



Commands Quick Reference

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

Chart

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

Chart (Continued)

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

Communication

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Accept Incoming Communication	AcceptIncomingCommunication(<i>Communication Handle</i>)	f
Clear Communication Receive Buffer	ClearCommunicationReceiveBuffer(<i>Communication Handle</i>)	p
Close Communication	CloseCommunication(<i>Communication Handle</i>)	f
Communication Open?	IsCommunicationOpen(<i>Communication Handle</i>)	f
Get Active Interrupt Mask ^{1, 4}	GetActiveInterruptMask()	f
Get Communication Handle Value	GetCommunicationHandleValue(<i>From, To</i>)	f
Get End-Of-Message Terminator	GetEndOfMessageTerminator (<i>Communication Handle</i>)	f
Get Number of Characters Waiting	GetNumCharsWaiting(<i>On Communication Handle</i>)	f
HTTP Get	HttpGet(<i>Response Content, Response Header, Get Header, Security Mode, URL Path, Put HTTP Status In, Port, Hostname</i>)	f
HTTP Post from String Table	HttpPostFromStringTable(<i>Response Content, Response Header, Post Content, Post Header, Security Mode, URL Path, Put HTTP Status In, Port, Hostname</i>)	f
HTTP Post Calculate Content Length	HttpPostCalcContentLength(<i>Post Content, Post Header, Length Index</i>)	f
Listen for Incoming Communication	ListenForIncomingCommunication(<i>Communication Handle</i>)	f
Open Outgoing Communication	OpenOutgoingCommunication(<i>Communication Handle</i>)	f
Receive Character	ReceiveChar(<i>Communication Handle</i>)	f
Receive N Characters	ReceiveNChars(<i>Put In, Number of Characters, Communication Handle</i>)	f
Receive Numeric Table	ReceiveNumTable (<i>Length, Start at Index, Of Table, Communication Handle</i>)	f
Receive Numeric Table Ex	ReceiveNumTableEx (<i>Length, Start at Index, Endian Mode, Bytes per Value, Of Table, Communication Handle</i>)	f
Receive Numeric Variable	ReceiveNumVariable (<i>Endian mode, Number of Bytes, Put in, Communication Handle</i>)	f
Receive Pointer Table	ReceivePtrTable (<i>Length, Start at Index, Of Table, Communication Handle</i>)	f
Receive String	ReceiveString(<i>Put In, Communication Handle</i>)	f
Receive String Table	ReceiveStrTable (<i>Length, Start at Index, Of Table, Communication Handle</i>)	f
Send Communication Handle Command	SendCommunicationHandleCommand(<i>Communication Handle, Command</i>)	f
Send Email	SendEmail(<i>Server Information, Recipients, Message Body</i>)	f
Send Email with Attachments	SendEmailWithAttachments (<i>Server Information, Recipients, Message Body, Attachment File Names</i>)	f
Set Communication Handle Value	SetCommunicationHandleValue(<i>Value, Communication Handle</i>)	p
Set End-Of-Message Terminator	SetEndOfMessageTerminator(<i>Communication Handle, To Character</i>)	p
Transfer N Characters	TransferNChars(<i>Destination Handle, Source Handle, Num Chars</i>)	f
Transmit Character	TransmitChar(<i>Character, Communication Handle</i>)	f
Transmit NewLine	TransmitNewLine(<i>Communication Handle</i>)	f
Transmit Numeric Table	TransmitNumTable (<i>Length, Start at Index, Of Table, Communication Handle</i>)	f
