to TIM unit");
stdout.WriteLine("Press Enter when ready");
stdin.ReadLine();
// home the focuser
var pos = F.Focuser_position();
stdout.WriteLine("The focuser position is: " + pos);
F.showFocuserPage();
F.Focuser_home();
stdout.WriteLine("Focuser is homing");
// wait until the focuser is homed
var homed = F.Focuser_isHomed();
while(!homed)
{
	homed = F.Focuser_isHomed();
}
stdout.WriteLine("The rotator finished homing");
stdout.WriteLine("Press Enter to exit");
stdin.ReadLine();
```

## Slewing the Focuser / Rotator

Slewing the Focuser / Rotator works in similar fashion like the previous homing example. Issue the slew command and call the method Rotator_isStopped() / Focuser_isStopped() until the method returns true. The following example slews the rotator and waits until the slew is finished. Slewing the focuser works in similar fashion.

```
// obtain TIM COM object
var F = new ActiveXObject("timcontrolapp.TIMApi");
var stdout = WScript.StdOut;
var stdin = WScript.StdIn;

stdout.WriteLine("Setting instance default");
// set default instance
F.setInstance("default");
// start sthe connection dialog
F.System_connect();
while (!F.System_isConnected())
{
	WScript.Sleep(500);
}
stdout.WriteLine("Connected to TIM unit");
stdout.WriteLine("Press Enter when ready");
stdin.ReadLine();
// start moving the rotator
var pos = F.Rotator_position();
stdout.WriteLine("The rotator position is: " + pos);
F.showRotatorPage();
F.Rotator_moveRelative(20);
stdout.WriteLine("Rotator is slewing");
// wait until the rotator is stopped
idle = F.Rotator_isStopped();
while (!idle)
{
	idle = F.Rotator_isStopped();
}
stdout.WriteLine("The rotator finished slewing");
stdout.WriteLine("Press Enter to exit");
stdin.ReadLine();
```

*Note: Like the GUI, the focuser will not listen to a slew command if the Focuser hit one of the hard-limit switches. This state should not occur on new style Focuser systems with optical limit switches which would be considered an error. On old style focuser systems this hitting a physical limit is normal.*

*Hitting a physical limit is displayed on the user GUI by a red position display. This error needs to be acknowledged by pressing the stop button (or sending the stop command) before another slew command can be issued! The user can determine the limit status using the limitTriggered() focuser API command.*

# COM API Reference

## System Methods

void setInstance(String instance);

*Sets the instance name of the TIM control application. Application parameters such as connection settings are stored per instance name. It is thus possible to use various instances (through various COM object instantiations) to connect to multiple TIM units. If only one TIM unit is attached to the computer, please use the "default" instance.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F = null;
```

void setupDialog();

*This command brings up the connection dialog to define the connection parameters such as network or COM port settings. Once the settings have been set, they will be stored according to the instance name. Once the connection parameters are stored, this dialog does not need to be called every time the COM object is created.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.setupDialog();
F = null;
```

void System_connect();

*This call establishes a connection to the TIM unit. Once connection parameter have been set, this method will use the connection parameters from previous sessions. Otherwise, connection parameters have to be established using the setupDialog command.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.System_connect();
F = null;
```

void System_disconnect();

*This call causes a disconnect from the TIM unit*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.System_disconnect();
F = null;
```

bool System_isConnected();

*This querying command determines the status of the connection and returns true when the connection is established.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.System_connect();
if (F.System_isConnected())
      WScript.Echo("System is connected");
F = null;
```

void setConnectionETH(String ip_addr, unsigned int port);

*This API function has been added in software version 1.2.1. This function sets the connection parameters for the Ethernet connection. The arguments must include the IP-address as String and the TCP port (usually 10001 for the TIM unit). It is important that the user sets the instance name (see setInstance(String name)).*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.setConnectionETH("192.168.1.10", 10001);
F.System_connect();
F.System_disconnect();
F = null;
```

void setConnectionUSB(unsigned int com_port);

*This API function has been added in software version 1.2.1. This function sets the connection parameters for the USB connection. The argument must include the virtual com port (VCP) as assigned by the USB driver. It is important that the user sets the instance name (see setInstance(String name)).*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.setConnectionUSB(12); // this sets the VCP COM12 port
F.System_connect();
F.System_disconnect();
F = null;
```

void showFocuserPage ();

*This command shows the GUI Focuser page. This command only affects the GUI and not the TIM unit.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.showFocuserPage();
```

void showFanPwmPage ();

*This command shows the GUI Fan and Temperature control page. This command only affects the GUI and not the TIM unit.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.showFanPwmPage();
```

void showRotatorPage ();

*This command shows the GUI Rotator page. This command only affects the GUI and not the TIM unit.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.showRotatorPage();
```

void showMirrrorCoverPage ();

*This command shows the GUI Mirror Cover page. This command only affects the GUI and not the TIM unit.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.showMirrorCoverPage();
```

