PAC Control™

Commands Quick Reference, Legacy Edition

Key

[1] Available only in PAC Control™ Professional
[2] Ethernet I/O™, Ultimate I/O™, and Simple I/O™ units
[3] Original High-Density Digital (HDD) modules
[4] *mistic*™ I/O units

The Type column shows whether the OptoScript™ command is a function command (f) or a procedure command (p). Function commands return a value from their action; procedure commands do not.

To enable commands for Ethernet I/O, Ultimate I/O, Simple I/O, High-Density Digital modules, and *mistic* I/O units:
1. On PAC Control's menu bar, click File > Strategy Options.
2. On the Legacy tab, click the option you want to enable, and then click Yes.
3. To enable more than one option, repeat step 2.
4. When finished, click OK.

Analog Point

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Calculate & Set Analog Gain	`CalcSetAnalogGain(On Point)`	f
Calculate & Set Analog Offset	`CalcSetAnalogOffset(On Point)`	f
Get & Clear Analog Filtered Value[1, 4]	`GetClearAnalogFilteredValue(From)`	f
Get & Clear Analog Maximum Value	`GetClearAnalogMaxValue(From)`	f
Get & Clear Analog Minimum Value	`GetClearAnalogMinValue(From)`	f
Get & Clear Analog Totalizer Value	`GetClearAnalogTotalizerValue(From)`	f
Get Analog Filtered Value[1, 4]	`GetAnalogFilteredValue(From)`	f
Get Analog Maximum Value	`GetAnalogMaxValue(From)`	f
Get Analog Minimum Value	`GetAnalogMinValue(From)`	f
Get Analog Square Root Filtered Value[1, 4]	`GetAnalogSquareRootFilteredValue(From)`	f
Get Analog Square Root Value[1, 4]	`GetAnalogSquareRootValue(From)`	f
Get Analog Totalizer Value	`GetAnalogTotalizerValue(From)`	f
Get HART Unique Address	`GetHARTUniqueAddress (Point, Polling Address, Unique Address, Timeout)`	f
Ramp Analog Output	`RampAnalogOutput(Ramp Endpoint, Units/Sec, Point to Ramp)`	p
Receive HART Response	`ReceiveHARTResponse(Point, Unique Address, Response, Timeout)`	f
Receive HART Burst Response	`ReceiveHARTBurstResponse(Point, Unique Address, Response, Timeout)`	f
Send/Receive HART Command	`SendReceiveHARTCommand (Point, Unique Address, Command, Parameters, Response, Timeout)`	f
Set Analog Filter Weight	`SetAnalogFilterWeight(To, On Point)`	p
Set Analog Gain	`SetAnalogGain(To, On Point)`	p
Set Analog Load Cell Fast Settle Level	`SetAnalogLoadCellFastSettleLevel(To, On Point)`	p
Set Analog Load Cell Filter Weight	`SetAnalogLoadCellFilterWeight(To, On Point)`	p
Set Analog Offset	`SetAnalogOffset(To, On Point)`	p
Set Analog Totalizer Rate	`SetAnalogTotalizerRate(To Seconds, On Point)`	p
Set Analog TPO Period	`SetAnalogTpoPeriod(To, On Point)`	p

Chart

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Call Chart	`CallChart(Chart)`	f
Calling Chart Running?	`IsCallingChartRunning()`	f
Calling Chart Stopped?	`IsCallingChartStopped()`	f
Calling Chart Suspended?	`IsCallingChartSuspended()`	f
Chart Running?	`IsChartRunning(Chart)`	f
Chart Stopped?	`IsChartStopped(Chart)`	f
Chart Suspended?	`IsChartSuspended(Chart)`	f
Continue Calling Chart	`ContinueCallingChart()`	f
Continue Chart	`ContinueChart(Chart)`	f

Chart (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Get Chart Status	`GetChartStatus(Chart)`	f
Start Chart	`StartChart(Chart)`	f
Stop Chart	`StopChart(Chart)`	f
Suspend Chart	`SuspendChart(Chart)`	f

Communication

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Accept Incoming Communication	`AcceptIncomingCommunication(Communication Handle)`	f
Clear Communication Receive Buffer	`ClearCommunicationReceiveBuffer(Communication Handle)`	p
Close Communication	`CloseCommunication(Communication Handle)`	f
Communication Open?	`IsCommunicationOpen(Communication Handle)`	f
Get Active Interrupt Mask[1, 4]	`GetActiveInterruptMask()`	f
Get Communication Handle Value	`GetCommunicationHandleValue(From, To)`	f
Get End-Of-Message Terminator	`GetEndOfMessageTerminator (Communication Handle)`	f
Get Number of Characters Waiting	`GetNumCharsWaiting(On Communication Handle)`	f
HTTP Get	`HttpGet(Response Content, Response Header, Get Header, Security Mode, URL Path, Put HTTP Status In, Port, Hostname)`	f
HTTP Post from String Table	`HttpPostFromStringTable(Response Content, Response Header, Post Content, Post Header, Security Mode, URL Path, Put HTTP Status In, Port, Hostname)`	f
HTTP Post Calculate Content Length	`HttpPostCalcContentLength(Post Content, Post Header, Length Index)`	f
Listen for Incoming Communication	`ListenForIncomingCommunication(Communication Handle)`	f
Open Outgoing Communication	`OpenOutgoingCommunication(Communication Handle)`	f
Receive Character	`ReceiveChar(Communication Handle)`	f
Receive N Characters	`ReceiveNChars(Put In, Number of Characters, Communication Handle)`	f
Receive Numeric Table	`ReceiveNumTable (Length, Start at Index, Of Table, Communication Handle)`	f
Receive Numeric Table Ex	`ReceiveNumTableEx (Length, Start at Index, Endian Mode, Bytes per Value, Of Table, Communication Handle)`	f
Receive Numeric Variable	`ReceiveNumVariable (Endian mode, Number of Bytes, Put in, Communication Handle)`	f
Receive Pointer Table	`ReceivePtrTable (Length, Start at Index, Of Table, Communication Handle)`	f
Receive String	`ReceiveString(Put In, Communication Handle)`	f
Receive String Table	`ReceiveStrTable (Length, Start at Index, Of Table, Communication Handle)`	f
Send Communication Handle Command	`SendCommunicationHandleCommand(Communication Handle, Command)`	f
Send Email	`SendEmail(Server Information, Recipients, Message Body)`	f
Send Email with Attachments	`SendEmailWithAttachments (Server Information, Recipients, Message Body, Attachment File Names)`	f
Set Communication Handle Value	`SetCommunicationHandleValue(Value, Communication Handle)`	p
Set End-Of-Message Terminator	`SetEndOfMessageTerminator(Communication Handle, To Character)`	p
Transfer N Characters	`TransferNChars(Destination Handle, Source Handle, Num Chars)`	f
Transmit Character	`TransmitChar(Character, Communication Handle)`	f
Transmit NewLine	`TransmitNewLine(Communication Handle)`	f
Transmit Numeric Table	`TransmitNumTable (Length, Start at Index, Of Table, Communication Handle)`	f
Transmit Pointer Table	`TransmitPtrTable (Length, Start at Index, Of Table, Communication Handle)`	f
Transmit/Receive Mistic I/O Hex String[1, 4]	`TransReceMisticIoHexStringWithCrc (Hex String, On Port, Put Result in)`	f
Transmit/Receive String	`TransmitReceiveString(String, Communication Handle, Put Result in)`	f
Transmit String	`TransmitString(String, Communication Handle)`	f
Transmit String Table	`TransmitStrTable (Length, Start at Index, Of Table, Communication Handle)`	f