Transmit Pointer Table	TransmitPtrTable (<i>Length, Start at Index, Of Table, Communication Handle</i>)	f
Transmit/Receive Mistic I/O Hex String ^{1, 4}	TransReceMisticIoHexStringWithCrc (<i>Hex String, On Port, Put Result in</i>)	f
Transmit/Receive String	TransmitReceiveString(<i>String, Communication Handle, Put Result in</i>)	f
Transmit String	TransmitString(<i>String, Communication Handle</i>)	f
Transmit String Table	TransmitStrTable (<i>Length, Start at Index, Of Table, Communication Handle</i>)	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(<i>File System Type</i>)	f
Get Control Engine Address	GetControlEngineAddress()	f
Get Control Engine Type	GetEngineType()	f
Get Firmware Version	GetFirmwareVersion(<i>Put in</i>)	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(<i>On I/O Unit</i>)	p
Clear Counter	ClearCounter(<i>On Point</i>)	p
Clear Off-Latch	ClearOffLatch(<i>On Point</i>)	p
Clear On-Latch	ClearOnLatch(<i>On Point</i>)	p
Generate N Pulses	GenerateNPulses (<i>On Time (Seconds)</i> , <i>Off Time (Seconds)</i> , <i>Number of Pulses</i> , <i>On Point</i>)	p
Get & Clear Counter	GetClearCounter(<i>From Point</i>)	f
Get & Clear Off-Latch	GetClearOffLatch(<i>From Point</i>)	f
Get & Clear On-Latch	GetClearOnLatch(<i>From Point</i>)	f
Get & Restart Off-Pulse Measurement	GetRestartOffPulseMeasurement(<i>From Point</i>)	f
Get & Restart Off-Time Totalizer	GetRestartOffTimeTotalizer(<i>From Point</i>)	f
Get & Restart On-Pulse Measurement	GetRestartOnPulseMeasurement(<i>From Point</i>)	f
Get & Restart On-Time Totalizer	GetRestartOnTimeTotalizer(<i>From Point</i>)	f
Get & Restart Period	GetRestartPeriod(<i>From Point</i>)	f
Get Counter	GetCounter(<i>From Point</i>)	f
Get Frequency	GetFrequency(<i>From Point</i>)	f
Get Off-Latch	GetOffLatch(<i>From Point</i>)	f
Get Off-Pulse Measurement	GetOffPulseMeasurement(<i>From Point</i>)	f
Get Off-Pulse Measurement Complete Status	GetOffPulseMeasurementCompleteStatus(<i>From Point</i>)	f
Get Off-Time Totalizer	GetOffTimeTotalizer(<i>From Point</i>)	f
Get On-Latch	GetOnLatch(<i>From Point</i>)	f
Get On-Pulse Measurement	GetOnPulseMeasurement(<i>From Point</i>)	f
Get On-Pulse Measurement Complete Status	GetOnPulseMeasurementCompleteStatus(<i>From Point</i>)	f
Get On-Time Totalizer	GetOnTimeTotalizer(<i>From Point</i>)	f
Get Period	GetPeriod(<i>From Point</i>)	f
Get Period Measurement Complete Status	GetPeriodMeasurementCompleteStatus(<i>From Point</i>)	f
Off?	IsOff(<i>Point</i>)	f
Off-Latch Set?	IsOffLatchSet(<i>On Point</i>)	f
On?	IsOn(<i>Point</i>)	f
On-Latch Set?	IsOnLatchSet(<i>On Point</i>)	f
Set TPO Percent	SetTpoPercent(<i>To Percent</i> , <i>On Point</i>)	p
Set TPO Period	SetTpoPeriod(<i>To Seconds</i> , <i>On Point</i>)	p
Start Continuous Square Wave	StartContinuousSquareWave (<i>On Time (Seconds)</i> , <i>Off Time (Seconds)</i> , <i>On Point</i>)	p
Start Counter	StartCounter(<i>On Point</i>)	p
Start Off-Pulse	StartOffPulse(<i>Off Time (Seconds)</i> , <i>On Point</i>)	p
Start On-Pulse	StartOnPulse(<i>On Time (Seconds)</i> , <i>On Point</i>)	p
Stop Counter	StopCounter(<i>On Point</i>)	p
Turn Off	TurnOff(<i>Output</i>)	p
Turn On	TurnOn(<i>Output</i>)	p

