

Software Manual





NovaTech, LLC 13555 West 107<sup>th</sup> Street Lenexa, KS 66215 Phone (913) 451-1880 www.novatechautomation.com orion.support@novatechautomation.com orion.sales@novatechautomation.com

Data classification: Public use

© Copyright 2007- 2021 by NovaTech, LLC

All Rights Reserved. All NovaTech trademarks are owned by NovaTech, LLC. All other trademarks are the property of their respective owners.

The NovaTech Configuration Director (NCD) software may be installed on multiple computers as needed under the following conditions:

- The computers must be owned by the end user or its subsidiary.
- The NCD installation must be used for an Orion-related project.

All files installed by NCD are protected by copyright and may not be shared with any third party. By installing NCD on a computer, you agree to these terms and conditions.

#### Disclaimer

This manual contains information that is correct to the best of NovaTech's knowledge. It is intended to be a guide and should be used as such. It should not be considered a sole source of technical instruction, replacing good technical judgment, since all possible situations cannot be anticipated. If there is any question as to the installation, configuration, or use of this product, contact NovaTech, LLC at (913) 451-1880.

To ensure that the equipment described in this User's Manual, as well as all equipment connected to and used with it, operates in a satisfactory and safe manner, all applicable local and national codes that apply to installing and operating the equipment must be followed. Since these codes can vary geographically and can change with time, it is the user's responsibility to determine which codes and standards apply, and to comply with them.



# Failure to follow the instructions provided in this manual, and/or failure to comply with applicable codes and safety standards can result in damage to this equipment, damage to connected devices, and/or serious injury to personnel.

All links to external websites have been verified as correct and appropriate at the time of the publication of this document. However, these links and websites, being outside of NovaTech LLC's control, are subject to change and may no longer be correct. In this case, please contact:

orion.support@novatechautomation.com



The documentation for the Orion products is structured as follows.

Manual name (see cover page of each manual)	Purpose
Quick Startup Guide	Describes out-of-the-box setup for quick installation.
User manuals: • OrionLX/OrionLX+ • OrionLXm • Orion I/O • OrionMX	<ul> <li>Description of Orion hardware and hardware options.</li> <li>List of software protocol options.</li> <li>NovaTech Configuration Director (NCD) software description:         <ul> <li>Installation of software on PC</li> <li>Information on running NCD</li> </ul> </li> <li>User interface information for monitoring and maintenance:         <ul> <li>Using the Orion Webpage</li> <li>Using MMI console</li> </ul> </li> <li>Setup and operation of the video option (-MMB for OrionLX. MMC for OrionLX+)</li> </ul>
Orion Applications Manual	Specific setup and operation of Orion features, such as firewall, SFTP access, installation of SSL certificate, access to Orion's SQL database with Microsoft Access® and Excel®, and others.
Protocol/Software Manuals	Software manuals explain all aspects of setup and operation of protocols such as DNP3 client and software options (Archive, Logic, DA Logic, etc.)
Tech Notes	Tech Notes provide solutions for general integration, such as scaling or setup of RS-485 networks.
Field Instructions	Field Instructions provide step-by-step instructions for installation of new hardware or software in the field.

See also <u>Appendix E – Additional References</u> for a list and locations of the recommended reference manuals.



#### Styles and Symbols

In this document, fonts, text styles and symbols are used to distinguish standard text from keyboard input, program text, GUI messages, and hyperlinks as follows. Warnings and safety notices are indicated with ANSI symbols.

Displayed text or symbol	Description
This is normal text.	Standard text.
See Port Options	Hyperlink to text in same document.
www.novatechautomation.com	Hyperlink to website.
orion.support@novatechautomtion.com	Clicking this link starts email client on the PC.
See OrionLX User Manual	Document name.
Minimum value	Menu item or text displayed by software.
Name of the data point	Text to be entered in input field or window.
Save	GUI button to be clicked.
if frequency < $60.0$ then	Program code.
<enter>, <ctrl>+<g>, <g></g></g></ctrl></enter>	Key to be pressed.
	This yellow triangle indicates a warning that must be observed by the users in order to avoid possible equipment damage or personal injury.
	This yellow triangle indicates an electrical hazard.
	Electrostatic sensitive device requires proper handling and grounding procedures to avoid equipment damage.
	DANGER indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.
	WARNING indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.
	CAUTION indicates a hazardous situation which, if not avoided, may result in minor or moderate injury.

Note that depending on the Windows<sup>®</sup> display settings on the computer running NCD, some of the screen shot details may appear differently than shown in this manual. The screen shots in this manual have been taken using Windows 10<sup>®</sup>.

## A NovaTech.

## Advanced Math & Logic for Orion

#### Table of Contents

I.	Introduction	11
II.	Setting Up Logic in NCD	12
III.	Editing Logic in NCD	13
	Introduction	13
	Port Options	14
	Logic Editor	14
	Standard Tab	16
	Inputs Menu	17
	Outputs Menu	19
	Logic Inputs Menu	21
	Logic Outputs Menu	23
	Events Menu	25
	Commands Menu	27
	Functions Tab	28
	Syntax Checking	28
IV.	Logic Editor Tools	30
	Settings	31
	Right Click Menu	32
	Advanced Menu	33
	Window Properties Dialog	34
v.	Logic Simulator Tools	36
	Diagnostic Output	38
	Log Output	38
	Spy List	39
	Forcing Values and Attributes	39
VI.	Logic Examples	41
	Single Comm Fail Point for Multiple Field Devices	41
	Automatic Switchover Between Values from Primary and Secondary Devices	46
	Simple Calculation	51
	Time-Delayed Output Operation	54
Ap	pendix A – List of Advanced Math & Logic Commands	57
	Programming Elements	57
	local Variable Declaration	57
	11 Conditional Statement	58 58
	while Loop	59
	repeat Loop	59
	function Function	60

## A NovaTech.

## Advanced Math & Logic for Orion

Database Operations	52 52 53 54 65
Logging Operations	36 36 66
Logic Operations	37
==	37
>	37
>=6	37
<	<u>3</u> 7
<=	37
~=	57
and	58
or	59
Math Operations7	70
+	70
7	70
*	70
7	70
% 	70
	70
math.abs	71
math.asin	71
math.acos	71
math.atan	70
orion Int	72
math log	73
math log10	73
math now	73
math sin	73
math.cos	73
math.tan	73
math.sort	74
math.rad	74
math.deg7	74
math.ceil7	75
math.floor7	75
math.min7	75
math.max7	76
math.exp7	76
math.pi7	76
bit.band	77
bit.bor	77
bit.bxor	"
String Operations	78
string.byte7	78
string.char	78
string.find	79
"	79
#	30
string.ien	50

## A NovaTech

## Advanced Math & Logic for Orion

:	string.sub	. 81
9	string.lower	. 81
:	string.upper	.82
(	prion.StrComp	82
0	prion. I rim	
(	prion.inStr	
0	Drion.Join	.83
0	Drion.Left	
	Dion.wid	04
Time	er Operations	86
9	prion English Timer	.00
	Dion.Enable i linei	.00
	prion DisableTimer?	.00
	arion EnableTimer2	
	prion CreateTimer2	87
Time		
TIME	e Operations	
9	Drion.GetDay	
	prion GetMinute	.00
	Dion.GetMonth	
	prion GetSecond	.03
	prion GetVear	.03
	arian GetDOW	0.05
		00
	arion SystemTime	90
A 1		04
Alar	ming Operations	
	Dion.GetonacknowledgedAllarme	
Port	Operations	
(	Drion.ResetPortCounters	.93
Obje	ects and Object Operations	94
(	prion.point	94
:	get	94
:	Set	
0	prion.timer	.96
:	GISADIE	
-	enaple	.96
	Drion.gevice	9/
	yel	
	Set	
0	JIIIII.itu	.99
-	yel arian nallaraun	100
	not	100
	you	101
	- On avatiana	400
wak	e Operations	102
Append	ix B – Tools and Buttons on the Input and Output Menus	103
1.	Display/Hide Tag Name List	103
2	Display/Hide Menu Tree	103
	Search Point Name Lists	104
5.	Gearch i Ghit Manie Lists	104

## A NovaTech.

## Advanced Math & Logic for Orion

4.	Toggle Show Point Details	104
5.	Display/Hide Alias Name	105
6.	View Report	105
7.	Edit Common Attributes	105
8.	Font Size	107
Appendi	x C – Logic in .ncd and .lua File	108
Appendix	x D – Lua Quick Reference Libraries	109
The L	ua Cheat Sheet	109
R	eserved identifiers and comments	109
т	ypes (the string values are the possible results of base library function type ())	109
C	perators, decreasing precedence	109
A	ssignment and coercion	109
C	ontrol structures	109
F	unction definition	109
F	unction call	109
The I	Mathematical Library [math]	110
B	asic operations	110
E	xponential and logarithmic	110
Т	rigonometrical	
S	plitting on powers of 2	
P	seudo-random numbers	110
The S	String Library [string]	110
В	asic operations	110
C	haracter codes	
F	unction storage	
F	ormatting	110
The I	O Library	111
C	complete I/O	111
S	imple I/O	
S	tandard files and utility functions	111
The C	Dperating System Library [os]	111
S	ystem interaction	111
D	ate/time	111
The C	Drion Library [orion]	112
D	atabase Operations	112
L	ogging Operations	112
N	lath Operations	112
т	imer Operations	112
Т	ime Operations	112
Appendi	x E – Additional References	113

### List of Figures

Figure 1: Logic Setup in NCD	
Figure 2: Main Logic Window	
Figure 3: Port Options	14

## A NovaTech.

## Advanced Math & Logic for Orion

Figure 4: NCD Logic Editor	14
Figure 5: NCD Logic Editor Properties	15
Figure 6: Standard Tab Options	16
Figure 7: Inputs Menu	17
Figure 8: Outputs Menu	19
Figure 9: Logic Inputs Menu	21
Figure 10: Logic Outputs Menu	23
Figure 11: Events Menu	25
Figure 12: Event Type Selection	26
Figure 13: Commands Menu	27
Figure 14: Functions Tab	28
Figure 15: No Syntax Error.	29
Figure 16: Syntax Error Detected	29
Figure 17: Logic Editor Toolbar	30
Figure 18: Edit Settings Dialog Box	31
Figure 19: Right Click Menu	32
Figure 20: Advanced Menu	
Figure 21: Properties Menu – Color/Font tab	
Figure 22: Properties Menu – Language/Tabs tab	
Figure 23: Properties Menu – Keyboard tab	35
Figure 24: Properties Menu – Misc tab	35
Figure 25: Logic Simulator Toolbar	36
Figure 26: Logic Simulator	37
Figure 27: Logic Simulator Diagnostic Output	38
Figure 28: Logic Simulator Log Output	38
Figure 29: Logic Simulator Soy List	39
Figure 30: Logic Simulator Force Point Values Dialog	40
Figure 31: Create the CommEailEvent	42
Figure 32: Select the Comm Fail Points	.43
Figure 33: Create Logic Input Point for Result	
Figure 34: Setting un Comm Fail Logic	45
Figure 35: Create Event for Automatic Switchover	47
Figure 36: Select Primary/Secondary Input Points	48
Figure 37: Create Logic Input Point List	49
Figure 38: Automatic Switchover Logic	
Figure 39: Create Event for Calculation	51
Figure 40: Select Phase Current Input Points	52
Figure 41: Create Logic Point List	
Figure 42: Phase Calculation Logic	53
Figure 43: Create Events for Shutoff Logic	54
Figure 44: Select Device Input Point	55
Figure 45: Select Device Output Point	55
Figure 46: Shutoff Logic	56
Figure 47: Tools on Input, Output, Logic Input and Logic Output Menus	103
Figure 48: Filter Point Name List	104
Figure 49: Show Point Details	104
Figure 50: View Report	105
Figure 51: Edit Common Attributes	106
Figure 52: Edit Common Attributes - Details	106
Figure 53: Font Size Slider	107
-	



List of Tables

Table 1: Logic Options	14
Table 2: Standard Tab Descriptions	16
Table 3: Inputs Menu Parameters	
Table 4: Insert Button	18
Table 5: Outputs Menu Parameters	
Table 6: Outputs Menu Button	20
Table 7: Logic Inputs Menu Parameters	22
Table 8: Logic Inputs Menu Buttons	22
Table 9: Logic Outputs Tab Parameters	24
Table 10: Logic Outputs Tab Buttons	24
Table 11: Events Menu Buttons	
Table 12: Commands Menu Buttons	27
Table 13: Logic Editor Tools	
Table 14: Logic Editor Setting Parameters	31
Table 15: Right Click Menu	
Table 16: Right Click Menu	
Table 17: Logic Simulator Tools	37
Table 18: Logic Simulator Diagnostic and Log Tools	
Table 19: Logic Simulator – Force Point Value	

х

### NovaTech.

### I. Introduction

The Advanced Orion Math & Logic module provides the capability to perform math or logic operations on any input or output points in the Orion. The result of the operation can be written to a logic (virtual) input point which can be mapped to a server port and polled by a client station. The result can also be written to a logic (virtual) output point which can then be sent to a server device to execute a control.

Typical applications include the following.

- ORing Comm Fail points from multiple devices into a single Logic Comm Fail.
- Primary/Secondary decisions based on the Comm Fail point for the Primary and Secondary devices.
- Totaling of numeric values (kWh, A, etc.) received from multiple devices.
- Controlling outputs in server devices based upon input values.

The logic within an Orion is event-driven and consists of one or more functions. Each function typically handles a specific task, for example "Initialization", "Open Breaker", "Hourly Total", etc.

There are different types of events which trigger the execution of a logic function. Each function is set up to be triggered by a specific event. There can be multiple logic functions, each triggered by a different event.

- Load Event: The Load event (default is DefaultStartFunction) is triggered only once, i.e. when the Orion boots up.
- Timer Event: One or more timers can be set up to run at certain intervals, e.g. 100ms or 5 seconds. Every time the timer's value is reached, an event is generated which triggers the execution of the associated function(s).
- DataChange Event: Each point can trigger an event based on data Refresh or data changing outside of user deadband.
  - Refresh: Any point can be set up to trigger a DataChange event when the point is written to the real-time database. Such an event would then trigger the execution of the associated function(s).
  - DeadBand: Any point can be set up to trigger an event when the point's value changes outside of the deadband range from the previous value. Such an event would then trigger the execution of the associated function(s).

The logic is set up by using an editing window within NCD. Advanced Math & Logic uses the Lua programming language. NCD provides helpful command information and examples.



### II. Setting Up Logic in NCD

Logic is part of the Orion configuration, and is set up in NCD as follows. After opening the desired .ncd file (File, Open) in the NCD main menu, navigate to the Configure menu, then to Logic. Select Advanced Math & Logic (Figure 1) from the dropdown list.