## Focuser Methods

void Focuser_home ();

*This command issues a home command to the Focuser.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.Focuser_home();
F = null;
```

bool Focuser_isHomed ();

*This command returns true if the Focuser has been homed successfully.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
if (F.Focuser_isHomed())
	WScript.Echo("Focuser is homed");
F = null;
```

void Focuser_stop();

*This command stops the Focuser either during a slew or during homing. If the homing is interrupted with a stop command, the Focuser is considered not homed and the Focuser_isHomed() command will return false.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.Focuser_stop();
F = null;
```

bool Focuser_isStopped ();

*This command returns true if the Focuser is idle.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
if (F.Focuser_isStopped())
	WScript.Echo("Focuser is idle");
F = null;
```

void Focuser_ moveRelative (int counts);

*This command starts a movement of the Focuser relative to the current position in positive or negative counts.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.Focuser_moveRelative(1000);
F = null;
```

void Focuser_ moveAbsolute (int position);

*This command moves the Focuser to an absolute position.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.Focuser_moveAbsolute(800);
F = null;
```

int Focuser_ limitTriggered (void);

*This command returns the limit event / overcurrent status. The limit event status is cleared when a Focuser_stop() command is issued. This method returns the following possible values:*

```
#define  NO_LIMIT_HIT            0
#define  POS_OPT_LIMIT_HIT       +1
#define  NEG_OPT_LIMIT_HIT       -1
#define  POS_HARD_LIMIT_HIT      +2
#define  NEG_PHY_LIMIT_HIT       -2
#define  OVER_CURRENT_EVENT      3
```

void Focuser_ enableTempComp(bool enable);

*This command enables or disables the temperature compensation of the Focuser.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.Focuser_enableTempComp(false);
F = null;
```

void Focuser_setAutoTempComp();

*This call enables the automatic temperature compensation. Temperature compensation of the Focuser must be enabled before this command is called.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F. Focuser_enableTempComp(true);
F.Focuser_setAutoTempComp();
F = null;
```

void Focuser_setManTempComp();

*This call enables the manual temperature compensation. Temperature compensation of the Focuser must be enabled before this command is called.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F. Focuser_enableTempComp(true);
F.Focuser_setManTempComp();
F = null;
```

void Focuser_syncTempComp();

*This call synchronizes the temperature compensation in manual temperature compensation mode. Temperature compensation of the Focuser must be enabled before this method is called!*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F. Focuser_enableTempComp(true);
F.Focuser_setManTempComp();
F.Focuser_syncTempComp();
F = null;
```

bool Focuser_tempComp();

*This call returns true if the Focuser temperature compensation is enabled.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
var comp = F.Focuser_tempComp();
if (comp)
      WScript.Echo("The Focuser temp. comp. is enabled");
F = null;
```

int Focuser_position();

*This call returns the current Focuser position in counts*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
var position = F.Focuser_position();
WScript.Echo("The Focuser position is" + position);
F = null;
```

int Focuser_targetPosition();

*This call returns the last Focuser target position as commanded via a slew command or the GUI.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
var position = F.Focuser_targetPosition();
WScript.Echo("The Focuser target position is" + position);
F = null;
```

## Fan and Temperature Methods

double PWM_ambientTemperature();

*This call returns the temperature of the ambient temperature sensor. The temperature is returned in degrees Celsius independent of the GUI or system settings. Once the system is connected it takes a moment to query the system temperatures. A call to query temperature might return 0 until the temperatures are updated.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
var temp = F.PWM_ambientTemperature();
WScript.Echo("The ambient temperature is" + temp);
F = null;
```

double PWM_primaryTemperature();

*This call returns the temperature of the primary temperature sensor. The temperature is returned in degrees Celsius independent of the GUI or system settings. Once the system is connected it takes a moment to query the system temperatures. A call to query temperature might return 0 until the temperatures are updated.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
var temp = F.PWM_primaryTemperature();
WScript.Echo("The primary temperature is" + temp);
F = null;
```

double PWM_secondaryTemperature();

*This call returns the temperature of the secondary temperature sensor. The temperature is returned in degrees Celsius independent of the GUI or system settings. Once the system is connected it takes a moment to query the system temperatures. A call to query temperature might return 0 until the temperatures are updated.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
var temp = F.PWM_secondaryTemperature();
WScript.Echo("The secondary temperature is" + temp);
F = null;
```

void PWM_setFanPwm(unsigned int pwm);

*This command sets the pulse width modulation value of the fan. The value of the argument can range from 0 (0%) to 255(100%). Values greated than 255 are truncated.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.PWM_setFanPwm(127); // 50%
F = null;
```

void PWM_setAutoFan(bool e);

*This command enables or disables the auto-fan feature.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.PWM_setAutoFan(true);
F = null;
```

bool PWM_autoFanEnabled();