Error Handling

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

Event/Reaction

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

High Density Digital Module

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Clear HDD Module Off-Latches ³	ClearHddModuleOffLatches(<i>I/O Unit, Module Number, Clear Mask</i>)	f
Clear HDD Module On-Latches ³	ClearHddModuleOnLatches(<i>I/O Unit, Module Number, Clear Mask</i>)	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(<i>Item, Bit to Set</i>)	f
Bit Shift	$x \ll nBitsToShift$	f
Bit Test	BitTest(<i>Item, Bit to Test</i>)	f
Bit XOR	$x \text{ bitxor } y$	f
Bit XOR?	See Bit XOR	f
Equal to Numeric Table Element?	$n == nt[0]$	f
Equal?	$x == y$	f
Flip Flop JK	FlipFlopJK(<i>Set [J], Reset [K], Output [Q]</i>)	p
Float to Int32 Bits	FloatToInt32Bits(<i>Server URL</i>)	f
Get High Bits of Integer 64	GetHighBitsOfInt64(<i>High Bits From</i>)	f
Get Low Bits of Integer 64	GetLowBitsOfInt64(<i>Integer 64</i>)	f
Greater Than Numeric Table Element?	$x > nt[0]$	f
Greater Than or Equal to Numeric Table Element?	$x \geq t[0]$	f
Greater Than or Equal?	$x \geq y$	f
Greater?	$x > y$	f
Int32 to Float Bits	Int32ToFloatBits(<i>nInt32</i>)	f
Less Than Numeric Table Element?	$x < nt[0]$	f
Less Than or Equal to Numeric Table Element?	$x \leq nt[0]$	f
Less Than or Equal?	$x \leq y$	f
Less?	$x < y$	f
Make Integer 64	MakeInt64(<i>High Integer, Low Integer</i>)	f
Move 32 Bits	Move32Bits(<i>From, To</i>)	p
NOT	not x	f
Not Equal to Numeric Table Element?	$n \neq nt[0]$	f
Not Equal?	$x \neq y$	f
NOT?	not x	f
Numeric Table Element Bit Clear	NumTableElementBitClear (<i>Element Index, Of Integer Table, Bit to Clear</i>)	p
Numeric Table Element Bit Set	NumTableElementBitSet (<i>Element Index, Of Integer Table, Bit to Set</i>)	p
Numeric Table Element Bit Test	NumTableElementBitTest (<i>Element Index, Of Integer Table, Bit to Test</i>)	f
OR	$x \text{ or } y$	f
OR?	See OR	f
Set Variable False	SetVariableFalse(<i>Variable</i>)	p
Set Variable True	SetVariableTrue(<i>Variable</i>)	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(<i>Variable</i>)	f
Variable True?	IsVariableTrue(<i>Variable</i>)	f
Within Limits?	IsWithinLimits(<i>Value, Low Limit, High Limit</i>)	f
XOR	$x \text{ xor } y$	f
XOR?	See XOR	f

Mathematical

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Absolute Value	AbsoluteValue(<i>Of</i>)	f
Add	$x + y$	f
Arccosine	Arccosine(<i>Of</i>)	f
Arcsine	Arcsine(<i>Of</i>)	f
Arctangent	Arctangent(<i>Of</i>)	f
Clamp Float Table Element	ClampFloatTableElement (<i>High Limit, Low Limit, Element Index, Of Float Table</i>)	p