Control Engine

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Calculate Strategy CRC	`CalcStrategyCrc()`	f
Erase Files in Permanent Storage	`EraseFilesInPermanentStorage()`	f
Get Available File Space	`GetAvailableFileSpace(File System Type)`	f
Get Control Engine Address	`GetControlEngineAddress()`	f
Get Control Engine Type	`GetEngineType()`	f
Get Firmware Version	`GetFirmwareVersion(Put in)`	p
Load Files From Permanent Storage	`LoadFilesFromPermanentStorage()`	f
Retrieve Strategy CRC	`RetrieveStrategyCrc()`	f
Save Files To Permanent Storage	`SaveFilesToPermanentStorage()`	f
Start Alternate Host Task	`StartAlternateHostTask()`	f

Digital Point

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Clear All Latches	`ClearAllLatches(On I/O Unit)`	p
Clear Counter	`ClearCounter(On Point)`	p
Clear Off-Latch	`ClearOffLatch(On Point)`	p
Clear On-Latch	`ClearOnLatch(On Point)`	p
Generate N Pulses	`GenerateNPulses (On Time (Seconds), Off Time (Seconds), Number of Pulses, On Point)`	p
Get & Clear Counter	`GetClearCounter(From Point)`	f
Get & Clear Off-Latch	`GetClearOffLatch(From Point)`	f
Get & Clear On-Latch	`GetClearOnLatch(From Point)`	f
Get & Restart Off-Pulse Measurement	`GetRestartOffPulseMeasurement(From Point)`	f
Get & Restart Off-Time Totalizer	`GetRestartOffTimeTotalizer(From Point)`	f
Get & Restart On-Pulse Measurement	`GetRestartOnPulseMeasurement(From Point)`	f
Get & Restart On-Time Totalizer	`GetRestartOnTimeTotalizer(From Point)`	f
Get & Restart Period	`GetRestartPeriod(From Point)`	f
Get Counter	`GetCounter(From Point)`	f
Get Frequency	`GetFrequency(From Point)`	f
Get Off-Latch	`GetOffLatch(From Point)`	f
Get Off-Pulse Measurement	`GetOffPulseMeasurement(From Point)`	f
Get Off-Pulse Measurement Complete Status	`GetOffPulseMeasurementCompleteStatus(From Point)`	f
Get Off-Time Totalizer	`GetOffTimeTotalizer(From Point)`	f
Get On-Latch	`GetOnLatch(From Point)`	f
Get On-Pulse Measurement	`GetOnPulseMeasurement(From Point)`	f
Get On-Pulse Measurement Complete Status	`GetOnPulseMeasurementCompleteStatus(From Point)`	f
Get On-Time Totalizer	`GetOnTimeTotalizer(From Point)`	f
Get Period	`GetPeriod(From Point)`	f
Get Period Measurement Complete Status	`GetPeriodMeasurementCompleteStatus(From Point)`	f
Off?	`IsOff(Point)`	f
Off-Latch Set?	`IsOffLatchSet(On Point)`	f
On?	`IsOn(Point)`	f
On-Latch Set?	`IsOnLatchSet(On Point)`	f
Set TPO Percent	`SetTpoPercent(To Percent, On Point)`	p
Set TPO Period	`SetTpoPeriod(To Seconds, On Point)`	p
Start Continuous Square Wave	`StartContinuousSquareWave (On Time (Seconds), Off Time (Seconds), On Point)`	p
Start Counter	`StartCounter(On Point)`	p
Start Off-Pulse	`StartOffPulse(Off Time (Seconds), On Point)`	p
Start On-Pulse	`StartOnPulse(On Time (Seconds), On Point)`	p
Stop Counter	`StopCounter(On Point)`	p
Turn Off	`TurnOff(Output)`	p
Turn On	`TurnOn(Output)`	p

Error Handling

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Add Message to Queue	`AddMessageToQueue(Severity, Message)`	p
Add User Error to Queue	`AddUserErrorToQueue(Error Number)`	p
Add User I/O Unit Error to Queue	`AddUserIoUnitErrorToQueue(Error Number, I/O Unit)`	p
Caused a Chart Error?	`HasChartCausedError(Chart)`	f
Caused an I/O Unit Error?	`HasIoUnitCausedError(I/O Unit)`	f
Clear All Errors	`ClearAllErrors()`	p
Copy Current Error to String	`CurrentErrorToString(Delimiter, String)`	p
Disable I/O Unit Causing Current Error	`DisableIoUnitCausingCurrentError()`	p
Enable I/O Unit Causing Current Error	`EnableIoUnitCausingCurrentError()`	p
Error?	`IsErrorPresent()`	f
Error on I/O Unit?	`IsErrorOnIoUnit()`	f
Get Error Code of Current Error	`GetErrorCodeOfCurrentError()`	f
Get Error Count	`GetErrorCount()`	f
Get ID of Block Causing Current Error	`GetIdOfBlockCausingCurrentError()`	f
Get Line Causing Current Error	`GetLineCausingCurrentError()`	f
Get Name of Chart Causing Current Error	`GetNameOfChartCausingCurrentError(Put in)`	p
Get Name of I/O Unit Causing Current Error	`GetNameOfIoUnitCausingCurrentError(Put in)`	p
Get Severity of Current Error	`GetSeverityOfCurrentError()`	f
Remove Current Error and Point to Next Error	`RemoveCurrentError()`	p
Stop Chart on Error	`StopChartOnError()`	p
Suspend Chart on Error	`SuspendChartOnError()`	f