* NovaTech Configuration Director		
File Edit Configure Communications	Window Help	
CrionLX_Demo.ncd	General Hardware Model Serial Ports Network NKI - Time Interface Add Ons Hardware I/O	
	Logic +	Math and Logic
		Advanced Math and Logic
		LogicPak
		DA
		IEC 61131-3

Figure 1: Logic Setup in NCD

**A** NovaTech<sub>®</sub>

### Advanced Math & Logic for Orion

### **III. Editing Logic in NCD**

### Introduction

Logic is set up and edited through the various elements on the <code>Standard</code> and <code>Functions</code> tabs as shown in the following window. This chapter will discuss all aspects and options. In the next chapter, several examples show how specific applications can be implemented using Advanced Math & Logic.

۱ 🗋 ا	ogic Client Port 124 - Advar	iced Math and Logic (OrionLX_Demo.ncd)*			_ • •
Port	Options: Delete Port Clos	se Port 🗋 📔 🗇 🕼 🖄 🖫 🎾		🔀 🛛 😤 Settings 🎽 Simulate	e Line: 1 💡
	<ul> <li>Standard</li> <li>Functions</li> </ul>	function DefaultStartFuncti	.on ( )		^
	Inputs	end			
	Logic Inputs				
	Logic Outputs				
	Events				
	▲ Commands				
	⊳ All ⊳ Control				
	Define				
	Evaluation				
	Loop				¥
	⊳ Math	<			>
	Drion	<ul> <li>No code errors detected</li> </ul>			
	b String	Inputs	Inputs (0 Ite	em(s) - Filter By: All Points);	nt 🛛 12
		🔎 <all ports=""> 🗸 📉</all>	Search		Configured Inpu
		Point Name	Orion P	oint Name Event Event Ty	pe Deadband Mi
		Orion EstErr @Orion Internal			
		Orion MaxErr @Orion Internal			
		Orion PLL Freq @Orion Internal			
		Orion PLL Offset @Orion Internal			
		Orion PLL Status @Orion Internal			
		$\odot$			
		-			

Figure 2: Main Logic Window



### **Port Options**

In the upper left corner of each NCD display, the Port Options (Figure 3) menu is displayed.

Port Options: Delete Port Close Port 📋

Figure 3: Port Options

Command	Description
Delete Port	Removes the configuration, releasing the current logic. The logic is deleted and all associated points are deleted from the overall configuration.
Close Port	Closes the logic window to allow for the configuration of other ports. An * after the file name indicates that there are unsaved changes. Click on the into the main NCD window to save the configuration before closing.

Table 1: Logic Options

#### Logic Editor

The NCD Logic Editor (Figure 4) is a standard Windows-style text editor. It supports standard text functions such as Cut, Copy, Paste, Find, and Replace. In addition, it provides color-coding for certain logic statements and symbols.

The text functions, as well as the color settings and find/replace options, are made available by right-clicking within the NCD Logic window. The color settings, fonts, etc. can be customized by selecting Properties (Figure 4).

<pre>function DefaultStartFunction() end</pre>	Cut	^
	Сору	
<pre>function OneSecond_Timer()</pre>	Paste	
IncrementValue("HeartBeat_1 @Logic" IncrementValue("HeartBeat_2 @Logic"	Find Replace Go To	
end	Undo	
function IncrementValue(PointName)	Redo	
local val	Advanced >	
<pre>val = orion.GetPoint(PointName).val</pre>	Properties	
val = val + 1		¥
		>

Figure 4: NCD Logic Editor



In the example shown (Figure 5), the Keywords are set to be displayed in Blue color, Bold font style, and Courier New 11pt font.

Window Properties	×
Color/Font Language/Tabs Keyboard	i Misc Color: Background: Automatic ~ Font style: Bold ~
	Font: Courier New, 11pt
ОК	Cancel Apply

Figure 5: NCD Logic Editor Properties



#### **Standard Tab**

NCD provides the following Standard tab tree selections for logic setup. All other ports on the Orion should be configured prior to setting up Advanced Math & Logic so that the desired points are available for Advanced Math & Logic. The table below (<u>Table 2</u>) briefly describes each choice. Each selection is described in greater detail in paragraphs following the table below.



Figure 6: Standard Tab Options

Tab	Description					
Inputs	Inputs are defined on Orion client ports. Once the Orion client ports have been configured, they will appear in the point list.					
Outputs	Outputs are defined on Orion client ports. Once the Orion client ports have been configured, they will appear in the point list.					
Logic Inputs	Logic input points can be mapped to Orion server ports (i.e. one or more client stations can poll the value).					
Logic Outputs	Logic output points can be mapped to Orion client ports (i.e. the new value is sent to one or more server devices).					
Events	The events specify when (on startup, timer-based, or change-based) the functions run. One or more events can be specified.					
Commands	For each event, a specific function must be defined. The Commands selection provides syntax and examples for built in functions and logic operators. The functions/operators can be filtered by All, Control, Define, Evaluation, Loop, Math, Orion and String by selecting the appropriate tree element.					

Table 2: Standard Tab Descriptions



#### **Inputs Menu**

Under the Inputs menu (Figure 7), the Orion input points which are to be monitored by logic must be set up. The input points are polled from server devices through the Orion's client ports (such as DNP client, Modbus client, SEL client). This means that the input points to be monitored in logic must first be created in a client protocol.

All available points are listed in the  ${\tt Inputs}$   ${\tt List}$  box on the left.

Any input point to be used in logic must be highlighted by first selecting Inputs and then selecting the point in the Point Name list. Then move the point to the Configured Inputs box by clicking on the >> button. Likewise, points that are not needed in Configured Inputs can be moved out with << .



Figure 7: Inputs Menu

For a complete description of all available tools and buttons in this menu, see <u>Appendix B – Tools and</u> <u>Buttons on the Input and Output Menus</u>.

The following tables describe the input point parameters.

Parameter	Description
Event	All functions available in the Event list must first be set up under the Events menu as a DataChange event type. If an input point is linked to a DataChange function, it will call that function every time that input point is written to when configured as a Refresh event type, or when the value exceeds the deadband value when configured as a Deadband event type.
	Multiple input and/or output points can be assigned to the same event/function. This means that if any point(s) change, the corresponding function is executed. If an input point is configured with an event/function and is written to in a Timer event type, the Event field is generally left blank.
Event Type	Specifies when the DataChange event/function is executed. The value Refresh causes the function to be executed every time the input point is written to the real-time database. Deadband causes the function to be executed every time its real-time database value exceeds the deadband range. The DataChange function will be called when the input's online/offline status or string attribute change, regardless of the Refresh or Deadband event type.
Deadband	When the event type is Deadband and the current point value changes by more than the specified deadband value, an event is generated. Example: A deadband of 0.9 would detect a change of 0 to 1 (because 1 exceeds the deadband of 0.9), and trigger an event. In this example, if the point value changes from 0 to 0.5, no event would be generated because the value change is less than 0.9. Default: 10% of the point's min/max range.
Min Value Max Value	Each input point is set up with a minimum and maximum value. Points default to the min and max values defined on their client port. Internally, the Orion uses the current value of the point, and stores the normalized value, based on the minimum and maximum values <sup>1</sup> . Depending on the application, the minimum and maximum values might be desired to be different than the input point as used on the client port.
Queue Size	Specifies the number of events to be queued. When not 0, a per point queue holds events for retrieval using the orion.GetQueuedPoint command. orion.GetQueuedPoint allows logic to retrieve momentary changes, such as recloser operations.

Table 3: Inputs Menu Parameters

Button	Description
.↓	Inserts the associated point name from the Configured Inputs box at the current cursor position in the Logic editor (Figure 7).

Table 4: Insert Button

<sup>1</sup> For a complete discussion of scaling, refer to the Analog/Accumulator Scaling Technical Note.



#### **Outputs Menu**

Under the Outputs menu (Figure 8), the Orion output points which are to be monitored by logic must be set up. The output points generally receive their settings from other client station(s) to which the Orion is a server.

All available points are listed in the Outputs List on the left.

Any output point to be used in logic must be highlighted by first selecting Inputs and then selecting the point in the Point Name list. Then move the point to the Configured Inputs box by clicking on the >> button. Likewise, points that are not needed in Configured Outputs can be moved out with << .

Port Options: Delete Port Close Port Standard Function: Standard Function: Standar	Logic Client Port 124 - Adva	nced Math and Logic (OrionLX_Demo.ncd)*									
Inputs       function       function       function         Inputs       Inputs       end       end         Inputs       Inputs       end       end         Inputs       Inputs       end       end         Inputs       Inputs       end       end         Inputs       Inputs       Inputs       Inputs         Inputs       Inputs       Inputs       Input s         Inputs       Input s       Input s       Input s       Input s         Input s       Input s       Input s       Input s       Input s       Input s         Input s       Input s       Input s       Input s       Input s       Input s       Input s       Input s       Input s       Input s       Input s	Port Options: Delete Port Clo	ise Port 🗋 📋 🗖 🗽 🗇 🖫 🎽 🔑		🗡 🛛 🚰 Se	ttings 🎙	🔓 Simulate	Line: 1 Ch	aracte	r: 10		
Main       Contours detected       Control       Contro       Contro       Control </th <th></th> <th>function DefaultStartFuncti end</th> <th>on ()</th> <th></th> <th>curg- p</th> <th></th> <th></th> <th>1018-000</th> <th>. 10</th> <th></th> <th>~</th>		function DefaultStartFuncti end	on ()		curg- p			1018-000	. 10		~
● Orion       > String         ● Disorde errors detected         Outputs       Font         ● All Pota>         ● All Pota>         ● Conserved etected         ● Orion Point Name         Franker 1 @Feeder1         Reset Prack Dreaker 1 @Feeder1         RB1 @Feeder1         RB2 @Feeder1         RB3 @Feeder1         RB3 @Feeder1         RB4 @Feeder1         RB4 @Feeder1         RB5 @Feeder1         RB5 @Feeder1         RB5 @Feeder1         RB6 @	⊳ Math	٢							>		
Outputs         Outputs         Outputs         Outputs         Outputs         Outputs         Form         Configured Outputs           Point Name         Point Name         Point Name         Event         Point Name         Configured Outputs           Point Name         Point Name         Event         Point Name         Event         Point Name         Configured Outputs           Reset Prak Dreader 1         Reset Prak Dreader 1         Reset Prak Dreader 1         0.1         0         1         0           RB1 @Feeder 1         RB3 @Feeder 1         0.1         0         1         0         1         0           RB3 @Feeder 1         RB3 @Feeder 1         0.1         0         1	Drion	No code errors detected									
CAll Ports>       Configured Outputs         Point Name       Configured Outputs         Breaker 1 @Feeder1       Orion Point Name       Event Type       Deadband       Min       Max       Queue :         Breaker 1 @Feeder1       0.1       0       1       0       1       0         Reset Breaker Wear @Feeder1       RB1 @Feeder1       0.1       0       1       0       1       0         RB2 @Feeder1       0.1       0       1       0       1       0       1       0         RB3 @Feeder1       0.1       0       1       0       1       0       1       0         RB4 @Feeder1       0.1       0       1       0       1       0       1       0         RB4 @Feeder1       0.1       0       1       0       1       0       1       0         RB5 @Feeder1       0.1       0       1       0       1       0       1       0         RB5 @Feeder1       0.1       0       1       0       1       0       1       0         RB5 @Feeder1       0.1       0       1       0       1       0       1       0         RB5 @Feeder1 </td <td>b String</td> <td>Outputs</td> <td>0</td> <td>utputs (11 Item(s) - F</td> <td>ilter By</td> <td>: All Points)</td> <td></td> <td>Font</td> <td></td> <td></td> <td>12</td>	b String	Outputs	0	utputs (11 Item(s) - F	ilter By	: All Points)		Font			12
Point Name         Crion Point Name         Event Type         Deadband         Min         Max         Queue :           Breaker 1         @Feeder1          Breaker 1         0         1         0		🔎 <all ports=""></all>	<u>ہ</u>	Search	×	1	Configured (	Dutputs			
Breaker 1 @Feeder1         0.1         0         1         0         ^           Reset Breaker Vear @Feeder1         Reset Preak Demand @Feeder1         Reset Preak Demand @Feeder1         0.1         0         1         0         ^           Reset Preak Demand @Feeder1         RB2 @Feeder1         0.1         0         1         0         1         0         ^           RB1 @Feeder1         RB2 @Feeder1         0.1         0         1         0 <td< th=""><th></th><th>Point Name</th><th></th><th>Orion Point Name</th><th>Event</th><th>Event Type</th><th>Deadband</th><th>Min</th><th>Max</th><th>Queue</th><th>:</th></td<>		Point Name		Orion Point Name	Event	Event Type	Deadband	Min	Max	Queue	:
Reset Dreak Dreader1         C         RB1 @Feeder1         0.1         0         1         0           Rest Peak Dreamad @Feeder1          RB1 @Feeder1         0.1         0         1         0           RB2 @Feeder1          RB3 @Feeder1         0.1         0         1         0           RB3 @Feeder1           RB3 @Feeder1         0.1         0         1         0           RB4 @Feeder1           RB4 @Feeder1         0.1         0         1         0           RB5 @Feeder1           RB5 @Feeder1         0.1         0         1         0           RB5 @Feeder1           RB5 @Feeder1         0.1         0         1         0		Breaker 1 @Feeder1 ^		Breaker 1 @Feeder1			0.1	0	1	0	^
Reset Peak Demand @Feeder1         0.1         0         1         0           RB1 @Feeder1         0.1         0         1         0         1         0           RB2 @Feeder1         0.1         0         1         0         1         0           RB2 @Feeder1         0.1         0         1         0         1         0           RB3 @Feeder1         0.1         0         1         0         1         0           RB4 @Feeder1         0.1         0         1         0         1         0           RB5 @Feeder1         0.1         0         1         0         1         0           RB5 @Feeder1         0.1         0         1         0         1         0           RB5 @Feeder1         0.1         0         1         0         1         0		Reset Breaker Wear @Feeder1		RB1@Feeder1			0.1	0	1	0	
RB3 @Feeder1         0.1         0         1         0           RB3 @Feeder1         0.1         0         1         0         1         0           RB3 @Feeder1         0.1         0         1         0         1         0         1         0           RB3 @Feeder1         0.1         0         1         0         1         0         1         0           RB4 @Feeder1         0.1         0         1         0         1         0         1         0           RB5 @Feeder1         0.1         0         1         0		Reset Peak Demand @Feeder1		RB2 @Feeder1			0.1	0	1	0	
RB2 @Feeder1         0.1         0         1         0           RB3 @Feeder1         0.1         0         1         0           RB4 @Feeder1         0.1         0         1         0           RB5 @Feeder1         0.1         0         1         0           RB5 @Feeder1         0.1         0         1         0		RB1@Feeder1	11	RB3 @Feeder1			0.1	0	1	0	
RB3 @Feeder1         RB5 @Feeder1         0.1         0         1         0           RB5 @Feeder1         0.1         0         1		RB2 @Feeder1		RB4 @Feeder1			0.1	0	1	0	
R56 @Feder1         0.1         0         1         0           R56 @Feder1         0.1         0         1         0         V		RB3@Feeder1		RRF @Feeder1			0.1	0		0	
RB0@Feeder1 0.1 0 1 0 v		RB4 @Feeder1		RES @Feeder1			0.1	0	-	0	L
RB6 @Eaeder1		RB5@Feeder1		RB0@Feeder1			0.1	0		U	. ~
			<							>	

Figure 8: Outputs Menu

For a complete description of all available tools and buttons in this menu, see <u>Appendix B – Tools and</u> <u>Buttons on the Input and Output Menus</u>.