*This command returns the current status of the auto-fan feature.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
if (F.PWM_autoFanEnabled())
	F.PWM_setAutoFan(false);
F = null;
```

void PWM_setFanDeadBand(double db);

*This command sets dead-band value for the auto-fan feature. Please refer to the TIM user manual for further explanantion of the deadband value.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.PWM_setFanDeadband(5.0); // set deadband to 5 degrees
F = null;
```

void PWM_setSecPwm(unsigned int pwm);

*This command sets the pulse width modulation value of the secondary heater. The value of the argument can range from 0 (0%) to 255(100%). Values greated than 255 are truncated.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.PWM_setSecPwm(255); // 100%
F = null;
```

void PWM_setAuxPwm(unsigned int pwm);

*This command sets the pulse width modulation value of the auxiliary output. The value of the argument can range from 0 (0%) to 255(100%). Values greated than 255 are truncated.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.PWM_setAuxPwm(0); // off
F = null;
```

void PWM_setRcaPwm(unsigned int pwm);

*This command sets the pulse width modulation value of the fan. The value of the argument can range from 0 (0%) to 255(100%). Values greated than 255 are truncated.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.PWM_setRcaPwm(200);
F = null;
```

int PWM_getFanPwm();

*This command queries the current pulse width modulation setting of the fan as set by PWM_setFanPWM or the GUI.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
var pwm = 100.0 * F.PWM_getFanPwm()/ 255;
WScript.Echo("The Fan setting is" + pwm + "percent");
F = null;
```

int PWM_getSecPwm();

*This command queries the current pulse width modulation setting of the secndary heater as set by PWM_setSecPWM or the GUI.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
var pwm = 100.0 * F.PWM_getSecPwm()/ 255;
WScript.Echo("The Sec setting is" + pwm + "percent");
F = null;
```

int PWM_getAuxPwm();

*This command queries the current pulse width modulation setting of the auxiliary output as set by PWM_setAuxPWM or the GUI.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
var pwm = 100.0 * F.PWM_getAuxPwm()/ 255;
WScript.Echo("The AUX setting is" + pwm + "percent");
F = null;
```

int PWM_getRcaPwm();

*This command queries the current pulse width modulation setting of the RCA output as set by PWM_setRcaPWM or the GUI.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
var pwm = 100.0 * F.PWM_getRcaPwm()/ 255;
WScript.Echo("The RCA setting is" + pwm + "percent");
F = null;
```

## Rotator Methods

void Rotator_home();

*This command issues a home command to the Rotator.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.Rotator_home();
F = null;
```

bool Rotator_isHomed();

*This call returns true if the rotator is homed.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
var homed = F.Rotator_isHomed();
if (homed)
	WScript.Echo("Rotator is homed");
F = null;
```

void Rotator_stop();

*This command stops the Rotator movement. If the rotator is homing, a stop command will interrupt the homing and the rotator is considered not homed.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.Rotator_stop();
F = null;
```

bool Rotator_isStopped();

*This call returns true if the rotator is idle.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
var idle = F.Rotator_isStopped();
if (idle)
	WScript.Echo("Rotator is idle");
F = null;
```

void Rotator_moveRelative(double degrees);

*This command initiates a positive or negative relative move of the rotator in degrees. The internal rotator resolution is 0.1 degrees.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.Rotator_moveRelative(-100.2);
F = null;
```

void Rotator_moveAbsolute(double degrees);

*This command moves the rotator to an absolute position given in degrees. The internal rotator resolution is 0.1 degrees*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.Rotator_moveAbsolute(180.0);
F = null;
```

double Rotator_position();

*This call returns the current position of the rotator in degrees.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
var pos = Rotator_position();
WScript.Echo("The rotator position is: " + pos);
F = null;
```

double Rotator_targetPosition();

*This call returns the current target position of the rotator in degrees as commanded by a slew command or the GUI.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
var pos = Rotator_targetPosition();
WScript.Echo("The rotator target position is: " + pos);
F = null;
```

## Mirror Cover Methods

void MirrorCover_open();

*This method issues an open command to the mirror covers. Use the status query command to obtain the mirror cover status.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.MirrorCover_open();
F = null;
```

void MirrorCover_close();

*This method issues a close command to the mirror covers. Use the status query command to obtain the mirror cover status.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F. MirrorCover_close();
F = null;
```

void MirrorCover_abort ();

*This call aborts any mirror cover motion.*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
F.MirrorCover_abort();
F = null;
```

String MirrorCover_status();

*The method returns the current mirror cover status as string. Possible return values are:*

*"MID POSITION" - the mirror cover is between the open and closed position*
*"OPENING" - the mirror cover is opening but has not yet reached the open position*
*"CLOSING" - the mirror cover is closing but is not yet closed*
*"OPEN" - the mirror cover is in the open position*
*"CLOSED" - the mirror cover is in the closed position*

Example:

```
var F = new ActiveXObject("timcontrolapp.TIMApi");
F.setInstance("default");
var status = F.MirrorCover_status();
WScript.Echo("The mirror cover is: " + status);
F = null;
```