Event/Reaction

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Clear All Event Latches[1,4]	`ClearAllEventLatches(On I/O Unit)`	p
Clear Event Latch[1,4]	`ClearEventLatch(On Event/Reaction)`	p
Clear I/O Unit Interrupt[1,4]	`ClearIoUnitInterrupt(On I/O Unit)`	p
Disable Interrupt on Event[1,4]	`DisableInterruptOnEvent(Event/Reaction)`	p
Disable Scanning for All Events[1,4]	`DisableScanningForAllEvents(On I/O Unit)`	p
Disable Scanning for Event[1,4]	`DisableScanningForEvent(Event/Reaction)`	p
Disable Scanning of Event/Reaction Group[1,4]	`DisableScanningOfEventReactionGroup(E/R Group)`	p
Enable Interrupt on Event[1,4]	`EnableInterruptOnEvent(Event/Reaction)`	p
Enable Scanning for All Events[1,4]	`EnableScanningForAllEvents(On I/O Unit)`	p
Enable Scanning for Event[1,4]	`EnableScanningForEvent(Event/Reaction)`	p
Enable Scanning of Event/Reaction Group[1,4]	`EnableScanningOfEventReactionGroup()`	p
Event Occurred?[1,4]	`HasEventOccurred(Event/Reaction)`	f
Event Occurring?[1,4]	`IsEventOccurring(Event/Reaction)`	f
Event Scanning Disabled?[1,4]	`IsEventScanningDisabled(Event/Reaction)`	f
Event Scanning Enabled?[1,4]	`IsEventScanningEnabled(Event/Reaction)`	f
Get & Clear Event Latches[1,4]	`GetClearEventLatches(E/R Group)`	f
Get Event Latches[1,4]	`GetEventLatches(E/R Group)`	f
Generating Interrupt?[1,4]	`IsGeneratingInterrupt(I/O Unit)`	f
Interrupt Disabled for Event?[1,4]	`IsInterruptDisabledForEvent(Event/Reaction)`	f
Interrupt Enabled for Event?[1,4]	`IsInterruptEnabledForEvent(Event/Reaction)`	f
Read Event/Reaction Hold Buffer[1,4]	`ReadEventReactionHoldBuffer(Event/Reaction)`	f

High Density Digital Module

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Clear HDD Module Off-Latches[3]	`ClearHddModuleOffLatches(I/O Unit, Module Number, Clear Mask)`	f
Clear HDD Module On-Latches[3]	`ClearHddModuleOnLatches(I/O Unit, Module Number, Clear Mask)`	f

High Density Digital Module (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Get & Clear All HDD Module Off-Latches[3]	GetClearAllHddModuleOffLatches (*I/O Unit, Start Index, Put Result In*)	f
Get & Clear All HDD Module On-Latches[3]	GetClearAllHddModuleOnLatches(*I/O Unit, Start Index, Put Result In*)	f
Get & Clear HDD Module Counter[3]	GetClearHddModuleCounter (*I/O Unit, Module Number, Point Number, Put Result In*)	f
Get & Clear HDD Module Counters[3]	GetClearHddModuleCounters (*I/O Unit, Module Number, Start Table Index, Put Result In*)	f
Get & Clear HDD Module Off-Latches[3]	GetClearHddModuleOffLatches(*I/O Unit, Module Number, Put Result In*)	f
Get & Clear HDD Module On-Latches[3]	GetClearHddModuleOnLatches(*I/O Unit, Module Number, Put Result In*)	f
Get All HDD Module Off-Latches[3]	GetAllHddModuleOffLatches(*I/O Unit, Start Index, Put Result In*)	f
Get All HDD Module On-Latches[3]	GetAllHddModuleOnLatches(*I/O Unit, Start Index, Put Result In*)	f
Get All HDD Module States[3]	GetAllHddModuleStates(*I/O Unit, Start Index, Put Result In*)	f
Get HDD Module Counters[3]	GetHddModuleCounters (*I/O Unit, Module Number, Start Table Index, Put Result In*)	f
Get HDD Module Off-Latches[3]	GetHddModuleOffLatches(*I/O Unit, Module Number, Put Result In*)	f
Get HDD Module On-Latches[3]	GetHddModuleOnLatches(*I/O Unit, Module Number, Put Result In*)	f
Get HDD Module States[3]	GetHddModuleStates(*I/O Unit, Module Number, Put Result In*)	f
Set HDD Module from MOMO Masks[3]	SetHddModulefromMOMOMasks (*I/O Unit, Module Number, Must-On Mask, Must-Off Mask*)	f
Turn Off HDD Module Point[3]	TurnOffHDDModulePoint(*I/O Unit, Module Number, Point Number*)	f
Turn On HDD Module Point[3]	TurnOnHddModulePoint(*I/O Unit, Module Number, Point Number*)	f