Clamp Float Variable	ClampFloatVariable(<i>High Limit, Low Limit, Float Variable</i>)	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(<i>PID Loop</i>)	f
Get PID Gain	GetPidGain(<i>PID Loop</i>)	f
Get PID Input	GetPidInput(<i>PID Loop</i>)	f
Get PID Input High Range	GetPidInputHighRange(<i>PID Loop</i>)	f
Get PID Input Low Range	GetPidInputLowRange(<i>PID Loop</i>)	f
Get PID Max Output Change	GetPidMaxOutputChange(<i>PID Loop</i>)	f
Get PID Min Output Change	GetPidMinOutputChange(<i>PID Loop</i>)	f
Get PID Mode	GetPidMode(<i>PID Loop</i>)	f
Get PID Output	GetPidOutput(<i>PID Loop</i>)	f
Get PID Output High Clamp	GetPidOutputHighClamp(<i>PID Loop</i>)	f
Get PID Output Low Clamp	GetPidOutputLowClamp(<i>PID Loop</i>)	f
Get PID Scan Time	GetPidScanTime(<i>PID Loop</i>)	f
Get PID Setpoint	GetPidSetpoint(<i>PID Loop</i>)	f
Get PID Status Flags	GetPidStatusFlags(<i>PID Loop</i>)	f
Get PID Tune Derivative	GetPidTuneDerivative(<i>PID Loop</i>)	f
Get PID Tune Integral	GetPidTuneIntegral(<i>PID Loop</i>)	f
Set PID Configuration Flags	SetPidConfigFlags(<i>PID Loop</i> , <i>Configuration Flags</i>)	p
Set PID Feed Forward	SetPidFeedForward(<i>PID Loop</i> , <i>Feed Forward</i>)	p
Set PID Feed Forward Gain	SetPidFeedForwardGain(<i>PID Loop</i> , <i>Feed Fwd Gain</i>)	p
Set PID Forced Output When Input Over Range	SetPidForcedOutputWhenInputOverRange(<i>PID Loop</i> , <i>Forced Output</i>)	p
Set PID Forced Output When Input Under Range	SetPidForcedOutputWhenInputUnderRange(<i>PID Loop</i> , <i>Forced Output</i>)	p
Set PID Gain	SetPidGain(<i>PID Loop</i> , <i>Gain</i>)	p
Set PID Input	SetPidInput(<i>PID Loop</i> , <i>Input</i>)	p
Set PID Input High Range	SetPidInputHighRange(<i>PID Loop</i> , <i>High Range</i>)	p
Set PID Input Low Range	SetPidInputLowRange(<i>PID Loop</i> , <i>Low Range</i>)	p
Set PID Max Output Change	SetPidMaxOutputChange(<i>PID Loop</i> , <i>Max Change</i>)	p
Set PID Min Output Change	SetPidMinOutputChange(<i>PID Loop</i> , <i>Min Change</i>)	p
Set PID Mode	SetPidMode(<i>PID Loop</i> , <i>Mode</i>)	p
Set PID Output	SetPidOutput(<i>PID Loop</i> , <i>Output</i>)	p
Set PID Output High Clamp	SetPidOutputHighClamp(<i>PID Loop</i> , <i>High Clamp</i>)	p
Set PID Output Low Clamp	SetPidOutputLowClamp(<i>PID Loop</i> , <i>Low Clamp</i>)	p
Set PID Scan Time	SetPidScanTime(<i>PID Loop</i> , <i>Scan Time</i>)	p
Set PID Setpoint	SetPidSetpoint(<i>PID Loop</i> , <i>Setpoint</i>)	p
Set PID Tune Derivative	SetPidTuneDerivative(<i>PID Loop</i> , <i>Derivative</i>)	p
Set PID Tune Integral	SetPidTuneIntegral(<i>PID Loop</i> , <i>Integral</i>)	p