The following tables describe the output point parameters.

Parameter	Description
Event	All functions available in the Event list must first be set up under the Events menu with a DataChange event type. If an output point is linked to a DataChange function, it will call that function every time that output point is written to when configured as a Refresh event type, or when the value exceeds the deadband value when configured as a Deadband event type.
	Multiple input and/or output points can be assigned to the same event/function. This means that if any point(s) change, the corresponding function is executed. If an output point is configured with an event/function and is written to in a Timer event type, the Event field is generally left blank.
Event Type	Specifies when the DataChange event/function is executed. The value Refresh causes the function to be executed every time the output point is written to the real-time database. Deadband causes the function to be executed every time its real-time database value changes outside of the deadband range. The DataChange function will be called when the output's string attribute changes, regardless of the Refresh or Deadband event type.
Deadband	When the event type is Deadband and the current point value changes by more than the specified deadband value, an event is generated. Example: A deadband of 0.9 would detect a change of 0 to 1 (because 1 exceeds the deadband of 0.9), and trigger an event. In this example, if the point value changes from 0 to 0.5, no event would be generated because the value change is less than 0.9. Default: 10% of the point's min/max range.
Min Value Max Value	Each output point is set up with a minimum and maximum value. Points default to the min and max values defined on their client port. Internally, the Orion takes the current value of the point, and stores the normalized value, based on the minimum and maximum values <sup>1</sup> . However, depending on the application, the minimum and maximum values might be desired to be different than the output point as used on the client port.
Queue Size	Specifies the number of events to be queued. When not 0, a per point queue holds events for retrieval using the orion.GetQueuedPoint command. orion.GetQueuedPoint allows logic to retrieve momentary changes, such as recloser operations.

Table 5: Outputs Menu Parameters

Button	Description
.↓	Inserts the currently highlighted point name in the $Configured$ Outputs box at the current cursor position in the $Logic$ editor (Figure 8).

Table 6: Outputs Menu Button

<sup>1</sup> For a complete discussion of scaling, refer to the Analog/Accumulator Scaling Technical Note.



#### Logic Inputs Menu

The Logic Inputs menu (Figure 9) allows new virtual input points to be created for use in Advanced Math & Logic. Points added under the Logic Inputs menu can be used for the following two purposes.

- Logic input points can be mapped to any server port in the Orion. When a function writes the result of
  a math or a logic operation to a logic input point, the value can be polled by a client station connected
  to the Orion.
- The logic input points provide the capability to store internal logic values that can be synchronized between redundant Orions.

Logic Client Port 124 - Adva	nced Math and Logic (OrionLX_Demo.ncd)*						
Port Options: Delete Port Clo	ise Port 🗈 📔 🗂 🗗 🗽 🗇 🖫 🎽		🗡 🛛 🚰 Setti	ngs  🎽 Sin	nulate   Line: 1	Character:	10
Standard Functions     Inputs     Outputs     Logic Inputs     Logic Inputs     Logic Outputs     Logic Outputs     Events     Commands     Dall     Define     Evaluation     Loop     Math     Orion     String	function DefaultStartFu end • No code errors detected Inputs (1 Item(s) - Filter By: All Point Add new point P Search	<pre>s) </pre>	Configured Inpu	ts		Font	× > 12
	Orion Point Name	Event Event Type	Deadband Min	Max Q	eue Size		
	Average Current	1	1 0	1 0			
	$\odot$						

Figure 9: Logic Inputs Menu

For a complete description of all available tools and buttons in this menu, see <u>Appendix B – Tools and</u> <u>Buttons on the Input and Output Menus</u>.



The following tables describe the Logic Input point parameters.

Parameter	Description
Event	All functions available in the Event list must first be set up under the Events menu with DataChange event type. If a logic input point is linked to a DataChange function, it will call that function every time that input point is written to when configured as a Refresh event type, or when the value exceeds the deadband value when configured as a Deadband event type.
	Multiple input points and/or output points can be assigned to the same event/function. This means that if any point(s) change, the corresponding function is executed.
	If a logic input point is configured with an event/function and is written to in a Timer event type, the Event field is generally left blank.
Event Type	Specifies when the DataChange event/function is executed. The value Refresh causes the function to be executed every time the logic input point is written to the real-time database. Deadband causes the function to be executed every time its real-time database value changes outside of the deadband range.
	The DataChange function will be called when the input's online/offline status or string attribute change, regardless of the Refresh or Deadband event eype.
Deadband	If the current point value changes by more than the deadband value, an event is generated.
	Example: A deadband of 0.9 would detect a change of 0 to 1 (because 1 exceeds the deadband of 0.9), and trigger an event. In this example, if the point value changes from 0 to 0.5, no event would be generated because the value change is less than 0.9. Default: 10% of the point's min/max range.
Min Value Max Value	Each logic input point must be set up with a minimum and maximum value. The default values for discrete points are 0 and 1, and vary for analog points <sup>1</sup> .
Queue Size	Specifies the number of events to be queued. When not 0, a per point queue holds events for retrieval using the orion.GetQueuedPoint command. orion.GetQueuedPoint allows logic to retrieve momentary changes, such as recloser operations.

Table 7: Logic Inputs Menu Parameters

Button	Description
t.	Inserts the currently highlighted point name in the Configured Inputs box at the current cursor position in the Logic editor (Figure 9).
Add new point	Creates a new row in the Configured Inputs box for the user to enter the logic input point name.

### Table 8: Logic Inputs Menu Buttons

<sup>&</sup>lt;sup>1</sup> For a complete discussion of scaling, refer to the Analog/Accumulator Scaling Technical Note.



### Logic Outputs Menu

The Logic Outputs menu (Figure 10) allows new output points to be created for Advanced Math & Logic to write results to. Points added under this tab should be mapped to an Orion client port for the purpose of sending an output or control to an IED, PLC, etc. triggered from a logic function.

📋 Logic Client Port 124 - Adva	nced Math and Logic (OrionLX_Demo.ncd)*					- • ×
Port Options: Delete Port Clo	ise Port 🗋 📔 🗖 🖿 😭 📲 🛛 🔎		🗡 🛛 🚰 Settings 🎽 Simulate	Line: 1 C	haracter: 10	
Standard         Functions           Inputs         Inputs           Logic Inputs         Events           Events         Commands           > All         > Control           > Define         > Evaluation           > Loop         Ioop           > Math         > Orion	function DefaultStartFund end	ction()				× >
String	Outputs (0 Item(s) - Filter By: All Points	ts)			Font	12
	Add new point 📫 🔎 Search	X 🗎 🛛 🗉	Configured Outputs			
	Orion Point Name Event Event Ty	npe Deadband Min	Max Queue Size			

Figure 10: Logic Outputs Menu

For a complete description of all available tools and buttons in this menu, see <u>Appendix B – Tools and</u> <u>Buttons on the Input and Output Menus</u>.



The following tables describe the Logic Output point parameters.

Parameter	Description
Event	All functions available in the Event list must first be set up under the Events menu with a DataChange event type. If a logic output point is linked to a DataChange function, it will call that function every time that output point is written to when configured as a Refresh event type, or when the value exceeds the deadband value when configured as a Deadband event type.
	Multiple input and/or output points can be assigned to the same event/function. This means that if any point(s) change, the corresponding function is executed. If a logic output point is configured with an event/function and is written to in a Timer event type, the Event field is generally left blank.
Event Type	Specifies when the DataChange event/function is executed. The value Refresh causes the function to be executed every time the logic output point is written to the real-time database. Deadband causes the function to be executed every time its real-time database value changes outside of the deadband range.
	The DataChange function will be called when the string attribute changes, regardless of the Refresh or Deadband event type.
Deadband	If the current point value changes by more than the deadband value, an event is generated. Example: A deadband of 0.9 would detect a change of 0 to 1 (because 1
	point value changes from 0 to 0.5, no event would be generated because the value change is less than 0.9. Default: 10% of the point's min/max range.
Min Value Max Value	Each logic output point must be set up with a minimum and maximum value. The default values for discrete points are 0 and 1, and vary for analog points <sup>1</sup> .
Queue Size	Specifies the number of events to be queued. When not 0, a per point queue holds events for retrieval using the orion.GetQueuedPoint command. orion.GetQueuedPoint allows logic to retrieve momentary changes, such as recloser operations.

Table 9: Logic Outputs Tab Parameters

Button	Description
t.	Inserts the currently highlighted point name in the Configured Outputs box at the current cursor position in the Logic editor (Figure 9).
Add new point	Creates a new row in the Configured Outputs box for the user to enter the logic output point name.

Table 10: Logic Outputs Tab Buttons

<sup>1</sup> For a complete discussion of scaling, refer to the Analog/Accumulator Scaling Technical Note.



#### **Events Menu**

Under the Events menu, all events which will cause a function to be processed are defined. As new events are defined, the corresponding (still empty) functions are inserted into the logic editor. The functions must then be set up with the application-specific logic.

Functions triggered by the event type  ${\tt Timer}$  will automatically have  $\_{\tt Timer}$  added to their name in the Logic editor.

Logic Client Port 124	4 - Advanced	Math and Logic (Or	ionLX_Demo.ncd)*					
Port Options: Delete P	ort Close Po	nt 🗋   🗖 🗗 🚺	🕐 🗇 🗉 🐂	$\mathbf{P}$		🔀 🕴 🚰 Settings	🎽 Simulate	Line: 1 Character: 1
Standard Fu      Inputs     Outputs     Logic Inputs     Logic Inputs     Logic Outputs     Events     All     Commands     All     Control     Define     Evaluation	nctions	function I end function C end function E end	DefaultStart DneSecond_Ti Breaker123()	Function()				~
⊳ Loop ⊳ Math		No code errore det	ected					
⊳ Orion		No code enfors del	ected					
▷ String	C	reate New Event:		Create				
		Event Name	Event Type	Timer Interval	Timer Enable	d		
	0	OneSecond	Timer	1000	True			
		Breaker123	DataChange					
		-			L			
	(	0						

Figure 11: Events Menu



The following command buttons are available.

Button	Description
Create	Creates a new event. The Create button will be ebabled after a function name is entered in the Create New Event textbox. After clicking the Create button, a new row will be inserted with the Event Name entered. You can then change the Event Type to DataChange, Load, or Timer.
	Figure 12: Event Type Selection
	For event types DataChange and Load, no additional parameters are required. For event type Timer, the Timer Interval and Timer Enabled parameters are required. Timer Interval specifies in milliseconds how often the associated function is executed. The minimum value is 0ms, and the maximum value is 99,999,999ms. Timer Enabled specifies whether the timer is enabled (True or False) upon startup of the Orion. Timers can also be enabled/disabled in logic.
×	Deletes the event on that row. The associated function remains still in the $\tt Logic$ editor, but now has no event associated with it anymore.

Table 11: Events Menu Buttons

#### Data Change Events in Advanced Math & Logic

A function that is executed by a DataChange event can be associated to not just one but multiple data points. In this case, the function is executed if any one of these data points changes. The name of the data point that cause the execution can be obtained in the called function as follows.

```
function check_Values(strPointName)
    orion.PrintDiag("Execution triggered by the following point:")
    orion.PrintDiag(strPointName)
--other code to be executed
end
```



#### **Commands Menu**

The Commands menu lists common operators and functions. When one of these functions is highlighted (for is selected in the example below), the command parameters are listed in the lower right section with an example.

Port Options: Delete Port Close Port       Close Port       Close Port       Simulate       Line: 3       Character: 1 <ul> <li>Standard</li> <li>Functions</li> <li>Commands</li> <li>Control</li> <li>Define</li> <li>Evaluation</li> <li>Command: for</li> <li>Evaluation</li> <li>Control</li> <li>Define</li> <li>Evaluation</li> <li>Command: for</li> <li>Description: The Yor's tatement is a looping mechanism like the "while" statement. The "for" statement, can add a control while</li> <li>String</li> </ul> <li>Notation: The Yor's tatement is a looping mechanism like the "while" statement. The "for" statement, can add a control while also checking to see if it meets the end value. An absent atep value is assumed to be 1.</li> <li>String</li>	Logic Client Port 124 - Advanc	ed Math and Logic (OrionLX_Demo.ncd)*	
Standard Functions             • Standard Functions             • Inputs             • Inputs             • Logic Inputs             • Logic Inputs             • Control             • Control             • Evaluation             • Evaluation             • Define             • Evaluation             • Math             • Math             • String             • String	Port Options: Delete Port Close	Port 🗋   🗖 🖿 🕤 📱 📲   🔎 🛛 🗡   🚰 Settings 🕍 Simulate   Line: 3	Character: 1
Control     Define     Evaluation     Loop     Moth     Orion     String     String     String     Control     Control	Standard Functions     Inputs     Outputs     Logic Outputs     Events     Commands     All	function DefaultStartFunction() end	~
Evaluation     Loop     While     Math     Orion     String     String     String     No code errors detected     Command: for     Description: The 'for' statement is a looping mechanism like the "while" statement. The 'for' statement, unlike the "while" statement, can add a     counter while also checking to see if it meets the end value. An absent step value is assumed to be 1.     Surface: for Counter/Variables = clinial/Values, <endvalues <stepvalues]="" [.="" contents="" do="" end="" example:="" int="" intoop="" intoop<="" local="" oop="" td=""><td>Define</td><td>&lt;</td><td>&gt;</td></endvalues>	Define	<	>
Loop     Command: for     Description: The for statement is a looping mechanism like the "while" statement. The "for" statement, unlike the "while" sta	Evaluation	No code errors detected	
	▲ Loop For repeat/until while b Math b Orion b String	Command: for Description: The "for" statement is a looping mechanism like the "while" statement. The "for" statement, unlike the "while" statement Syntax: for <counter\viniteles <initialvalue="" =="">, <endvalue> [, <stepvalue>] do </stepvalue></endvalue></counter\viniteles>	ient, can add a

Figure 13: Commands Menu

Features	Description
Insert Example	Examples can be highlighted and either copied/pasted or dragged/dropped from the Example textbox to the Logic editor.
Insert Command	Operators and Functions can be inserted by dragging/dropping the command from the Commands menu to the Logic editor.