I/O Unit

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Clear I/O Unit Configured Flag	ClearIoUnitConfiguredFlag(*I/O Unit*)	p
Get I/O Unit as Binary Value	GetIoUnitAsBinaryValue(*I/O Unit*)	f
Get I/O Unit as Binary Value 64	GetIoUnitAsBinaryValue64(*I/O Unit*)	f
Get Target Address State[1]	GetTargetAddressState(*Enable Mask, Active Mask, I/O Unit*)	p
I/O Unit Ready?	IsIoUnitReady(*I/O Unit*)	f
IVAL Move Numeric Table to I/O Unit	IvalMoveNumTableToIoUnit(*Start at Index, Of Table, Move to*)	p
IVAL Move Numeric Table to I/O Unit Ex	IvalMoveNumTableToIoUnitEx (*From Table, With Starting Index, To I/O Unit*)	p
Move I/O Unit to Numeric Table	MoveIoUnitToNumTable(*I/O Unit, Starting Index, Of Table*)	p
Move I/O Unit to Numeric Table Ex	MoveIoUnitToNumTableEx(*From I/O Unit, To Table, With Starting Index*)	p
Move Numeric Table to I/O Unit	MoveNumTableToIoUnit(*Start at Index, Of Table, Move to*)	p
Move Numeric Table to I/O Unit Ex	MoveNumTableToIoUnitEx(*From Table, With Starting Index, To I/O Unit*)	p
Set All Target Address States[1]	SetAllTargetAddressStates(*Must-On Mask, Must-Off Mask, Active Mask*)	p
Set I/O Unit Configured Flag	SetIoUnitConfiguredFlag(*For I/O Unit*)	p
Set I/O Unit from MOMO Masks	SetIoUnitFromMomo(*Must-On Mask, Must-Off Mask, Digital I/O Unit*)	p
Set Target Address State[1]	SetTargetAddressState (*Must-On Mask, Must-Off Mask, Active Mask, I/O Unit*)	p
Write I/O Unit Configuration to EEPROM	WriteIoUnitConfigToEeprom(*On I/O Unit*)	p

I/O Unit—Event Message

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Get I/O Unit Event Message State	GetIoUnitEventMsgState(*I/O Unit, Event Message #, Put Result in*)	f
Get I/O Unit Event Message Text	GetIoUnitEventMsgText(*I/O Unit, Event Message #, Put Result in*)	f
Set I/O Unit Event Message State	SetIoUnitEventMsgState(*I/O Unit, Event Message #, State*)	f
Set I/O Unit Event Message Text	SetIoUnitEventMsgText(*I/O Unit, Event Message #, Message Text*)	f

I/O Unit—Memory Map

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Read Number from I/O Unit Memory Map	`ReadNumFromIoUnitMemMap(I/O Unit, Mem address, To)`	f
Read Numeric Table from I/O Unit Memory Map	`ReadNumTableFromIoUnitMemMap (Length, Start Index, I/O Unit, Mem address, To)`	f
Read String from I/O Unit Memory Map	`ReadStrFromIoUnitMemMap(Length, I/O Unit, Mem address, To)`	f
Read String Table from I/O Unit Memory Map	`ReadStrTableFromIoUnitMemMap (Length, Start Index, I/O Unit, Mem address, To)`	f
Write Number to I/O Unit Memory Map	`WriteNumToIoUnitMemMap(I/O Unit, Mem address, Variable)`	f
Write Numeric Table to I/O Unit Memory Map	`WriteNumTableToIoUnitMemMap (Length, Start Index, I/O Unit, Mem address, Table)`	f
Write String Table to I/O Unit Memory Map	`WriteStrTableToIoUnitMemMap (Length, Start Index, I/O Unit, Mem address, Table)`	f
Write String to I/O Unit Memory Map	`WriteStrToIoUnitMemMap(I/O Unit, Mem address, Variable)`	f

I/O Unit—Scratch Pad

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Get I/O Unit Scratch Pad Bits	`GetIoUnitScratchPadBits(I/O Unit, Put Result in)`	f
Get I/O Unit Scratch Pad Float Element	`GetIoUnitScratchPadFloatElement(I/O Unit, Index, Put Result in)`	f
Get I/O Unit Scratch Pad Float Table	`GetIoUnitScratchPadFloatTable (I/O Unit, Length, From Index, To Index, To Table)`	f
Get I/O Unit Scratch Pad Integer 32 Element	`GetIoUnitScratchPadInt32Element(I/O Unit, Index, Put Result in)`	f
Get I/O Unit Scratch Pad Integer 32 Table	`GetIoUnitScratchPadInt32Table (I/O Unit, Length, From Index, To Index, To Table)`	f
Get I/O Unit Scratch Pad String Element	`GetIoUnitScratchPadStringElement (I/O Unit, Index, Put Result in)`	f
Get I/O Unit Scratch Pad String Table	`GetIoUnitScratchPadStringTable (I/O Unit, Length, From Index, To Index, To Table)`	f
Set I/O Unit Scratch Pad Bits from MOMO Mask	`SetIoUnitScratchPadBitsFromMomo (I/O Unit, Must-On Mask, Must-Off Mask)`	f
Set I/O Unit Scratch Pad Float Element	`SetIoUnitScratchPadFloatElement(I/O Unit, Index, From)`	f
Set I/O Unit Scratch Pad Float Table	`SetIoUnitScratchPadFloatTable (I/O Unit, Length, To Index, From Index, From Table)`	f
Set I/O Unit Scratch Pad Integer 32 Element	`SetIoUnitScratchPadInt32Element(I/O Unit, Index, From)`	f
Set I/O Unit Scratch Pad Integer 32 Table	`SetIoUnitScratchPadInt32Table (I/O Unit, Length, To Index, From Index, From Table)`	f
Set I/O Unit Scratch Pad String Element	`SetIoUnitScratchPadStringElement(I/O Unit, Index, From)`	f
Set I/O Unit Scratch Pad String Table	`SetIoUnitScratchPadStringTable (I/O Unit, Length, To Index, From Index, From Table)`	f

Logical

PAC Control Command	OptoScript Equivalent (Arguments)	Type
AND	`x and y`	f
AND?	`See AND`	f
Bit AND	`x bitand y`	f
Bit AND?	`See Bit AND`	f
Bit Change	`BitChange(Set flag, Bit to Change, Output)`	p
Bit Clear	`BitClear(Item, Bit to Clear)`	f
Bit Copy	`BitCopy(Server Bit to Set, Destination, Destination Index, Bit to Read, Source, Source Index)`	f
Bit NOT	`bitnot x`	f
Bit NOT?	`See Bit NOT`	f
Bit Off in Numeric Table Element?	`IsBitOffInNumTableElement(At Index, Of Table, Bit)`	f
Bit Off?	`IsBitOff(In, Bit)`	f
Bit On in Numeric Table Element?	`IsBitOnInNumTableElement(At Index, Of Table, Bit)`	f
Bit On?	`IsBitOn(In, Bit)`	f
Bit OR	`x bitor y`	f
Bit OR?	`See Bit OR`	f
Bit Rotate	`BitRotate(Item, Count)`	f