PID—Mistic

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

PID—Mistic (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Get Mistic PID Setpoint ^{1,4}	GetMisticPidSetpoint(<i>PID Loop</i>)	f
Set Mistic PID Control Word ^{1,4}	SetMisticPidControlWord(<i>On-Mask, Off-Mask, For PID Loop</i>)	p
Set Mistic PID D Term ^{1,4}	SetMisticPidDTerm(<i>To, On PID Loop</i>)	p
Set Mistic PID I Term ^{1,4}	SetMisticPidITerm(<i>To, On PID Loop</i>)	p
Set Mistic PID Input ^{1,4}	SetMisticPidInput(<i>Input, PID Loop</i>)	p
Set Mistic PID Mode to Auto ^{1,4}	SetMisticPidModeToAuto(<i>On PID Loop</i>)	p
Set Mistic PID Mode to Manual ^{1,4}	SetMisticPidModeToManual(<i>On PID Loop</i>)	p
Set Mistic PID Output Rate of Change ^{1,4}	SetMisticPidOutputRateOfChange(<i>To, On PID Loop</i>)	p
Set Mistic PID P Term ^{1,4}	SetMisticPidPTerm(<i>To, On PID Loop</i>)	p
Set Mistic PID Scan Rate ^{1,4}	SetMisticPidScanRate(<i>To, On PID Loop</i>)	p
Set Mistic PID Setpoint ^{1,4}	SetMisticPidSetpoint(<i>PID Loop, Setpoint</i>)	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(<i>Name, Pointer</i>)	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	DisableCommunicationToAllIoPoints()	p
Disable Communication to All I/O Units	DisableCommunicationToAllIoUnits()	p
Disable Communication to Event/Reaction ^{1,4}	DisableCommunicationToEventReaction (<i>Event/Reaction</i>)	p
Disable Communication to I/O Unit	DisableCommunicationToIoUnit(<i>I/O Unit</i>)	p
Disable Communication to Mistic PID Loop ^{1,4}	DisableCommunicationtoMisticPidLoop(<i>PID Loop</i>)	p
Disable Communication to PID Loop	DisableCommunicationtoPidLoop(<i>PID Loop</i>)	p
Disable Communication to Point	DisableCommunicationToPoint(<i>Point</i>)	p
Disable Event/Reaction Group ¹	DisableEventReactionGroup(<i>E/R Group</i>)	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(<i>Event/Reaction</i>)	p
Enable Communication to I/O Unit	EnableCommunicationToIoUnit(<i>I/O Unit</i>)	p
Enable Communication to Mistic PID Loop ^{1,4}	EnableCommunicationToMisticPidLoop(<i>PID Loop</i>)	p
Enable Communication to PID Loop	EnableCommunicationtoPidLoop(<i>PID Loop</i>)	p
Enable Communication to Point	EnableCommunicationToPoint(<i>Point</i>)	p
Enable Event/Reaction Group ^{1,4}	EnableEventReactionGroup(<i>E/R Group</i>)	p
Event/Reaction Communication Enabled? ^{1,4}	IsEventReactionCommEnabled (<i>Event/Reaction</i>)	f
Event/Reaction Group Communication Enabled? ^{1,4}	IsEventReactionGroupEnabled(<i>E/R Group</i>)	f
I/O Point Communication Enabled?	IsIoPointCommEnabled(<i>I/O Point</i>)	f
I/O Unit Communication Enabled?	IsIoUnitCommEnabled(<i>I/O Unit</i>)	f
IVAL Set Analog Filtered Value	IvalSetAnalogFilteredValue(<i>To, On Point</i>)	p
IVAL Set Analog Maximum Value	IvalSetAnalogMaxValue(<i>To, On Point</i>)	p
IVAL Set Analog Minimum Value	IvalSetAnalogMinValue(<i>To, On Point</i>)	p
IVAL Set Analog Point	IvalSetAnalogPoint(<i>To, On Point</i>)	p
IVAL Set Counter	IvalSetCounter(<i>To, On Point</i>)	p
IVAL Set Frequency	IvalSetFrequency(<i>To, On Point</i>)	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 Mystic PID Control Word ^{1,4}	IvalSetPidControlWord(On Mask, Off Mask, For PID Loop)	p
IVAL Set Mystic 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(<i>From Value, To String, Start Index, Width, Endian Type, Put Result In</i>)	f
Pack String into String	PackStringIntoString(<i>From Value, To String, Start Index, Width, Data Type, Put Result In</i>)	f
Set Nth Character	SetNthCharacter(<i>To, In String, At Index</i>)	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(<i>String, Option</i>)	f
Unpack String	UnpackString(<i>From String, Start Index, Width, To Value, Data Type, Endian Type, Put Result In</i>)	f
Verify Checksum on String	VerifyChecksumOnString(<i>Start Value, On String</i>)	f
Verify Forward CCITT on String	VerifyForwardCcittOnString(<i>Start Value, On String</i>)	f
Verify Forward CRC-16 on String	VerifyForwardCrcl6OnString(<i>Start Value, On String</i>)	f
Verify Reverse CCITT on String	VerifyReverseCcittOnString(<i>Start Value, On String</i>)	f
Verify Reverse CRC-16 on String	VerifyReverseCrcl6OnString(<i>Start Value, On String</i>)	f

Time/Date

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Copy Date to String (DD/MM/YYYY)	DateToStringDDMMYYYY(<i>String</i>)	p
Copy Date to String (MM/DD/YYYY)	DateToStringMMDDYYYY(<i>String</i>)	p
Copy Time to String	TimeToString(<i>String</i>)	p
Convert Date & Time to NTP Timestamp	DateTimeToNtpTimestamp(<i>Date&Time, NTP Timestamp, Put Result in</i>)	f
Convert NTP Timestamp to Date & Time	NtpTimestampToDateTime(<i>Date&Time, NTP Timestamp, Put Result in</i>)	f
Get Date & Time	GetDateTime(<i>Table</i>)	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(<i>Configuration, Description</i>)	f
Get Time Zone Offset	GetTimeZoneOffset(<i>Configuration</i>)	f
Get Year	GetYear()	f
Set Date	SetDate(<i>To</i>)	p
Set Day	SetDay(<i>To</i>)	p
Set Hours	SetHours(<i>To</i>)	p
Set Minutes	SetMinutes(<i>To</i>)	p
Set Month	SetMonth(<i>To</i>)	p
Set Seconds	SetSeconds(<i>To</i>)	p
Set Time	SetTime(<i>To</i>)	p
Set Time Zone Configuration	SetTimeZoneConfiguration(<i>Configuration</i>)	f
Set Year	SetYear(<i>To</i>)	p
Synchronize Clock SNTP	SynchronizeClockSNTP(<i>Timeout, Server URL, Put Result in</i>)	f

Timing

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

Timing (Continued)

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