Table 12: Commands Menu Buttons

A complete list of commands, including a detailed description of all parameters, is included in <u>Appendix A</u> <u>– List of Advanced Math & Logic Commands</u>.



### **Functions Tab**

The Functions tab provides a list of functions in the Logic editor in alphabetic order. Clicking on the function name will display that function in the Logic editor.

Logic Client Port 124 - Advanced	Math and Logic (OrionL	X_Demo.ncd)*					- • •		
Port Options: Delete Port Close Po	ort 🗋 🕴 🗖 🔟 😭	M I 🗄 🖌	Ď		🔀 🕴 🚰 Settings 🏾 🎽 Sim	ulate Line: 1	Character: 1		
standard Functions     Breaker123()     DefaultStanfunction()     OneSeond_Timer()	<pre>runction Defaultstartrunction() end function OneSecond_Timer() end function Breaker123() end</pre>								
	<ul> <li>No code errors detection</li> </ul>	ted							
	Create New Event:		Creat	е					
	Event Name DefaultStartFunction	Event Type Load	Timer Interval	Timer Enabled	1				
	X OneSeond	Timer	1000	True					
	× Breaker123	DataChange							
	$\odot$								

Figure 14: Functions Tab

### Syntax Checking

NCD automatically checks syntax as code is entered in the Logic editor. The figure below shows an example with "No code errors detected" (syntax notification located between the Logic editor and the Inputs box).



	ancea matriana cogic (ononex_bembh	icoj								-	
et Options: Delete Port	lose Port 🗋 📔 🖸 🕼 🛣 🗇 🖫 🦉	#   🔑 🔄			🗡 🛛 😁 Setting	s 🎬 Sin	nulate Line	e: 6 Charact	ten 22		
Standard      Function	function DefaultSta	rtFuncti	on ()								•
Inputs	end										
Outputs											
Logic Inputs	function OneSeond_T	imer()									
Logic Outputs	local i					-					
Events	i = orion.GetPo	<pre>int("eth</pre>	0 act	ivit	y @Orion Interna	1").v	alue m	issing q	uote	5	
> All	end										
Control	6										
Define	function Breaker123	0									
Evaluation	ena										
b Loop					-						*
b Orion	< /										>
String	No code errors detected										
	Inputs			Input	s (1 Item(s) - Filter By:	All Poir	nts)	Font			12
	D (Al Pote)			Po	arch	× 🗈		found locute			
	P VERMAN		1								
				•	Drion Point Name	Event	Event Type	Deadband	Min	Max	Que
	Point Name								0	1	0
	Point Name Orion EstErr @Orion Internal	^	>>	•	th0 activity @Orion Internal			0.1	*		
	Point Name Orion EstErr @Orion Internal Orion MaxErr @Orion Internal	^	» «	<u>e</u> 6	th0 activity ⊜Orion Internal			0.1			
	Point Name Orion EstErr @Orion Internal Orion MaxErr @Orion Internal Orion PLL Freq @Orion Internal Orion PLL Offret @Orion Internal	^	» «	(±) e	th0 activity ⊜Orion Internal			0.1			
	Point Name Orion EstErr @Orion Internal Orion MaxErr @Orion Internal Orion PLL Freq @Orion Internal Orion PLL Offset @Orion Internal Orion PLL Status @Orion Internal		>>	• •	th0 activity @Orion Internal			0.1			
	Point Name Orion EstErr @Orion Internal Orion MaxErr @Orion Internal Orion PLL Freq @Orion Internal Orion PLL Status @Orion Internal Orion PLL Status @Orion Internal		» «	< e	th0 activity @Orion Internal			0.1			>

Figure 15: No Syntax Error

The figure below shows an example with an error because the point name was entered without quotes (syntax notification located between the Logic editor and the Inputs box).

et Options: Delete Port Clo	ise Port 🗋 🗌 🗗 🖉 🔛 🗐 📲 🔑	1			X Setting	s 📷 Sin	nulate Line	s 6 Charact	er: 75		_
Standard Infunctions     Imputs     Outputs     Logic Inputs     Logic Inputs     Logic Inputs     Logic Inputs     Logic Outputs     Events     Commands     Parlie     Control     Define     Evaluation     Loop     Math     Onion	function DefaultStartPun end function OneSecond_Timer( local i i = orion.GetPoint(e end function Breaker123() 1 Emor(s) ) yexpected new 'schicky (Line 6)	cti ) th0	on() acti	.vit	y @Orion Internal	.).val	ue mis	sing quot	tes		~
p sung	Inputs			Inpu	uts (1 Item(s) - Filter By	All Poir	nts)	Fent			- 52
	P (Al Pots> · X			P	Search	8	Con	figured Inputs			
	Point Name		1		Orion Point Name	Event	Event Type	Deadband	Min	Max	Que
	Orion EstErr @Orion Internal	1	[22]		eth0 activity @Orion Interna	N		0.1	0	1	0
	Orion MaxErr @Orion Internal	1000									
	Orion PLL Freq @Orion Internal		<<								
	Orion PLL Offset @Orion Internal										

Figure 16: Syntax Error Detected



### IV. Logic Editor Tools

NCD includes some common tools to aid in development. These tools allow you to maximize/restore the logic editor, undo/redo actions, comment/uncomment logic, search for text in the logic editor, and configure the logic editor. The following figure shows the Logic Editor Toolbar.

00000		🗡 🕴 🚰 Settings
-------	--	----------------

Figure 17: Logic Editor Toolbar

The following table describes each of the toolbar features.

Feature	Tool Tip	Description						
	Maximize code window	When this button is clicked, the left pane containing the tree and the bottom pane will be hidden to maximize the logic editor.						
٥	Restore code window	When this button is clicked, the left pane containing the tree and the bottom pane will be restored.						
	Undo	Undo the previous action. <ctrl>+<z> and <alt>+<backspace> are also supported.</backspace></alt></z></ctrl>						
2	Redo	Redo the previous undo action. <ctrl>+<y> and <ctrl>+<shift>+<z> are also supported.</z></shift></ctrl></y></ctrl>						
hui	Comment out selection	Comment out the current line or selected lines.						
	Uncomment selection	Uncomment the current selection.						
×	Search	Search for matching text in the logic editor.						
Provide the settings	Settings	Set online status, show line numbers, and set font and colors as described below.						

Table 13: Logic Editor Tools



### Settings

The Edit Settings button opens the  ${\tt Edit}$  Settings dialog box.



Figure 18: Edit Settings Dialog Box

Parameter	Description
Logic Inputs Online	When the Logic Inputs Online checkbox is checked, the Logic Input points will default to online status on startup. This setting defaults to checked.
Show Line Numbers	When the Show Line Numbers checkbox is checked, the editor will display line numbers on the left side of the logic editor. This setting defaults to unchecked.
Font	Allows the user to select the Font, Font Style and Size of the text editor.
Color Theme	Allows the user to select background and foreground colors of the text editor.

Table 14: Logic Editor Setting Parameters



### **Right Click Menu**

When you right click on the logic editor window, the context menu with editor features such as Copy/Paste, Find/Replace, and Undo/Redo is displayed. Additionally, the settings in Advanced and Properties allow you to change the look of the editor.



Figure 19: Right Click Menu

Menu Item	Description	
Cut/Copy/Paste	Cut removes the selected text from the logic editor and copies it to the clipboard. Copy leaves the selected text in the logic editor and copies it to the clipboard. Paste inserts the text from the clipboard at the current cursor position.	
Find/Replace/Go To…	Find displays the Find dialog, allowing you to use common search methods. Replace displays the Find & Replace dialog, allowing you to use common search and replace methods. Go To displays the Go To Line dialog allowing you to go to the entered line number. <ctrl>+<alt>+<f3> will also display the dialog.</f3></alt></ctrl>	
Undo/Redo	Undo the previous action. <ctrl>+<z> and <alt>+<backspace> are also supported. Redo the previous undo action. <ctrl>+<y> and <ctrl>+<shift>+<z> are also supported.</z></shift></ctrl></y></ctrl></backspace></alt></z></ctrl>	
Advanced	Displays the Advanced menu.	
Properties	Displays the Window Properties dialog.	

Table 15: Right Click Menu



#### **Advanced Menu**

The Advanced menu contains less common features including allowing you to see Whitespace (spaces and tabs), convert selected text to Upper/Lower case, Comment/Uncomment selected logic, Sort lines Ascending/Descending, as well as display the Set Repeat Count and Choose Command dialogs.



Figure 20: Advanced Menu

Menu Item	Description	
Show Whitespace	Enables/disables whitespace marking. When enabled, spaces are shown as bullets (•) and tabs are shown as double arrows (»). When disabled, spaces and tabs are shown as empty space. <ctrl>+<alt>+<t> will toggle between enabled/disabled. The default setting is disabled.</t></alt></ctrl>	
Make Uppercase	Makes the selected text all uppercase or lowercase. <ctrl>+<shift>+<u> will</u></shift></ctrl>	
Make Lowercase	convert the text to all uppercase. <ctrl>+<u> will convert the text to all lowercase.</u></ctrl>	
Comment Selection	Either comments out the current line/selected lines or uncomments the curren	
Uncomment Selection	selection.	
Sort Lines Ascending	Sorts the selected lines in ascending or descending order.	
Sort Lines Descending		
Set Repeat Count	Displays the Set Repeat Count dialog. The repeat count dialog prompts	
	the user for the number of times to repeat the next command.	
Choose Command	Displays the Choose Command dialog, which allows you to use all commands supported by the text editor.	

Table 16: Right Click Menu



### **Window Properties Dialog**

When you right click on the logic editor and select the Properties... menu option, the Window Properties dialog will be displayed. The Color/Font tab allows you to set the colors and fonts for various editor settings.

olor/Font	Language/Tabs	Keyboard	Misc	
⊡. Misc		^	Color:	
Bo	okmarks			~
Bra	ace Highlights		D and care unde	
···· Ho	rz Divider Lines		Backyrounu.	
Le	ft Margin			~
- Lin	e Numbers		Font style:	
···· ve	nt Divider Lines			
⊡lua	ndow			
Att	ribute Block			
Blo	ock Comments			
Hy	perlinks			
Ke	ywords		Font:	
···· Lin	e Comments	~	Courier New, 11pt	

Figure 21: Properties Menu – Color/Font tab

The Language/Tabs tab allows you to configure the Lua language logic editor settings including adjusting case while typing, indentation, statement completion and tab parameters.

Window Properties	×
Color/Font Language/Tabs Keyboard Language: Lua ✓ Adjust text case while typing language keywords Auto indentation style ○ Off ④ Follow language scoping ○ Copy from previous line	Misc Statement Completion Auto list members Parameter information Tabs Tab size: 4 Convert tabs to spaces while typing
ОК	Cancel Apply

Figure 22: Properties Menu – Language/Tabs tab



The  ${\tt Keyboard}$  tab allows you to view and modify the key assignments for editor commands.

Window Pro	operties				×
Color/Font	Language/Tabs	Keyboard	Misc		
Command:			Key Assigr	nments:	
AppendNe Bookmark Bookmark Bookmark Bookmark Bookmark CaseSens CharLeft CharLeft CharLeft CharLeft	extCut ClearAll JumpToFirst JumpToLast Next Prev Toggle titiveOff titiveOff xtend AtendColumn	~	Assign New key a	Remove assignment:	Reset
1					_
		ОК	(	Cancel	Apply

Figure 23: Properties Menu – Keyboard tab

The  ${\tt Misc}$  tab allows you to set the following miscellaneous editor settings.

Color/Font Language/Tabs Keyboard	d Misc
Settings	Max undoable actions
Smooth scrolling	<ul> <li>Unlimited</li> </ul>
Line tooltips on scroll	O Limited to:
Enable virtual space	
Color syntax highlighting	Line numbering
Show horizontal scrollbar Show vertical scrollbar	Style: <none> ~</none>
One click URL Navigation	Start at: 1
Copy RTF text to clipboard Brace matching	Track selection in bold

Figure 24: Properties Menu - Misc tab



### V. Logic Simulator Tools

NCD includes a logic simulator to test code during development. The logic simulator allows you to start/pause code, view/force data values, set/remove breakpoints, step into/out of/over code and view call stack/variables. The simulator icons will appear when the Simulate button is clicked, which will start simulation, but break on the first executable line of code. Clicking the resume button  $\blacktriangleright$  will start running the code.

📷 End Simulate 🛛 🞄 ၊ 🖗 🕨 🛛 🖙 🗊 🗊 🖓 XY 🛛 STEPPING (Line: 15)
---

Figure 25: Logic Simulator Toolbar

The following table describes each of the toolbar features.

Feature	Tool Tip	Description
🕍 Simulate	Simulate	Starts logic simulation. The simulator will break at the first line of code in the $Load$ (DefaultStartFunction) function to allow debug of the Load function. Clicking the Resume button will run the logic until it reaches a user breakpoint (if present).
🎽 End Simulate	End Simulate	Ends logic simulation. Simulator logic values will be reset to zero when the next simulation session is started.
<u></u>	Toggle Spy List	Display or hide the Spy List. The Spy List allows you to drag in points from Inputs, Outputs, Logic Inputs and Logic Outputs. This can be usefully when working on logic that utilizes multiple points that cannot be displayed on the screen at the same time.
M	Toggle Breakpoint	Toggles a breakpoint. After clicking on the desired line in the logic editor, click this button to add a breakpoint if no breakpoint is present or remove a breakpoint if a breakpoint is present.
<b>"</b>	Remove all Breakpoints	Removes all breakpoints in logic.
	Resume	Starts or resumes logic solving. This button must be clicked when the Simulate button is pressed to start logic and after a breakpoint stops to resume logic solving.
II	Break on next executable code	Breaks logic solving on the next line of executed code. Depending upon Timer and DataChange events, the break could cause logic solving to halt on a line of code immediately or as long as it takes for a Timer to run or a DataChange event to occur.
FI	Step Into Function	Steps into a function. When this button is clicked, the logic will advance to the next line of code. If the current line of code is calling a function, the next line will be in that called function. This button is only enabled when logic is stepping (at a breakpoint).
Ç≣	Step Over Function	Steps over a function. When this button is clicked, the logic will advance to the next line of code in the current function, even if the current line of code is calling a function. This button is only enabled when logic is stepping (at a breakpoint).
## Advanced Math & Logic for Orion

Feature	Tool Tip	Description
ځ	Step Out of Function	Steps out of a function. When this button is clicked, the logic will continue execution of the current function until it returns to its caller. When using this button in a Timer or DataChange event, the code will continue to run until the next breakpoint (if present). This button is only enabled when logic is stepping (at a breakpoint).
6 <sup>2</sup>	View Call Stack	Opens a dialog box that displays the call stack. This is an advanced debugging feature that displays the functions that are currently on the call stack by order of how they were called. The first function in the list is the currently executed function.
ХҮ	View Variables	Opens a dialog box that displays global and local variables for the current function.
RUN	Run	When RUN is displayed, the logic is currently running.
STEPPING (Line: 2)	Stepping Code	When STEPPING is displayed, the logic is at a breakpoint indicated by the line number.