Logical (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Bit Set	`BitSet(Item, Bit to Set)`	f
Bit Shift	`x << nBitsToShift`	f
Bit Test	`BitTest(Item, Bit to Test)`	f
Bit XOR	`x bitxor y`	f
Bit XOR?	`See Bit XOR`	f
Equal to Numeric Table Element?	`n == nt[0]`	f
Equal?	`x == y`	f
Flip Flop JK	`FlipFlopJK(Set [J], Reset [K], Output [Q])`	p
Float to Int32 Bits	`FloatToInt32Bits(Server URL)`	f
Get High Bits of Integer 64	`GetHighBitsOfInt64(High Bits From)`	f
Get Low Bits of Integer 64	`GetLowBitsOfInt64(Integer 64)`	f
Greater Than Numeric Table Element?	`x > nt[0]`	f
Greater Than or Equal to Numeric Table Element?	`x >= t[0]`	f
Greater Than or Equal?	`x >= y`	f
Greater?	`x > y`	f
Int32 to Float Bits	`Int32ToFloatBits(nInt32)`	f
Less Than Numeric Table Element?	`x < nt[0]`	f
Less Than or Equal to Numeric Table Element?	`x <= nt[0]`	f
Less Than or Equal?	`x <= y`	f
Less?	`x < y`	f
Make Integer 64	`MakeInt64(High Integer, Low Integer)`	f
Move 32 Bits	`Move32Bits(From, To)`	p
NOT	`not x`	f
Not Equal to Numeric Table Element?	`n <> nt[0]`	f
Not Equal?	`x <> y`	f
NOT?	`not x`	f
Numeric Table Element Bit Clear	`NumTableElementBitClear (Element Index, Of Integer Table, Bit to Clear)`	p
Numeric Table Element Bit Set	`NumTableElementBitSet (Element Index, Of Integer Table, Bit to Set)`	p
Numeric Table Element Bit Test	`NumTableElementBitTest (Element Index, Of Integer Table, Bit to Test)`	f
OR	`x or y`	f
OR?	`See OR`	f
Set Variable False	`SetVariableFalse(Variable)`	p
Set Variable True	`SetVariableTrue(Variable)`	p
Test Equal	`See Equal?`	f
Test Greater	`See Greater?`	f
Test Greater or Equal	`See Greater Than or Equal?`	f
Test Less	`See Less?`	f
Test Less or Equal	`See Less Than or Equal?`	f
Test Not Equal	`See Not Equal?`	f
Test Within Limits	`See Within Limits?`	f
Variable False?	`IsVariableFalse(Variable)`	f
Variable True?	`IsVariableTrue(Variable)`	f
Within Limits?	`IsWithinLimits(Value, Low Limit, High Limit)`	f
XOR	`x xor y`	f
XOR?	`See XOR`	f

Mathematical

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Absolute Value	`AbsoluteValue(Of)`	f
Add	`x + y`	f
Arccosine	`Arccosine(Of)`	f
Arcsine	`Arcsine(Of)`	f
Arctangent	`Arctangent(Of)`	f
Clamp Float Table Element	`ClampFloatTableElement (High Limit, Low Limit, Element Index, Of Float Table)`	p
Clamp Float Variable	`ClampFloatVariable(High Limit, Low Limit, Float Variable)`	p

Mathematical (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Clamp Integer 32 Table Element	`ClampInt32TableElement (High Limit, Low Limit, Element Index, Of Integer 32 Table)`	p
Clamp Integer 32 Variable	`ClampInt32Variable(High Limit, Low Limit, Integer 32 Variable)`	p
Complement	`-x`	p
Cosine	`Cosine(Of)`	f
Decrement Variable	`DecrementVariable(Variable)`	p
Divide	`x / y`	f
Generate Random Number	`GenerateRandomNumber()`	f
Hyperbolic Cosine	`HyperbolicCosine(Of)`	f
Hyperbolic Sine	`HyperbolicSine(Of)`	f
Hyperbolic Tangent	`HyperbolicTangent(Of)`	f
Increment Variable	`IncrementVariable(Variable)`	p
Maximum	`Max(Compare, With)`	f
Minimum	`Min(Compare, With)`	f
Modulo	`x % y`	f
Multiply	`x * y`	f
Natural Log	`NaturalLog(Of)`	f
Raise e to Power	`RaiseEToPower(Exponent)`	f
Raise to Power	`Power(Raise, To the)`	f
Round	`Round(Value)`	f
Seed Random Number	`SeedRandomNumber()`	p
Sine	`Sine(Of)`	f
Square Root	`SquareRoot(Of)`	f
Subtract	`x - y`	f
Tangent	`Tangent(Of)`	f
Truncate	`Truncate(Value)`	f

Miscellaneous

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Comment (Block)	`/* block comment */`	p
Comment (Single Line)	`// single line comment`	f
Flag Lock	`FlagLock(Flag, Timeout)`	f
Flag Unlock	`FlagUnlock(Flag, Force unlock)`	f
Float Valid?	`IsFloatValid(Float)`	f
Generate Reverse CRC-16 on Table (32 bit)	`GenerateReverseCrc16OnTable32 (Start Value, Table, Starting Element, Number of Elements)`	f
Get Length of Table	`GetLengthOfTable(Table)`	f
Get Type From Name	`GetTypeFromName(Name)`	f
Get Value From Name	`GetValueFromName(Name, Put Result In)`	f
Move	`x = y;`	p
Move from Numeric Table Element	`x = nt[0];`	f
Move Numeric Table Element to Numeric Table	`nt1[0] = nt2[5];`	p
Move Numeric Table to Numeric Table	`MoveNumTableToNumTable (From Table, From Index, To Table, To Index, Length)`	p
Move to Numeric Table Element	`nt[0] = x;`	p
Move to Numeric Table Elements	`MoveToNumTableElements(From, Start Index, End Index, Of Table)`	p
Shift Numeric Table Elements	`ShiftNumTableElements(Shift Count, Table)`	p

PID—Ethernet

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Get PID Configuration Flags	`GetPidConfigFlags(PID Loop)`	f
Get PID Current Input	`GetPidCurrentInput(PID Loop)`	f
Get PID Current Setpoint	`GetPidCurrentSetpoint(PID Loop)`	f
Get PID Feed Forward	`GetPidFeedForward(PID Loop)`	f
Get PID Feed Forward Gain	`GetPidFeedForwardGain(PID Loop)`	f
Get PID Forced Output When Input Over Range	`GetPidForcedOutputWhenInputOverRange(PID Loop)`	f

