PAC Control™

Commands Quick Reference

Commands with an asterisk* are available only in PAC Control Professional. 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.

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*	`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*	`GetAnalogFilteredValue(From)`	F
Get Analog Maximum Value	`GetAnalogMaxValue(From)`	F
Get Analog Minimum Value	`GetAnalogMinValue(From)`	F
Get Analog Square Root Filtered Value*	`GetAnalogSquareRootFilteredValue(From)`	F
Get Analog Sqaure Root Value*	`GetAnalogSquareRootValue(From)`	F
Get Analog Totalizer Value	`GetAnalogTotalizerValue(From)`	F
Ramp Analog Output	`RampAnalogOutput(Ramp Endpoint, Units/Sec, Point to Ramp)`	P
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
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	`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*	`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

Control Engine (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
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 Handling (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
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

I/O Unit

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Clear I/O Unit Congfigured 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*	`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*	`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*	`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

I/O Unit—Memory Map (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
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 Clear	`BitClear(Item, Bit to Clear)`	F
Bit Change	`BitChange(Set flag, Bit to Change, Output)`	P
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
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

Logical (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
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
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

Mathematical (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
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
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

PID—Ethernet (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
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

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*	DisableCommunicationToEventReaction (*Event/Reaction*)	P
Disable Communication to I/O Unit	DisableCommunicationToIoUnit(*I/O Unit*)	P
Disable Communication to Mistic PID Loop*	DisableCommunicationtoMisticPidLoop(*PID Loop*)	P
Disable Communication to PID Loop	DisableCommunicationtoPidLoop(*PID Loop*)	P
Disable Communication to Point	DisableCommunicationToPoint(*Point*)	P
Disable Event/Reaction Group*	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*	EnableCommunicationToEventReaction(*Event/Reaction*)	P
Enable Communication to I/O Unit	EnableCommunicationToIoUnit(*I/O Unit*)	P
Enable Communication to Mistic PID Loop*	EnableCommunicationToMisticPidLoop(*PID Loop*)	P
Enable Communication to PID Loop	EnableCommunicationtoPidLoop(*PID Loop*)	P
Enable Communication to Point	EnableCommunicationToPoint(*Point*)	P
Enable Event/Reaction Group*	EnableEventReactionGroup(*E/R Group*)	P

Simulation (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
Event/Reaction Communication Enabled?	IsEventReactionCommEnabled (*Event/Reaction*)	F
Event/Reaction Group Communication Enabled?*	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 Point	IvalSetAnalogPoint(*To, On Point*)	P
IVAL Set Counter	IvalSetCounter(*To, On Point*)	P
IVAL Set Frequency	IvalSetFrequency(*To, On Point*)	P
IVAL Set I/O Unit from MOMO Masks	IvalSetiOUnitfromMOMO(On Mask, Off Mask, On I/O Unit)	P
IVAL Set Mistic PID Control Word*	IvalSetPidControlWord(*On Mask, Off Mask, For PID Loop*)	P
IVAL Set Mistic PID Process Term*	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?*	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 += 'a';	P
Append String to String	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*	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*	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

String (Continued)

PAC Control Command	OptoScript Equivalent (Arguments)	Type
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
Trim String	TrimString(*String, Option*)	F
Test Equal Strings	See String Equal?	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
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