Table 17: Logic Simulator Tools

Port Options: Delete Port Close Port    Standard  Functions	0   C	1 07 NC 201 73 75 4	🔑		2	X I I	Settin	ias 🎏 End Sim	ulate	a 🛋	🕰 💽 🖬 🖉	e Calde	- P. ww	STEPPING (User 17	1
Standard      Functions  Inputs Outputs	ence enc	d							ionore.		×		= (c) A1	STEPPING (Line: 17)	Line:
Logic Uputs Logic Cuputs Events 6 Cannands 9 Cathol 9 Evension 9 Evension 9 Evension 9 Evension 9 Evension 9 Orion 9 Siving 10 Debug Disposito Cutput Log Output	fur enc fur	IncrementVa IncrementVa d motion IncrementVa val = orion val = orion val = val + orion.Pr val = 0 orion.Pr end	GetPo GetPo intLag("F 1 6 ag(I 00000 f intLag()	imer() HeartBeat Hue(PointH Point(Point Point(PointName Chen Dog(PointName Hame = Po:	_1 @Logic 2 @Logic Name) tName).va " Val ame " intName,	") ") ue " Roll valu	7 over'	ral) ') ral, onlin	e = t	rue})					^
	No or	ode errors detected													
	Inputs (	(2 Item(s) - Filter E	y: All P	oints)										Font	12
	Add new	v point 💠 🔎 Search			🗙 📑 🗹 🖩	6	Configure	d Inputs							
	4	Orion Point Name	Event	Event Type	Deadband	Min	Max	Queue Size	Value	Online	Changes	String	Date	Time	
	<u></u>	HeartBeat_1			.1	0	32767	0	6	TRUE	6		2/1/2021	10:43:13.778 PM	
		Hand David 2			1	0	32767	0	5	TRUE	5		2/1/2021	10:43:09 858 PM	

Figure 26: Logic Simulator

The figure above shows the Advanced Math & Logic window during simulation. In this screen capture, the logic is at a breakpoint on line 17 and the mouse is hovering over the variable val, which is displaying a value of 6. The value will only be displayed on mouse over when the logic is at a breakpoint. The Logic Inputs menu at the bottom is displaying the real-time values and attributes in blue background. The values and attributes can be forced by double clicking on any of the blue background attributes to bring up the Force Point Values dialog. When viewing multiple points that aren't in the same point type (Input, Output, Logic Input, Logic Output), the Spy List allows you to drag/drop points of interest.



Debug menus are available in the Standard tree on the left. The Debug menus allow you to view the orion.PrintDiag(Diagnostic Output) and orion.PrintLog(Log Output) output. The following figures provide a sample of each type of output. Buttons are provided to sort the output in ascending/descending order, copy contents to the clipboard, save contents to a file, delete contents, and to float the window to another location.

### **Diagnostic Output**

The simulator provides the following diagnostic output.

	×
(19:39:49:863)	HeartBeat_1 @Logic Value 4986.0
(19:39:50:878)	HeartBeat_1 @Logic Value 4987.0
(19:39:50:878)	HeartBeat_2 @Logic Value 5020.0
(19:39:51:893)	HeartBeat_1 @Logic Value 4988.0
(19:39:52:905)	HeartBeat_1 @Logic Value 4989.0
(19:39:53:907)	HeartBeat_1 @Logic Value 4990.0
(19:39:54:920)	HeartBeat_1@Logic Value 4991.0

Figure 27: Logic Simulator Diagnostic Output

### Log Output

The simulator provides the following log output.

2/1/2021 10:37:44 PM	'StepInto' overridden to '1'
2/1/2021 10:37:45 PM	Logic started.
2/1/2021 10:57:41 PM	HeartBeat_1 @Logic Rollover
2/2/2021 3:27:00 PM	HeartBeat_2 @Logic Rollover
2/2/2021 3:27:33 PM	HeartBeat_1 @Logic Rollover
2/2/2021 6:15:22 PM	HeartBeat_2 @Logic Rollover
2/2/2021 6:15:55 PM	HeartBeat_1 @Logic Rollover

Figure 28: Logic Simulator Log Output

Button	Tool Tip	Description
	Toggle showing newest items at top or bottom of list	Sorts the output in ascending or descending order.
	Copy contents of log to clipboard	Copies the entire contents of the output to the clipboard.
	Save contents of log to a file	Saves the entire contents of the output to a file.

## Advanced Math & Logic for Orion

Button	Tool Tip	Description
×	Delete contents of log window	Deletes the entire contents of the output.
	Float window	Floats the window so it can be moved to another location. The window will no longer be constrained to the NCD container. Both the Diagnostic Output and Log Output can be floated. To return the output to the NCD container, click the down arrow (

Table 18: Logic Simulator Diagnostic and Log Tools

### Spy List

The Spy List allows you to view specific data values for the logic you are testing. This is useful when the data points are on different menus (Input, Output, Logic Input, Logic Output) or if the points cannot be sorted to display without scrolling. To add points to the Spy List, drag and drop from the appropriate menu. Real-time attributes are displayed with a blue background. Attributes can be forced by double clicking on any of the blue background attributes to bring up the Force Point Values dialog. To remove a point from the Spy List, click the red X next to the Orion Point Name.

**	◆ Spy List Drag/drop from configured point lists to this wi – □ ×											
Inp	Inputs/Outputs (4 Item(s) - Filter By: All Points)											
🔎 Search												
	Orion Point Name	Value	Online	Changes	String	Date	Time					
$\times$	HeartBeat_1 @Logic	0	TRUE	0		2/3/2021	4:06:20.205 PM					
$\sim$	HeartBeat_2 @Logic	0	TRUE	0		2/3/2021	4:06:20.205 PM					
$\times$	Output_1 @Logic	0	FALSE	0								
$\times$	Output_2 @Logic	0	FALSE	0								

Figure 29: Logic Simulator Spy List

#### **Forcing Values and Attributes**

Values and attributes can be forced to test logic. To force, double click on any value or attribute with a blue background. The following dialog will display, allowing you to force multiple values or attributes at one time. Check the box next to the desired values and attribute(s) to force. Depending on the Timed checkbox, values and attributes will either be forced for a duration or a single shot. Logic cannot write to values or attributes when they are time forced. Outputs and Logic Outputs cannot be time forced since they are always single shot.

# **NovaTech**

# Advanced Math & Logic for Orion

✤ Force Point Values (	Input)	×
Only selected attribute	es will be set.	Timed 🔽
	Time to r	emain forced (sec): 60
Point Name: HeartBea	t_1 @Logic	
Value:	9147	
Online:	✓	
Changes:	49176	
String:		
Date/Time (UTC):	4:03:25 PM	* *
	2/3/2021	15
	Remove Forced Value	Force Value Cancel

Figure 30: Logic Simulator Force Point Values Dialog

Feature	Description
Timed	Enables a timed forced attribute(s). After the Time to remain forced time, the attribute(s) will return to the contents before the force. Attributes that have been forced will display with a light green background. Logic cannot write to attributes while a point is time forced. Only present for Inputs and Logic Inputs, since outputs are always forced untimed.
Time to remain forced (sec)	When the <code>Timed</code> checkbox is checked, this sets the amount of time the attribute(s) will remained forced before returning to the contents before the force. Logic cannot write to attributes while a point is time forced. Only present for Inputs and Logic Inputs, since outputs are always forced untimed.
Value	When the checkbox to the left of Value is checked, the logic simulator will force the value attribute with the contents of the textbox to the right when the <b>Force Value</b> button is pressed.
Online	When the checkbox to the left of Online is checked, the logic simulator will force the online status attribute with the contents of the checkbox to the right when the <b>Force Value</b> button is pressed.
Changes	When the checkbox to the left of Changes is checked, the logic simulator will force the changes attribute with the contents of the textbox to the right when the <b>Force Value</b> button is pressed.
String	When the checkbox to the left of String is checked, the logic simulator will force the string attribute with the contents of the textbox to the right when the <b>Force Value</b> button is pressed.
Date/Time (UTC)	When the checkbox to the left of $Date/Time$ (UTC) is checked, the logic simulator will force the Date/Time attribute with the contents of the boxes to the right when the <b>Force Value</b> button is pressed.

Table 19: Logic Simulator – Force Point Value



### VI. Logic Examples

In this chapter, examples of several Advanced Math & Logic applications, including their setup, are provided. The main chapter topics are as follows.

- Single Comm Fail Point for Multiple Field Devices
- Automatic Switchover Between Values from Primary and Secondary Devices
- Simple Calculation
- Time-Delayed Output Operation

#### Single Comm Fail Point for Multiple Field Devices

When the Orion polls multiple field devices and the polled data is used by the Orion or a client station in a specific calculation, a communication error to any one of the devices will result in an invalid result for the overall calculation.

This logic monitors the Comm Fail points from two devices. As long as both devices are communicating, indicated by a 0 value, the combined Comm Fail flag will indicate good communications with a 0 value. If either of the devices (or both) are not communicating, indicated by a 1 value, the combined Comm Fail flag is set to a 1 value, indicating an error.

This example will explain the steps necessary to OR multiple Comm Fail points into a logic input using a timer. The logic will be executed every 1000ms and set the combined Comm Fail point to either 0 or 1.

#### Step 1: Create the Logic Timer Event (Figure 31)

- Under the Events menu, type CommFailEvent in the Create New Event textbox and click the Create button to create a new event.
- From the Event Type dropdown list, select Timer.
- Set the Timer Interval to 1000 milliseconds. This value determines how often the function (i.e. CommFailEvent Timer) will be executed.
- Set the Timer Enabled to True to enable the timer event.

The following window will be displayed (Figure 31). Notice the new CommFailEvent\_Timer () function has been added to the logic editor. The code for ORing the Comm Fail points will be placed into this function.

## Advanced Math & Logic for Orion

* NovaTech Configuration [	Director - [Logic Client Port 124 - Advanced Math and Logic (Example_OR.ncd)*]	
File Edit Configure	e Communications Window Help	_ 8 ;
🗋 💣 🗿   🏝 🛠 🖻   🧕	Ø Get NCD Update	
Port Options: Delete Port C	Close Port 🗋   🗖 🕼 😭 🖺 📮   🔎 🛛 🗡 🕅 🎇 Settings 🎇 Simulate   Line: 5 Character:	1
Commands     Erinder     Control     Define     Define     Define	<pre>function DefaultStartFunction() end function CommFailEvent_Timer() end</pre>	•
⊳ Math	<	>
b Urion b String	No code errors detected	
	Create New Event: Create	
	Event Name         Event Type         Timer Interval         Timer Enabled           DefaultStartFunction         Load         Image: Content of the state of	
	CommFailEvent Timer 1000 True	
	v.3.36.0013 2/12/20	21 3:30 PM

Figure 31: Create the CommFailEvent

### Step 2: Select Comm Fail Points of Field Devices (Figure 32)

The next step is to include both Comm Fail points of the respective field devices into the logic. Under the Inputs menu, (Figure 32), select the Comm Fail points for the respective field devices from the Point Name List and copy them to the Configured Inputs list.

Note that the Event field is left blank. Since a timer-based event is used to monitor the Comm Fail points, no data change event needs to be associated with the Comm Fail points.

# **A** NovaTech<sub>®</sub>

## Advanced Math & Logic for Orion

File Edit Configure	Communications Window	Help								- 1
🗁 🗊 🛤 🔆 🖻 🧕	) Get NCD Update lose Port 🗋   🗖 💽 🍸 🕥	3 <b>%</b> 3   🔎		× 🖙 s	ettings <sup>1</sup>	Simulate	Line: 5 C	haract	er: 1	_
⊕ Functions     ● Standard     Inputs     Outputs     Logic Unputs     Logic Unputs     Events         ∠ Commands         ▷ All         ▷ Control         ▷ Define         ▷ Evaluation         ▷ Loop	function DefaultStar end function CommFailEve end < No code errors detected Inputs	tFunction()	Inp	uts (2 Item(s) - Filter	By: All	Points)	Font			>
⊳ Math	🔎 <all ports=""></all>	× 🔀 📃	$\left \right $	Search	X	1	Configured In,	puts		
⊳ String	Point Name			Orion Point Name	Event	Event Type	Deadband	Min	Max	Que
	Amps C @Feeder 2	^ >	>	Comm Fail @Feeder 1			0.1	0	1	0
	Comm Fail @Feeder 1 Comm Fail @Feeder 2		× 🖄	Comm Fail @Feeder 2			0.1	0	1	0
	eth0 activity @Orion Internal eth0 link @Orion Internal	~	<		_					>

Figure 32: Select the Comm Fail Points

#### Step 3: Create the logic input point that will store the result.

Under the Logic Inputs menu, create the Logic Input point for storing the result of the logic (Figure 33). Under the Logic Inputs menu, click Add new point to create a new logic input point in the list. Type RelayCommFail for the Orion Point Name.

Note that the  ${\tt Event}~$  field is left blank for this point as well.

## Advanced Math & Logic for Orion

NovaTech Configuration I	irector - [Logic Client Port 124 - Advanced Math and Logic (Example_OR.ncd)*]	_	
🗋 File Edit Configure	Communications Window Help		- 8
Port Options: Delete Port C	ose Port 🗅   🗆 🗗 😰 🗇 🖫   🔎 🛛 🗡   📅 Settings 🅍 Simulate   Line	: 5 Character:	1
	<pre>function DefaultStartFunction() end function CommFailEvent_Timer() end</pre>		~
⊳ Math ⊳ Orion	No code errors detected		>
⊳ String	Inputs (1 Item(s) - Filter By: All Points)           Add new point          P         Scorch         Image: Configured Inputs	Font	12
	Orion Point Name         Event         Event Type         Deadband         Min         Max         Queue Size           RelayCommFail         .1         0         1         0		
	©	.36.0013 2/12/20	21 3:28 PM

Figure 33: Create Logic Input Point for Result

All input points required for the Comm Fail logic are now configured.