PID—Ethernet (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Get PID Forced Output When Input Under Range	GetPidForcedOutputWhenInputUnderRange(*PID Loop*)	f
Get PID Gain	GetPidGain(*PID Loop*)	f
Get PID Input	GetPidInput(*PID Loop*)	f
Get PID Input High Range	GetPidInputHighRange(*PID Loop*)	f
Get PID Input Low Range	GetPidInputLowRange(*PID Loop*)	f
Get PID Max Output Change	GetPidMaxOutputChange(*PID Loop*)	f
Get PID Min Output Change	GetPidMinOutputChange(*PID Loop*)	f
Get PID Mode	GetPidMode(*PID Loop*)	f
Get PID Output	GetPidOutput(*PID Loop*)	f
Get PID Output High Clamp	GetPidOutputHighClamp(*PID Loop*)	f
Get PID Output Low Clamp	GetPidOutputLowClamp(*PID Loop*)	f
Get PID Scan Time	GetPidScanTime(*PID Loop*)	f
Get PID Setpoint	GetPidSetpoint(*PID Loop*)	f
Get PID Status Flags	GetPidStatusFlags(*PID Loop*)	f
Get PID Tune Derivative	GetPidTuneDerivative(*PID Loop*)	f
Get PID Tune Integral	GetPidTuneIntegral(*PID Loop*)	f
Set PID Configuration Flags	SetPidConfigFlags(*PID Loop, Configuration Flags*)	p
Set PID Feed Forward	SetPidFeedForward(*PID Loop, Feed Forward*)	p
Set PID Feed Forward Gain	SetPidFeedForwardGain(*PID Loop, Feed Fwd Gain*)	p
Set PID Forced Output When Input Over Range	SetPidForcedOutputWhenInputOverRange(*PID Loop, Forced Output*)	p
Set PID Forced Output When Input Under Range	SetPidForcedOutputWhenInputUnderRange(*PID Loop, Forced Output*)	p
Set PID Gain	SetPidGain(*PID Loop, Gain*)	p
Set PID Input	SetPidInput(*PID Loop, Input*)	p
Set PID Input High Range	SetPidInputHighRange(*PID Loop, High Range*)	p
Set PID Input Low Range	SetPidInputLowRange(*PID Loop, Low Range*)	p
Set PID Max Output Change	SetPidMaxOutputChange(*PID Loop, Max Change*)	p
Set PID Min Output Change	SetPidMinOutputChange(*PID Loop, Min Change*)	p
Set PID Mode	SetPidMode(*PID Loop, Mode*)	p
Set PID Output	SetPidOutput(*PID Loop, Output*)	p
Set PID Output High Clamp	SetPidOutputHighClamp(*PID Loop, High Clamp*)	p
Set PID Output Low Clamp	SetPidOutputLowClamp(*PID Loop, Low Clamp*)	p
Set PID Scan Time	SetPidScanTime(*PID Loop, Scan Time*)	p
Set PID Setpoint	SetPidSetpoint(*PID Loop, Setpoint*)	p
Set PID Tune Derivative	SetPidTuneDerivative(*PID Loop, Derivative*)	p
Set PID Tune Integral	SetPidTuneIntegral(*PID Loop, Integral*)	p

PID—Mistic

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Clamp Mistic PID Output[1, 4]	ClampMisticPidOutput(*High Clamp, Low Clamp, On PID Loop*)	p
Clamp Mistic PID Setpoint[1, 4]	ClampMisticPidSetpoint(*High Clamp, Low Clamp, On PID Loop*)	p
Disable Mistic PID Output[1, 4]	DisableMisticPidOutput(*Of PID Loop*)	p
Disable Mistic PID Output Tracking in Manual Mode[1, 4]	DisableMisticPidOutputTrackingInManualMode(*On PID Loop*)	p
Disable Mistic PID Setpoint Tracking in Manual Mode[1, 4]	DisableMisticPidSetpointTrackingInManualMode(*On PID Loop*)	p
Enable Mistic PID Output[1, 4]	EnableMisticPidOutput(*On PID Loop*)	p
Enable Mistic PID Output Tracking in Manual Mode[1, 4]	EnableMisticPidOutputTrackingInManualMode(*On PID Loop*)	p
Enable Mistic PID Setpoint Tracking in Manual Mode[1, 4]	EnableMisticPidSetpointTrackingInManualMode(*On PID Loop*)	p
Get Mistic PID Control Word[1, 4]	GetMisticPidControlWord(*From PID Loop*)	f
Get Mistic PID D Term[1, 4]	GetMisticPidDTerm(*From PID Loop*)	f
Get Mistic PID I Term[1, 4]	GetMisticPidITerm(*From PID Loop*)	f
Get Mistic PID Input[1, 4]	GetMisticPidInput(*PID Loop*)	f
Get Mistic PID Mode[1, 4]	GetMisticPidMode(*PID Loop*)	f
Get Mistic PID Output[1, 4]	GetMisticPidOutput(*PID Loop*)	f
Get Mistic PID Output Rate of Change[1, 4]	GetMisticPidOutputRateOfChange(*From PID Loop*)	f
Get Mistic PID P Term[1, 4]	GetMisticPidPTerm(*From PID Loop*)	f
Get Mistic PID Scan Rate[1, 4]	GetMisticPidScanRate(*From PID Loop*)	f

PID—Mistic (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Get Mistic PID Setpoint[1, 4]	`GetMisticPidSetpoint(PID Loop)`	f
Set Mistic PID Control Word[1, 4]	`SetMisticPidControlWord(On-Mask, Off-Mask, For PID Loop)`	p
Set Mistic PID D Term[1, 4]	`SetMisticPidDTerm(To, On PID Loop)`	p
Set Mistic PID I Term[1, 4]	`SetMisticPidITerm(To, On PID Loop)`	p
Set Mistic PID Input[1, 4]	`SetMisticPidInput(Input, PID Loop)`	p
Set Mistic PID Mode to Auto[1, 4]	`SetMisticPidModeToAuto(On PID Loop)`	p
Set Mistic PID Mode to Manual[1, 4]	`SetMisticPidModeToManual(On PID Loop)`	p
Set Mistic PID Output Rate of Change[1, 4]	`SetMisticPidOutputRateOfChange(To, On PID Loop)`	p
Set Mistic PID P Term[1, 4]	`SetMisticPidPTerm(To, On PID Loop)`	p
Set Mistic PID Scan Rate[1, 4]	`SetMisticPidScanRate(To, On PID Loop)`	p
Set Mistic PID Setpoint[1, 4]	`SetMisticPidSetpoint(PID Loop, Setpoint)`	p