### Step 4: Writing the code to OR the Comm Fail points (Figure 34)

The logic obtains the value of the Comm Fail point for both devices (Comm Fail @Feeder 1 and Comm Fail @Feeder 2) using the orion.GetPoint function. If either point (or both) indicates a communications failure, the overall Comm Fail point (RelayCommFail @Logic) will be set to 1 using the orion.SetPoint to indicate a failure. If both points indicate good communications, RelayCommFail @Logic will be set to 0.

# Advanced Math & Logic for Orion

NovaT	ech Configurati	on Direct	or - [Log	ic Client Port 12	4 - Advanced	Math	and Log	c (Example_OR.ncd)] — I		$\times$
File	Edit Config	ure C	ommuni	cations Win	dow Help				-	<i>a</i> >
) 😂 🖉	💐 🛠 🖻	😧 Ge	t NCD Up	date						
ort Optio	ns: Delete Port	Close F	Port 🗋		21 표 월	🔎		🔀 🖙 Settings 🕍 Simulate 🔋 Line: 9 Character: 53		
) f	unction D nd	efaul	tStart	Function	()					^
f	unction C	ommFa	ilEver	t Timer (	)					
	If e	ither	Feede	r is in	Comm. Fai	п	set t	he logic point to 1, else set it to 0		
	if ori	on.Ge	tPoint	("Comm Fa	ail @Feed	ler :	L").v	alue == 1 or orion.GetPoint("Comm Fail @Feeder 2").value == 1 ther	1	
	or	ion.S	etPoir	t ({name=	'RelavCom	mFai	LI @L	ogic", value=1, online=true})		
	else							······································		
	==Both	derri		e online	cot the		nia T	pout to 0		
	Doon	ion S	et Poir	t ((name=	'RelayCom	mFa	1 AT.	agic" value=0 online=true})	- 1	
	end				1010303000			sgro / tarao o/ onrino traoj,		
	nd									
	na									×
• NG	code errors dete	cted								
Input	s (1 Item(s) -	Filter E	By: All P	oints)				Font		12
	and a state of the local state o					-				
Add I	ew point 📲 🤌	Searci	'n				Configur	ed Inputs		_
	Orion Poin	t Name	Event	Event Type	Deadband	Min	Max	Queue Size		
14	RelayComm	ail			.1	0	1	0		
0										
0										
								v.3.36.0013 2/12/2021	4:19	5 PM

Figure 34: Setting up Comm Fail Logic

Test and debug using the Logic Simulator integrated into the logic editor.

Note that in order to save logic to the .lua file,  ${\tt Save}$  or  ${\tt Save}$  as from the NCD main menu must be used.

#### Step 5: Map RelayCommFail @Logic to a server port

RelayCommFail @Logic (which now indicates the overall communications status) can be mapped to any server port on the Orion as needed. Once mapped, any client station connected to any of those server ports can obtain RelayCommFail @Logic, and thus a summary of the communication status of the field devices.

#### Automatic Switchover Between Values from Primary and Secondary Devices

When redundant devices are used for critical feeders, logic can be used to select which device's data is provided to SCADA. Under normal circumstances, the Orion uses the primary device's data. If the primary device goes offline, the Orion needs to use the secondary device's data. If both devices are offline, the data is set to zero and the points are marked offline. The logic running in the Orion will make these determinations as follows.

- 1) The logic checks the Comm Fail status for both the primary device and the secondary device.
- 2) If the primary device's Comm Fail point indicates the device is online, the logic copies the primary device's data to logical inputs. It also sets these logic inputs as being online so they are indicated as valid to the client station.
- 3) If the primary device's Comm Fail point indicates offline, and the secondary device's Comm Fail indicates online, then logic copies the secondary device's data to the same logical inputs. It also sets these logic inputs as being online so they are indicated as valid to the client station.
- 4) If both device's Comm Fail points indicate offline, then the logic sets all logical inputs to zero, and sets their communication status to offline. From the Comm Fail points, the client station can also see that both the primary device and the secondary device are offline.

#### Step 1: Create the Logic Timer Event (Figure 35)

- Under the Events menu, type an event name in the Create New Event textbox and click the Create button to create a new event.
- From the Event Type dropdown list, select Timer.
- Set the Timer Interval to 1000 milliseconds. This value determines how often the function (i.e. CheckFeeder1Switchover Timer) will be executed.
- Set the Timer Enabled to True to enable the timer event.

The following window will now be displayed (Figure 35). Notice the new CheckFeeder1Switchover\_Timer function has been added to the logic editor. The logic code needed for the automatic switchover will be placed into this function.



## Advanced Math & Logic for Orion

<b>D</b> 1	Logic Client Port 124 - Advance	ed Ma	ath and Logic (Example_P	rimary_Secor	ndary.ncd)*			8			
Port	Options: Delete Port Close	Port	00000	'I "I   🌶		×	Settings	Ŧ			
0	● Standard ● Functions Inputs Outputs Logic Inputs Logic Outputs Events  ▲ Commands  ▷ All ▷ Control ▷ Define ▷ Evaluation ▷ Loop	fu end fu end	<pre>function DefaultStartFunction() end function CheckFeeder1Switchover_Timer() end </pre>								
	⊳ Math ⊳ Orion	Cre	ate New Event:		Creat	е					
	b String		Event Name	Event Type	Timer Interval	Timer Enabled					
		$\times$	DefaultStartFunction	Load							
		$\times$	CheckFeeder1Switchover	Timer	1000	True					
		$\odot$									

Figure 35: Create Event for Automatic Switchover

# Step 2: Select the redundant input points from both the primary and secondary devices as logic inputs.

Under the Inputs menu, (Figure 36), select the redundant input points as well as the Comm Fail points from both the primary and secondary devices in the Point Name list, and copy them to the Configured Inputs list by clicking on >>. The Min and Max will default to the range defined at the client port which polls the devices.

Note that for all points, the Event field is left blank. Since a timer-based event is being used to monitor the Comm Fail points, association to a data change event to the Comm Fail points is not required.

## Advanced Math & Logic for Orion

Logic Client Port 124 - Adva	inced Math and Logic (Example_Primary_Secondary									_		
t Options: Delete Port Cl	ose Port 🗋 📔 🗊 😰 🖄 🖫 🔑		🔀 🛛 😤 Settings 🎽	Simulate	Line: 5	Character: 1						
Standard      Functions	function DefaultStartFunction	0									0	
Inputs	end										<u> </u>	
Outputs												
Logic Inputs	function CheckFeeder1Switchov	er_Tin	ner()									
Logic Outputs	end											
Events												
▲ Commands												
⊳ All												
Control												
Define												
	•											
p Evaluation	• ·· · · · · · ·											
p Evaluation p Loop p Math	<ul> <li>No code errors detected</li> </ul>											
⊳ Evenuation	No code errors detected  Inputs		Inputs (18 Item(s) - Filter By: Al	Il Points)					Fon	at 🕖	12	
p Evenuation ⊳ Loop ⊳ Math ⊳ Orion ⊳ String	No code errors detected Inputs		Inputs (18 Item(s) - Filter By: Al	ll Points)	1				Fon	at	72	
<ul> <li>▷ Loop</li> <li>▷ Math</li> <li>▷ Orion</li> <li>▷ String</li> </ul>	No code errors detected Inputs     (All Ports>     · X		Inputs (18 Item(s) - Filter By: Al	ll Points)	Configure	d Inputs			Fon	at 🖓	n I	
<ul> <li>⊵varuation</li> <li>b Loop</li> <li>Math</li> <li>Orion</li> <li>b String</li> </ul>	No code errors detected Inputs     CAll Ports>     V     Coint Name	]	Inputs (18 Item(s) - Filter By: Al	II Points)	Configure	d Inputs Deadband	Min	Max	For	nt 🖓	ť,	
<ul> <li>⊨ Evaluation</li> <li>⊨ Loop</li> <li>Math</li> <li>⊨ Orion</li> <li>⊨ String</li> </ul>	No code errors detected Inputs     Al Pots>     V     Responses @Feeder1 Secondary		Inputs (18 Item(s) - Filter By: Al	Il Points)	Configure	d Inputs Deadband 3276.7	Min 0	<b>Max</b> 32767	Fon Queue Size 0	at V	5	
p evanuation Loop Math Orion String	No code errors detected Inputs     call Pota>     v      inf Name Responses @Feeder1 Secondary     comm Fail @Feeder1 Secondary	) ) ) )	Inputs (18 Item(s) - Filter By: Al Search Orion Point Name Amps A @Feeder1 Primary Amps A @Feeder1 Primary	Il Points)	Configure	d Inputs Deadband 3276.7 3276.7	Min 0	Max 32767 32767	For Queue Size 0	nt V	ž.	
p Evaluation Loop Math Orion String	No code errors detected  Inputs  C4I Pots>  Point Name Responses @Feeder1 Secondary Comm Fail @Feeder1 Secondary Comm Fail @Feeder1 Secondary	~	Inputs (18 Item(s) - Filter By: Al Scorch Orion Point Name Amps A @Feeder1 Primary Amps A @Feeder1 Secondary Amps A @Feeder1 Secondary	Il Points)	Configured	d Inputs Deadband 3276.7 3276.7 3276.7	Min 0 0	Max 32767 32767	For Queue Size 0 0	at	2. 	
<ul> <li>⊵ varuation</li> <li>Loop</li> <li>Math</li> <li>Orion</li> <li>String</li> </ul>	No code errors detected  Inputs      Oal Ports.      Oal Ports.      Com Fail @Responses @Feder1 Secondary      Crom Fail @Reserved1 Secondary      Crom Fail @Reserved1 Secondary      Crom Fail @Reserved1 Secondary      Crom MaxIffr @Croin Internal	) > ~	Inputs (18 Item(s) - Filter By: Al Scarch Crion Point Name Amps A @Feeder1 Primay Amps A @Feeder1 Primay Amps B @Feeder1 Primay	Il Points)	Configure	d Inputs Deadband 3276.7 3276.7 3276.7	Min 0 0	Max 32767 32767 32767	For Queue Size 0 0 0	at 🖓	~	
p Evanuation Loop Math Orion String	No code errors detected      Inputs      Oli Ports>     Viet Name Responses @ Feeder1 Secondary Comm Fail @ Feeder1 Secondary Comm Fail @ Fooder1 Secondary Onion Estim @ Colon Internal Onion MacFire @ Onion Internal Onion PLL Freq Colon Internal	×	Inputs (18 Item(s) - Filter By; Al Scorch Orion Point Name Ampa A @Feeder1 Primary Ampa B @Feeder1 Primary Ampa B @Feeder1 Secondary	Il Points)	Configure	d Inputs Deadband 3276.7 3276.7 3276.7 3276.7	Min 0 0 0 0	Max 32767 32767 32767 32767	For Queue Size 0 0 0 0	at 🗸		
p Ecop p Loop p Math p Math p Otrion p String	No code errors detected  Inputs      dal Ports.      dal Ports.      dal Ports.      Com Fail @Responses @Feddr1 Secondary      Ornon EstErr @Orion Internal      Orion PLL: Freq @Orion Intern	) × «	Inputs (18 Item(s) - Filter By Al Storch Torion Point Name Amps & Greader1 Primary Amps & Greader1 Secondary Amps & Greader1 Secondary Amps & Greader1 Primary Amps & Greader1 Primary	Il Points)	Configure	d Inputs Deadband 3276.7 3276.7 3276.7 3276.7 3276.7 3276.7	Min 0 0 0 0 0 0	Max 32767 32767 32767 32767 32767	For Queue Size 0 0 0 0 0 0	at 🗸	~	
p = comunitation	No code errors detected      Inputs      Oli Ports      Oli Ports      Point Name Responses @Feder1 Secondary Comm Fail @Feder1 Secondary Comm Fail @Feder1 Secondary Orion Est:m @Orion Internal Orion PLL Freq @Orion Internal Orion PLL Offest @Orion Internal Orion PLL Freq @Orion Internal Orion PLL Offest @Orion Internal Orion PLL Freq @Orion Internal Orion PLL Fre	× ×	Inputs (18 Item(s) - Filter By Al Search Crion Point Name Amps & @Feeder1 Primary Amps & @Feeder1 Primary	Il Points)	Configure	d Inputs Deadband 3276.7 3276.7 3276.7 3276.7 3276.7 3276.7 3276.7	Min 0 0 0 0 0 0 0	Max 32767 32767 32767 32767 32767 32767	Fon 0 0 0 0 0 0 0 0 0 0 0 0 0	nt 🔾		
p Evonadion p Loop p Math p Orion p String	No code errors detected  Inputs      old Porto:      old:      old Porto:      old:      ol	× ×	Inputs (18 Item(s) - Filter By A)	Il Points)	Configure	d Inputs Deadband 3276.7 3276.7 3276.7 3276.7 3276.7 3276.7 0.1	Min 0 0 0 0 0 0 0 0 0 0	Max 32767 32767 32767 32767 32767 32767 1	Fon 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0			
p cranuation } Loop } Math } Orion ≥ String	No code errors detected      Inputs      Old Ports      Off Ports      Comm Fail @Feeder1 Secondary      Comm Fail @Feeder1 Secondary      Orion Est:rn @Orion Internal      Orion PLL Fore @Orion Internal      Orion PLL Offeet @Orion Internal      Orion PLL Offeet @Orion Internal      Orion Thus Crails @Orion Internal      Orion Time Quality @Orion Internal      Orion Time Quality @Orion Internal      Orion Time Quality @Orion Internal	× ×	Inputs (18 Item(s) - Filter By Al Search Orion Point Name Amps A @Feeder1 Primary Amps B @Feeder1 Primary Amps B @Feeder1 Primary Amps D @Feeder1 Primary Comm Fail @Feeder1 Primary Comm Fail @Feeder1 Primary	I Points)	Configure	d Inputs Deadband 3276.7 3276.7 3276.7 3276.7 3276.7 0.1 0.1	Min 0 0 0 0 0 0 0 0 0 0 0 0	Max 32767 32767 32767 32767 32767 32767 1	For Queue Size 0 0 0 0 0 0 0 0 0 0	zf 🔾		

Figure 36: Select Primary/Secondary Input Points

### Step 3: Create the logic input points to which the redundant input points will be copied.

Under the Logic Inputs menu, (Figure 37), create an appropriately named point for every pair of redundant input points. Depending on which one of the two redundant input points is online, the logic will copy one of the input points to the respective logic input point.

Under the Logic Inputs menu, click Add new point is to create a new logic input point in the list. Fill in the Orion Point Name. The Event field must be left blank (none of these points will trigger an event).

## Advanced Math & Logic for Orion

Options: Delete Port Clos	se Port 📋		-1	<i>~</i>		X	Se Se	ttings 📲 Simu	ulate Line:
<ul> <li>Standard</li> <li>Functions</li> </ul>	funct	tion DefaultSta	artFu	nction()					^
Inputs	end								
Outputs									
Logic Inputs	funct	tion CheckFeed	er1Sw:	itchover_	Timer()				
Logic Outputs	end								
Events									
Control									
Define									~
Evaluation	<								>
Loop	No c	code errors detected							
⊳ Loop ⊳ Math	No c	code errors detected (8 Item(s) - Filter B	v: All P	oints)				Gard	12
⊳ Loop ⊳ Math ⊳ Orion	No c     Inputs	code errors detected (8 Item(s) - Filter B	y: All P	oints)		_		Font	12
<ul> <li>Loop</li> <li>Math</li> <li>Orion</li> <li>String</li> </ul>	No c     Inputs     Add net	code errors detected (8 Item(s) - Filter B w point 🖶 🔎 Search	y: All P	oints)	× 🗊 🗹	Co	nfigured i	Font 🗸	12
<ul> <li>Loop</li> <li>Math</li> <li>Orion</li> <li>String</li> </ul>	No c     Inputs     Add ne	code errors detected (8 Item(s) - Filter B w point 📲 🔎 Search Orion Point Name	y: All P	oints) Event Type	X 📄 🗹 Deadband	Co Min	nfigured . Max	Font Inputs Queue Size	12
<ul> <li>Loop</li> <li>Math</li> <li>Orion</li> <li>String</li> </ul>	No of Inputs     Add net	code errors detected (8 Item(s) - Filter B w point P Search Orion Point Name Amps A	y: All P	oints) Event Type	Deadband	Co Min 0	nfigured A Max 32767	Font Font O	12
<ul> <li>Loop</li> <li>Math</li> <li>Orion</li> <li>String</li> </ul>	No c     Inputs     Add ne	code errors detected (8 Item(s) - Filter B w point Point Search Orion Point Name Amps A Amps B	y: All P	oints) Event Type	Deadband	<i>Col</i> <b>Min</b> 0	Max 32767 32767	Font Inputs Queue Size 0 0	12
⊳ Loop ⊳ Math ⊳ Orion ⊳ String	No c     Inputs     Add ne	code errors detected (8 Item(s) - Filter B w point Point Name Amps A Amps B Amps C	y: All P	oints) Event Type	Deadband .1 .1	<i>Co</i> Min 0 0	Max 32767 32767 32767	Font Inputs Queue Size 0 0 0	12
<ul> <li>Loop</li> <li>Math</li> <li>Orion</li> <li>String</li> </ul>	No c     Inputs     Add ne	code errors detected (8 Item(s) - Filter B w point Point Name Amps A Amps B Amps C VARs Total	Event	oints)	Deadband .1 .1 .1 .1	Co Min 0 0 0 -32768	Max 32767 32767 32767 32767 32767	Font Inputs Queue Size 0 0 0 0 0 0 0 0	12
<ul> <li>Loop</li> <li>Math</li> <li>Orion</li> <li>String</li> </ul>	No of Inputs     Add net	code errors detected (8 Item(s) - Filter B w point a point a construction Orion Point Name Amps A Amps B Amps C VARs Total Volts A	Event	oints) Event Type	Deadband .1 .1 .1 .1 .1	<i>Col</i> Min 0 0 0 -32768 0	Max 32767 32767 32767 32767 32767 32767	Font Font O	12
<ul> <li>Loop</li> <li>Math</li> <li>Orion</li> <li>String</li> </ul>	No c     Inputs     Add ne	code errors detected (8 Item(s) - Filter B w point in a scarch Orion Point Name Amps A Amps B Amps C VARs Total Volts A	Event	oints)	Deadband .1 .1 .1 .1 .1 .1	Col Min 0 0 -32768 0 0	nfigured / Max 32767 32767 32767 32767 32767 32767	Font Inputs Queue Size 0 0 0 0 0 0 0 0 0	12
<ul> <li>b Loop</li> <li>Math</li> <li>Orion</li> <li>String</li> </ul>	No c     Inputs     Add ne	code errors detected (8 Item(s) - Filter B we point i ) Search Orion Point Name Amps A Amps B Amps C VARs Total Volts A Volts B Volts C	Event	oints)	Deadband .1 .1 .1 .1 .1 .1 .1 .1 .1	Ca Min 0 0 0 -32768 0 0 0	nfigured 3 32767 32767 32767 32767 32767 32767 32767 32767	Font Inputs Queue Size 0 0 0 0 0 0 0 0 0	12
<ul> <li>b Loop</li> <li>Math</li> <li>Orion</li> <li>String</li> </ul>	Add ne	(8 Item(s) - Filter B w point Crion Point Name Amps A Amps B Amps C VARs Total Volts A Volts C Volts C	Event	oints)	Deadband .1 .1 .1 .1 .1 .1 .1 .1 .1	Con Min 0 0 -32768 0 0 0 0 -32768	Max 32767 32767 32767 32767 32767 32767 32767 32767	Font inputs Queue Size 0 0 0 0 0 0 0 0 0	12

Figure 37: Create Logic Input Point List

Now that all points required for the logic have been created, the next step is to write the actual logic.

### Step 4: Write logic to determine the live device and set the logic input points.

The logic consists of the following steps (Figure 38).

- The logic must first obtain the Comm Fail points for both devices.
- If the primary device is online, the logic copies the device's input points to the logic input points, and sets the logic input points to online.
- If the primary device is offline and the secondary device is online, the logic copies the input points from the secondary device to the logic input points, and sets the logic input points to online.
- If both devices are offline, the logic copies 0 to the logic input points and sets the logic input points to offline.

## Advanced Math & Logic for Orion



Figure 38: Automatic Switchover Logic

Test and debug using the Logic Simulator integrated into the logic editor.

#### Step 5: Map the points to server ports.

The Logic Inputs can be mapped to any server port on the Orion as needed. Once mapped, any client station connected to any of those server ports can obtain the data coming from the primary or secondary devices based on communication status.

Note that in order to save logic to the .lua file,  ${\tt Save}$  or  ${\tt Save}$  as from the NCD main menu must be used.



### Simple Calculation

When using older IEDs, it may be necessary to compute the 3 Phase Current from the A phase, B phase, and C phase values read from a device. The following example computes 3 Phase Current every time one or more of the phase values change.

### Step 1: Create the DataChange event (Figure 39)

- Under the Events menu, type Feeder1\_Amps3P in the Create New Event textbox and click the Create button to create a new event.
- From the Event Type dropdown list, select DataChange.

The following window will be displayed (Figure 39). Notice the new Feeder1\_Amps3P() function has been added to the logic editor. The logic code for the phase current calculation will be placed into this function.



Figure 39: Create Event for Calculation

#### Step 2: Select the A phase, B phase, and C phase points as inputs.

Under the Inputs menu, (Figure 40), select the phase current input points in the Point Name list, and copy them to the Configured Inputs list by clicking >> . Note that since the function Feeder1\_Amps3P() should be executed every time one or more of the three values change, each input point must be set up with Feeder1\_Amps3P as Event. Select the Deadband event type and change the deadband to 1 for all three points so the event will trigger when any of the three points change more than a value of 1.



## Advanced Math & Logic for Orion

Logic Client Port 124 - Advance	ed Math and Logic (Example_Ca	lculation.ncd)*									- 8
Port Options: Delete Port Close	Port 🗋 📔 🗖 🖿 😭	3 ×3   🔑 📃			🗡 🛛 🚰 Setti	ings   🎽 Simulate	Line: 1 C	haracter: 1			
◆ Standard     ● Functions       Inputs     Outputs       Logic Inputs     Logic Outputs       Events     ∠ Commands       ⇒ All     ▷ Control	<pre>function DefaultStartFunction() end function Feeder1_Amps3P() end &lt;</pre>										~
<ul> <li>Define</li> <li>Evaluation</li> </ul>	Inputs			Inp	uts (3 Item(s) - Filte	er By: All Points)			F	ont 🕖	12
⊳ Loop ⊳ Math	🔎 <all ports=""></all>	~ 🗙 🛛		$\mathbf{P}$	Search	🗙 📑 🗹	Configure	ed Inputs			
D Orion	Point Name				Orion Point Name	Event	Event Type	Deadband	Min	Max	Queue Size
⊳ String	Amps A @Feeder 1		^ >>		Amps A @Feeder 1	Feeder1_Amps3P	DeadBand	1	0	32767	0
	Amps B @Feeder 1				Amps B @Feeder 1	Feeder1_Amps3P	DeadBand	1	0	32767	0
	Amps C @Feeder 1 Volts A @Feeder 1 Volts B @Feeder 1				Amps C @Feeder 1	Feeder1_Amps3P	DeadBand	1	0	32767	0

Figure 40: Select Phase Current Input Points

### Step 3: Create the point to which the result of the phase current calculation will be copied.

Under the Logic Inputs menu (Figure 41), create a point for storing the result of the phase calculation

by clicking on Add new point is to create a new row in the list. Fill in the Orion Point Name. Then configure the Min and Max. Since this point is an analog point derived from analog inputs from the field, it should have the same Min and Max as the field inputs. The Event field must be left blank (computing the phase current does not trigger another event).

	Logic Client Port 124 - Advan	iced Math	and Logic (Example_Ca	lculation	n.ncd)*						23
Port	t Options: Delete Port Clos	se Port 🗋		3 5	$\left  \right\rangle$		7	K   🙍	P Settings  🎽 S	imulate	Ŧ
3	Standard      Functions      Inputs     Outputs     Logic Outputs     Events     Commands     b.All     b.Control     b.Define     Evaluation	funct end funct end	tion DefaultSt.	artFu: nps3P	nction()					~	
	⊳ Loop	<								>	
	⊳ Math	No c	ode errors detected								
	String	Inputs	(1 Item(s) - Filter B	y: All P	oints)			F	ont 🗸	12	
		Add net	w point 🐈 🔎 Search	1	2	K 🗈 🗹		Configure	ed Inputs		_
		4	Orion Point Name	Event	Event Type	Deadband	Min	Max	Queue Size		
		1	Feeder1_Amps3P			.1	0	32767	0		
		$\odot$									1

Figure 41: Create Logic Point List

Now that all points required for the logic have been set up or created, the next step is to write the actual logic.

# **NovaTech**

Step 4: Write logic for calculating the phase current (Figure 42)

This logic will be executed every time one of the three phase inputs from the field changes.



Figure 42: Phase Calculation Logic

#### Step 5: Map Feeder1 Amps3P @Logic to any server port on the Orion as needed.

Once server ports are mapped, any client station connected to any of those server ports can obtain the three phase values from the device, as well as the result of the calculation.



### **Time-Delayed Output Operation**

This logic monitors a discrete input status to determine whether a level has been reached. When the field input status changes to 1, the level has been reached and the logic starts a timer. After 5 seconds, the logic sends the output command 1 to the device. When the input status changes to 0, the logic immediately sends the output command 0 to the field device without delaying.

#### Step 1: Create the logic events (Figure 43)

#### IsReached DataChange event:

- Under the Events menu, type IsReached in the Create New Event textbox and click the Create button to create a new event.
- From the Event Type dropdown list, select DataChange. The event type is is the only parameter that needs to be set up.

#### Shutoff Timer event:

- Under the Events menu, type Shutoff in the Create New Event textbox and click the Create button to create a new event.
- From the Event Type dropdown list, select Timer.
- Set the Timer Interval to 5000 milliseconds. This value determines how often the function (i.e. Shutoff Timer) will be executed.
- Set the Timer Enabled to False to disable the timer event as shown below (Figure 43). This timer will only run when enabled in logic.

🕒 Logic Client Port 124 - Advanced N	th and Logic (Example_TimeDelayedOut	tput.ncd)*							
Port Options: Delete Port Close Port	🗅   🗆 🗗 🖿 🖄 🖫   🔑	🔀 🛛 🏹 🔀 🗲							
<ul> <li>◆ Standard ● Functions</li> <li>Inputs         <ul> <li>Outputs</li> <li>Logic Inputs</li> <li>Logic Outputs</li> <li>Events</li> <li>✓ Commands</li> <li>▷ All</li> <li>▷ Control</li> <li>▷ Define</li> </ul> </li> </ul>	<pre>local previous = 0 function IsReached() end function Shutoff_Timer() end</pre>								
▷ Evaluation ▷ Loop ▷ Math	No code errors detected Create New Event:	Create							
DOTION ► String	Event Name Event Type T	imer Interval Timer Enabled							
<i>p</i> oung	X DefaultStartFunction Load								
	× IsReached DataChange								
	X Shutoff Timer 50	000 False							
	$\odot$								

Figure 43: Create Events for Shutoff Logic

#### The following window will be displayed.

#### Step 2: Select LevelReached @I/O Device from Inputs.

Under the Inputs menu, (Figure 44), select LevelReached @I/O Device in the Point Name list, and copy it to the Configured Outputs list by clicking >>. Note that since the function IsReached should be executed every time LevelReached @I/O Device changes, LevelReached @I/O Device must be set up as the event for IsReached. Select the Deadband event type and change the deadband to 0.1 so the event will trigger when the point changes more than a value of 0.1.

Coptions. Delete Fort	Close Port 🗋 📘 🗇 🖿 🖼 📲 🔑			🗡 🛛 🚰 Setting	gs 🎽 Simu	late Line:	4 Character:	21	
Functions     Standard	function DefaultStartFunction() end								
Inputs Outputs Logic Inputs Logic Outputs Events Commands All Control Define	<pre>function IsReached() end function Shutoff_Timer() end </pre>								>
Evaluation	<ul> <li>No code errors detected</li> </ul>								
p Math	Inputs		inpu	ts (Titem(s) - Filter By	All Points	)	Font		- 14
D Orion	🔎 <al ports=""> 🗸 📉</al>		PS	earch	X 🗊 🗷	Configu	ured Inputs		
5 String	Point Name			Orion Point Name	Event	Event Type	Deadband	Min	Max
	Polls @I/O Device	>>	1	LevelReached @I/O Device	IsReached	DeadBand	0.1	0	1
		and the second se							

Figure 44: Select Device Input Point

Step 3: Select Shutoff @I/O Device from Outputs.

Under the Outputs menu, (Figure 45), select Shutoff @I/O Device in the Point Name list, and copy it to the Configured Outputs list by clicking >>. The Event field must be left blank.