Pointers

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Clear Pointer	`pn1 = null;`	f
Clear Pointer Table Element	`pt[0] = null;`	p
Get Pointer From Name	`GetPointerFromName(Name, Pointer)`	p
Move from Pointer Table Element	`pn = pt[0];`	f
Move to Pointer	`pn = &n;`	f
Move to Pointer Table Element	`pt[0] = &n;`	f
Pointer Equal to Null?	`pn == null`	f
Pointer Table Element Equal to Null?	`pt[0] == null`	f

Simulation

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Communication to All I/O Points Enabled?	`IsCommToAllIoPointsEnabled()`	f
Communication To All I/O Units Enabled?	`IsCommToAllIoUnitsEnabled()`	f
Disable Communication to All I/O Points	`DisableCommuncationToAllIoPoints()`	p
Disable Communication to All I/O Units	`DisableCommunicationToAllIoUnits()`	p
Disable Communication to Event/Reaction[1, 4]	`DisableCommunicationToEventReaction (Event/Reaction)`	p
Disable Communication to I/O Unit	`DisableCommunicationToIoUnit(I/O Unit)`	p
Disable Communication to Mistic PID Loop[1, 4]	`DisableCommunicationtoMisticPidLoop(PID Loop)`	p
Disable Communication to PID Loop	`DisableCommunicationtoPidLoop(PID Loop)`	p
Disable Communication to Point	`DisableCommunicationToPoint(Point)`	p
Disable Event/Reaction Group[1]	`DisableEventReactionGroup(E/R Group)`	p
Enable Communication to All I/O Points	`EnableCommunicationToAllIoPoints()`	p
Enable Communication to All I/O Units	`EnableCommunicationToAllIoUnits()`	p
Enable Communication to Event/Reaction[1, 4]	`EnableCommunicationToEventReaction(Event/Reaction)`	p
Enable Communication to I/O Unit	`EnableCommunicationToIoUnit(I/O Unit)`	p
Enable Communication to Mistic PID Loop[1, 4]	`EnableCommunicationToMisticPidLoop(PID Loop)`	p
Enable Communication to PID Loop	`EnableCommunicationtoPidLoop(PID Loop)`	p
Enable Communication to Point	`EnableCommunicationToPoint(Point)`	p
Enable Event/Reaction Group[1, 4]	`EnableEventReactionGroup(E/R Group)`	p
Event/Reaction Communication Enabled?[1, 4]	`IsEventReactionCommEnabled (Event/Reaction)`	f
Event/Reaction Group Communication Enabled?[1, 4]	`IsEventReactionGroupEnabled(E/R Group)`	f
I/O Point Communication Enabled?	`IsIoPointCommEnabled(I/O Point)`	f
I/O Unit Communication Enabled?	`IsIoUnitCommEnabled(I/O Unit)`	f
IVAL Set Analog Filtered Value	`IvalSetAnalogFilteredValue(To, On Point)`	p
IVAL Set Analog Maximum Value	`IvalSetAnalogMaxValue(To, On Point)`	p
IVAL Set Analog Minimum Value	`IvalSetAnalogMinValue(To, On Point)`	p
IVAL Set Analog Point	`IvalSetAnalogPoint(To, On Point)`	p
IVAL Set Counter	`IvalSetCounter(To, On Point)`	p
IVAL Set Frequency	`IvalSetFrequency(To, On Point)`	p

Simulation (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
IVAL Set I/O Unit from MOMO Masks	`IvalSetiOUnitfromMOMO(On Mask, Off Mask, On I/O Unit)`	p
IVAL Set Mistic PID Control Word[1, 4]	`IvalSetPidControlWord(On Mask, Off Mask, For PID Loop)`	p
IVAL Set Mistic PID Process Term[1, 4]	`IvalSetMisticPidProcessTerm(To, On PID Loop)`	p
IVAL Set Off-Latch	`IvalSetOffLatch(To, On Point)`	p
IVAL Set Off-Pulse	`IvalSetOffPulse(To, On Point)`	p
IVAL Set Off-Totalizer	`IvalSetOffTotalizer(To, On Point)`	p
IVAL Set On-Latch	`IvalSetOnLatch(To, On Point)`	p
IVAL Set On-Pulse	`IvalSetOnPulse(To, On Point)`	p
IVAL Set On-Totalizer	`IvalSetOnTotalizer(To, On Point)`	p
IVAL Set Period	`IvalSetPeriod(To, On Point)`	p
IVAL Set TPO Percent	`IvalSetTpoPercent(To, On Point)`	p
IVAL Set TPO Period	`IvalSetTpoPeriod(Value, On Point)`	p
IVAL Turn Off	`IvalTurnOff(Point)`	p
IVAL Turn On	`IvalTurnOn(Point)`	p
Mistic PID Loop Communication Enabled?[1, 4]	`IsMisticPidLoopCommEnabled(PID Loop)`	p
PID Loop Communication Enabled?	`IsPidLoopCommEnabled(PID Loop)`	f

String

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Append Character to String	`s1 = s1 + "a";`	p
Append String to String	`s1 = s1 + s2;`	p
Compare Strings	`CompareStrings(String 1, String 2)`	f
Convert Float to String	`FloatToString(Convert, Length, Decimals, Put Result in)`	p
Convert Hex String to Number	`HexStringToNumber(Convert)`	f
Convert IEEE Hex String to Number	`IEEEHexStringToNumber(Convert)`	f
Convert Integer 32 to IP Address String	`Int32ToIpAddressString(Convert, Put Result In)`	f
Convert IP Address String to Integer 32	`IpAddressStringToInt32(Convert)`	f
Convert Mistic I/O Hex String to Float[1, 4]	`MisticIoHexToFloat(Convert)`	f
Convert Number to Formatted Hex String	`NumberToFormattedHexString(Convert, Length, Put Result in)`	p
Convert Number to Hex String	`NumberToHexString(Convert, Put Result in)`	p
Convert Number to Mistic I/O Hex String[1, 4]	`NumberToMisticIoHex(Convert, Put Result in)`	p
Convert Number to String	`NumberToString(Convert, Put Result in)`	p
Convert Number to String Field	`NumberToStringField(Convert, Length, Put Result in)`	p
Convert String to Float	`StringToFloat(Convert)`	f
Convert String to Integer 32	`StringToInt32(Convert)`	f
Convert String to Integer 64	`StringToInt64(Convert)`	f
Convert String to Lower Case	`StringToLowerCase(Convert)`	p
Convert String to Upper Case	`StringToUpperCase(Convert)`	p
Find Character in String	`FindCharacterInString(Find, Start at Index, Of String)`	f
Find Substring in String	`FindSubstringInString(Find, Start at Index, Of String)`	f
Generate Checksum on String	`GenerateChecksumOnString(Start Value, On String)`	f
Generate Forward CCITT on String	`GenerateForwardCcittOnString(Start Value, On String)`	f
Generate Forward CRC-16 on String	`GenerateForwardCrc16OnString(Start Value, On String)`	f
Generate Reverse CCITT on String	`GenerateReverseCcittOnString(Start Value, On String)`	f
Generate Reverse CRC-16 on String	`GenerateReverseCrc16OnString(Start Value, On String)`	f
Get Nth Character	`GetNthCharacter(From String, Index)`	f
Get String Length	`GetStringLength(Of String)`	f
Get Substring	`GetSubstring (From String, Start at Index, Num. Characters, Put Result in)`	p
Move from String Table Element	`s = st[0];`	p
Move String	`s1 = s2;`	p
Move to String Table Element	`st[0] = s;`	p
Move to String Table Elements	`MoveToStrTableElements(From, Start Index, End Index, Of Table)`	p
Pack Float into String	`PackFloatIntoString(From Value, To String, Start Index, Width, Endian Type, Put Result In)`	f
Pack Integer 32 into String	`PackInt32IntoString(From Value, To String, Start Index, Width, Endian Type, Put Result In)`	f

String (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Pack Integer 64 into String	`PackInt64IntoString(From Value, To String, Start Index, Width, Endian Type, Put Result In)`	f
Pack String into String	`PackStringIntoString(From Value, To String, Start Index, Width, Data Type, Put Result In)`	f
Set Nth Character	`SetNthCharacter(To, In String, At Index)`	f
String Equal?	`s1 == s2`	f
String Equal to String Table Element?	`s == st[0]`	f
Test Equal Strings	`See String Equal?`	f
Trim String	`TrimString(String, Option)`	f
Unpack String	`UnpackString(From String, Start Index, Width, To Value, Data Type, Endian Type, Put Result In)`	f
Verify Checksum on String	`VerifyChecksumOnString(Start Value, On String)`	f
Verify Forward CCITT on String	`VerifyForwardCcittOnString(Start Value, On String)`	f
Verify Forward CRC-16 on String	`VerifyForwardCrc16OnString(Start Value, On String)`	f
Verify Reverse CCITT on String	`VerifyReverseCcittOnString(Start Value, On String)`	f
Verify Reverse CRC-16 on String	`VerifyReverseCrc16OnString(Start Value, On String)`	f

Time/Date

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Copy Date to String (DD/MM/YYYY)	`DateToStringDDMMYYYY(String)`	p
Copy Date to String (MM/DD/YYYY)	`DateToStringMMDDYYYY(String)`	p
Copy Time to String	`TimeToString(String)`	p
Convert Date & Time to NTP Timestamp	`DateTimeToNtpTimestamp(Date&Time, NTP Timestamp, Put Result in)`	f
Convert NTP Timestamp to Date & Time	`NtpTimestampToDateTime(Date&Time, NTP Timestamp, Put Result in)`	f
Get Date & Time	`GetDateTime(Table)`	f
Get Day	`GetDay()`	f
Get Day of Week	`GetDayOfWeek()`	f
Get Hours	`GetHours()`	f
Get Julian Day	`GetJulianDay()`	f
Get Minutes	`GetMinutes()`	f
Get Month	`GetMonth()`	f
Get Seconds	`GetSeconds()`	f
Get Seconds Since Midnight	`GetSecondsSinceMidnight()`	f
Get System Time	`GetSystemTime()`	f
Get Time Zone Description	`GetTimeZoneDescription(Configuration, Description)`	f
Get Time Zone Offset	`GetTimeZoneOffset(Configuration)`	f
Get Year	`GetYear()`	f
Set Date	`SetDate(To)`	p
Set Day	`SetDay(To)`	p
Set Hours	`SetHours(To)`	p
Set Minutes	`SetMinutes(To)`	p
Set Month	`SetMonth(To)`	p
Set Seconds	`SetSeconds(To)`	p
Set Time	`SetTime(To)`	p
Set Time Zone Configuration	`SetTimeZoneConfiguration(Configuration)`	f
Set Year	`SetYear(To)`	p
Synchronize Clock SNTP	`SynchronizeClockSNTP(Timeout, Server URL, Put Result in)`	f

Timing

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Continue Timer	`ContinueTimer(Timer)`	p
Delay (mSec)	`DelayMsec(Milliseconds)`	p
Delay (Sec)	`DelaySec(Seconds)`	p
Down Timer Expired?	`HasDownTimerExpired(Down Timer)`	f
Get & Restart Timer	`GetRestartTimer(Timer)`	f
Pause Timer	`PauseTimer(Timer)`	p

Timing (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Set Down Timer Preset Value	`SetDownTimerPreset(`*`Target Value, Down Timer`*`)`	p
Set Up Timer Target Value	`SetUpTimerTarget(`*`Target Value, Up Timer`*`)`	p
Start Timer	`StartTimer(`*`Timer`*`)`	p
Stop Timer	`StopTimer(`*`Timer`*`)`	p
Timer Expired?	`HasTimerExpired(`*`Timer`*`)`	f
Up Timer Target Time Reached?	`HasUpTimerReachedTargetTime(`*`Up Timer`*`)`	f