	Logic Client Port 124 - Adva	anced Math and Logic (Example_TimeDelayedOutput.ncd)*	
Por	t Options: Delete Port Clo	lose Port 🗅   🗖 😰 📹 🗐 📱 🗐 🔎 📉 🗡 🕅 🔀 Settings 🎦 Simulate   Line: 12 Character: 1	
٢	Standard Functions     Inputs     Outputs     Logic Inputs     Logic Outputs	<pre>local previous = 0 function IsReached() end</pre>	^
	Events Commands All Control	<pre>function Shutoff_Timer() end</pre>	•
	⊳ Define	<	>
	Evaluation	No code errors detected	
	⊳ Math	Outputs Outputs (1 Item(s) - Filter By: All Points) Font	12
	String	All Ports> 🗸 💽 🖉 Search 🔀 🛐 🖉 Configured Outputs	
		Point Name Orion Point Name Event Event Type Deadband Min Max Que	eue Size
		Shutoff @I/O Device         0.1         0         1         0	

Figure 45: Select Device Output Point



Now that all points required for the logic have been set up or created, the next step is to write the actual logic.

### Step 4: Write logic for triggering the shutoff commands (Figure 46)

The routine IsReached is to be executed every time LevelReached @I/O Device is updated. Typically, the value is updated when the client port polls the point LevelReached @I/O Device from the device. If LevelReached @I/O Device changes to 1, Shutoff @Device will be set to 1 after a delay of 5 seconds. If LevelReached @I/O Device changes to 0, Shutoff @I/O Device will be set to 0 immediately.

The timer <code>Shutoff</code> is enabled only if the previous value of <code>LevelReached @I/O</code> <code>Device</code> is 0, and the current value of <code>LevelReached @I/O</code> <code>Device</code> is 1. If the previous value is 1, and the current value is 0, the shutoff command is cleared immediately.



Figure 46: Shutoff Logic

### Appendix A – List of Advanced Math & Logic Commands

### **Programming Elements**

The Orion Advanced Math & Logic supports the following programming elements.

- Variable declarations
- Conditional statements
- Repetitive statements (loops)
- Functions

Orion Advanced Math & Logic utilizes the Lua programming language (Lua 5.3.5 – Copyright © 2015–2020 Lua.org, PUC-Rio). For a complete reference of the Lua programming language, refer to the complete online reference manual via the following link: <u>http://www.lua.org/manual/5.3/manual.html</u>.

#### local Variable Declaration

Local variables are defined with local inside of a function. Any variable is assumed to be global unless explicitly declared inside a function as a local. Global variables do not need to be declared, but can be declared using local outside of a function. Before the variable is used on the right side of an = assignment, it must be assigned a value. At the end of a function, local variables are not maintained to the next execution of the same function. Global variables store the value between two executions of the function and can be used by any function

Syntax:

local variable name

#### Example:

local i

```
--now i can be used
i = orion.GetPoint("Breaker Status @Feeder1")
```



#### if Conditional Statement

The if statement allows specific code to be executed only if a certain condition or sets of conditions are met. The conditions are specified with expressions (e.g. i > 5 or a == 1 and  $b \sim= 1$ ). Expressions are specified using Logic Operations to compare variables and values.

Syntax:

```
if expression then
   <list of statements>
[elseif expression then
   <list of statements>]
[else
   <list of statements>]
end
```

#### Example:

local i

```
--Returns the Value of "Point1 @Device1"

i = orion.GetPoint("Point1 @Device1")

if i.value > 50 then

--Do something here

elseif i.value > 25 then

--Do something here

else

--Do something here

end
```

#### for Loop

The for statement allows specific code to be repeatedly executed. The number of repetitions is specified ahead of time. A running counter is incremented for every repetition of the loop. The running counter can be incremented at every repetition by a specific increment specified in StepValue. If no increment value is specified, the running counter will be incremented by 1.

Syntax:

#### Example:

```
local i
--This will print 2 4 6 8 10 12 14 16 18 20 to the output window
for i = 1, 10 do
    orion.PrintDiag(i * 2)
end
```

#### while Loop

The while statement allows specific code to be repeatedly executed when the specified condition is met. The condition is checked before the first iteration of the loop and is repeated with every iteration until a specified condition no longer met. A while loop will not execute, if the condition is not met. The conditions are specified with expressions (e.g. i > 5 or a == 1 and  $b \sim= 1$ ). Expressions use Logic Operations to compare variables and values.

Syntax:

```
while expression do
    t of statements>
end
```

Example:

local i

```
--This will print 20 30 40 50

i = 10

while i < 50 do

i = i + 10

orion.PrintDiag(i)

end
```

#### repeat Loop

The repeat statement allows specific code to be repeatedly executed. The condition is checked after the first iteration of the loop and is repeated with every iteration until a specified condition is no longer met. A repeat loop will always execute at least once. The conditions are specified with expressions (e.g. i > 5 or a = 1 and  $b \sim 1$ ). Expressions use Logic Operations to compare variables and values.

Syntax:

```
repeat
  <list of statements>
until expression
```

#### Example:

local i

```
--This will print 20 30 40 50

i = 10

repeat

i = i + 10

orion.PrintDiag(i)

until i >= 50
```



#### function Function

If the same code, such as a math formula, is required in multiple locations in the logic, it is easier to encapsulate that code once as a function, and then call the function every time that formula needs to be computed. Thus, the function statement reduces overall code length and complexity. A function can be tied to an event and executed when that event occurs (see <u>Events Menu</u> section for setup details). In this case the function does not return results to the calling location.

#### Syntax:

```
function FunctionName (parameter {, parameter})
  <list of statements>
  return result
end
```

#### Parameters:

Zero or more parameters can be specified.

#### Return Value:

The return statement is used as an option to assign the result(s) of the function. The resulting value(s) are returned to the calling location.

#### Examples:

```
function GetStatusName(x)
local s
if ( x == 1 ) then
  s = "Meter1Status @Logic"
elseif ( x == 2 ) then
  s = "Meter2Status @Logic"
elseif ( x == 3 ) then
  s = "Meter3Status @Logic"
elseif ( x == 4 ) then
  s = "Meter4Status @Logic"
end
  return s
end
function Circumference(radius)
  return 2 * radius * 3.14159265
end
```



```
function MySub()
local sVar
local i

--This will print the values for database points
--"Meter1Status @Logic", "Meter2Status @Logic",
--"Meter4Status @Logic", and '"Meter4Status @Logic"
for i = 1, 4 do
   sVar = GetStatusName(i)
   i = orion.GetPoint(sVar)
   orion.PrintDiag(i)
end

--This will print "12.5663706"
orion.PrintDiag( Circumference (2) )
```



Advanced Math & Logic for Orion

### **Database Operations**

orion.GetPoint	Returns a specific point attribute or table of point attributes of the specified real-time database point. These attributes include:							
	name	String with name of the point. Example: "Breaker Status @Feeder1"						
	index	Number of index in the real-time database.						
	value	Value of the point.						
	changes	The number of value changes of the point.						
	online	The online status of the point. false indicates the point is Offline						
	string	true indicates the point is Online The optional string attribute of the point nil if the						
	SCIIIIg	string attribute hasn't been written to						
	time	The timestamp of the last write of the point to the real-time database in ISO format in UTC time. The time is undated with every write regardless of value						
		changes. Example time: "2019-09-17 22:06:30.807Z"						
Syntax:	orion.GetPoint	(name)						
Parameters:	name	Name of the point whose table of point attributes is to be obtained. Example: Breaker Status @Feeder1.						
Return Value:	The specific attribuch and the specific attribuch of the specified point of the specint of the specified point of	ute or a table with current numeric value, number of fline status, time and associated string (if applicable), int.						
Example:	<pre>local i = {}</pre>							
	Returns a ta "Breaker Sta i.value or i i = orion.GetE orion.PrintDia	able of current attributes of atus @Feeder1" [["value"] contains the point's value Point("Breaker Status @Feeder1") ag("Breaker Status is " i.value)						
	local i							
	Returns only i = orion.GetE orion.PrintDia	v "Breaker Status @Feeder1" value Point("Breaker Status @Feeder1").value ug("Breaker Status is " i)						

orion.SetPoint	Sets the value, duration, offt database point. The index. orion.Seconotification to inform	changes, online status, associated string, time, and/or pulses of the specified real-time the point can be specified either by name or by etPoint will also automatically generate a change mother interested modules of the point's change.
Syntax:	orion.SetPoint	(PointAttributes)
Parameters:	PointAttribute	s table entries:
	name	String with name of the point whose attributes are to be set.
	index	Number with index of the point whose attributes are to be set. This index can be retrieved from a prior
	value	New value that the point will be set to. This value should be within the set scaling range for the point. However, setting the value below or beyond the set scaling is not restricted.
	changes	This parameter is optional and sets the number of changes for the point.
	online	This parameter is optional and sets the point's online status if desired. false sets the point to Offline true sets the point to Online
	time	This parameter is optional and sets the timestamp of the point in ISO format in UTC time. Example time: "2019-09-17 22:06:30.807Z"
	duration	This parameter is optional and sets the point's control duration in milliseconds. This parameter will be used in conjunction with a Trip/Close point operation
	offtime	This parameter is optional and sets the point's time between pulses in milliseconds. This parameter will be used in conjunction with a Trip/Close point operation.
	pulses	This parameter is optional and sets the point's number of pulses. This parameter will be used in conjunction with a pulse train point operation.
	string	This parameter is optional and sets the point's string attribute.
Return Value:	None.	



Example:	Set the value to Online orion.SetPoint value=500, onl: Set the value its online s orion.SetPoint value=500})	e of the point ({name="kWh ine=true}) e of the point tatus ({name="kWh	to 500 Phase to 500 Phase	and it: A without A	s status @Feeder1", t changing @Feeder1",
orion.AssocPoint	Associates the spe The point must be specifies the function An optional object function. This function	cified point with the specified by nar on to be called wh argument specifie on can only be call	e specified ne, and t en a data es the tab ed in the l	l data cha the functi change le which ∟oad func	inge function. on argument event occurs. contains the tion.
Syntax:	orion.AssocPoin DataChangeType	nt(PointName, 3 , Deadband])	Function	n [, Ob <u></u>	ject,
Parameters:	PointName Function Object DataChangeType Deadband	String with name data change functi Example: "Breake Function to be ca occurs on the asso Table containing normal, global fun missing or nil. String indicating to refresh or dead this optional argun Deadband value a	of the po on is to be ar Statu lled when ociated po the functi ction if thi band. Th nent is mis associated	oint whose set. a GFeeco int. on. The s optiona change to e default ssing or n d with De	der1" change event default is a l argument is ype of either is refresh if il. adband data
		change type. Thi when DataCha Deadband. The 0.000000001 if th	s parame ngeType e default is optiona	eter is onl is s deadba al argume	y meaningful pecified as nd value is nt is missing.
Return Value:	None.				
Example:	Associate the feeder1 orion.AssocPoin feeder1.kwh_a,	e point with th nt("kWh Phase i feeder1, "dead	ne kwh_a A @Feede dband",	method er1", 1)	l in object

orion.GetQueuedPoint	Returns a table of point attributes of the specified point's change event. These attributes include:	
	name index value changes online string	String with name of the point. Example: "Breaker Status @Feeder1" Number of index in the real-time database. Event value of the point. Event number of value changes of the point. Event online status of the point. false indicates the point is Offline true indicates the point is Online Event string attribute of the point. nil if string attribute was not written to.
	time	Event timestamp in ISO format in UTC time. Example time: "2019-09-17 22:06:30.807Z"
Syntax:	orion.GetQueuedPoint(name)	
Parameters:	name Name of the point whose table of event attributes is to be obtained. Example: "Breaker Status @Feeder1".	
Return Value:	A table of queued event attributes including numeric value, number of changes, online/offline status, time and string. If no events are queued for this point, nil will be returned.	
Example:	<pre>local pt1 = {}</pre>	
	Get the fir pt1 = orion.Ge	<pre>st queued point change from the queue tQueuedPoint("Point1 @Device1")</pre>
	If nil, no while pt1 ~= n	events il do
	orion.Print <i>Get the</i> orion.GetQu	Diag("Value: " pt1.value) next queued event euedPoint("Point1 @Device1")
	end	

# Advanced Math & Logic for Orion

### Logging Operations

orion.PrintDiag	Prints the specified string to the View Communications – Logic Window of the Orion MMI, or the Diagnostic Output window of the Logic Simulator. Carriage return/line feed is automatically appended.	
Syntax:	orion.PrintDiag	g(String)
Parameters:	String String to be printed to communication window of the Orion, or the Diagnostic Output window of the Logic Simulator. It is always printed on a new line.	
Return Value:	None.	
Example:	<i>Prints "The value is 32"</i> orion.PrintDiag("The value is 32")	
orion.PrintLog	Prints the specified string to the System Menu – Event Log window of the Orion MMI, and the Log Output window of the Logic Simulator. Carriage return/line feed is automatically appended.	
	Note: Excessive the point that the information.	use can fill the Orion Event Log with entries to log becomes too large to extract meaningful
Syntax:	orion.PrintLog(String)	
Parameters:	String	String to be printed to the event log of the Orion, or the Log Output window the Logic Simulator. It is always printed on a new line.
Return Value:	None.	
Example:	Prints "The value is 32" to the event log orion.PrintLog("The value is 32")	

# Advanced Math & Logic for Orion

## Logic Operations

== > >= < < ~	Compare the values of two numbers, variables, or expressions.
Syntax:	operand operator operand where
	<ul> <li>the operand can be a number, a variable, or an expression</li> <li>the operator is one of the following: ==, &gt;, &gt;=, &lt;, &lt;=, ~=</li> </ul>
Parameters:	All comparison operators are binary operators, i.e. they require a constant, variable, or expression on the left side and on the right side.
Return Value:	Result of the comparison operation.
Example:	<pre>local i = {}</pre>
	<pre>i = orion.GetPoint("Tap Position @Transformer5")</pre>
	<pre>if i.value &gt; 5 then     orion.PrintDiag("i is greater than 5") elseif i.value == 5 then     orion.PrintDiag("i is equal to 5") else     orion.PrintDiag("i must be less than 5") end</pre>

# **NovaTech**

## Advanced Math & Logic for Orion

and	Performs the logical and operation on two variables or expressions.		
Syntax	operand1 <b>and</b> operand2		
	where the operands can be a variable or an expression.		
Parameters	and is a binary operator, i.e. it requires a variable or expression on the left side and on the right side.		
Return Value	: Result of the and operation as follows.		
	Operand 1 Operand 2 Result		

oporana i	oporana	Roodit
0 – False	0 – False	0 – False
0 – False	1 – True	0 – False
1 – True	0 – False	0 – False
1 – True	1 – True	1 – True

Example:

```
local br1 = { }
local br2 = { }
```

```
br1 = orion.GetPoint("Breaker 1 @Subl")
br2 = orion.GetPoint("Breaker 2 @Subl")
```

```
if br1.value and br2.value then
    orion.PrintDiag("Both breakers closed")
else
    orion.PrintDiag("At least one breaker not closed")
end
```

or

## Advanced Math & Logic for Orion

Performs the logica	or operation on two	variables or expressions.
---------------------	---------------------	---------------------------

Syntax: operand1 or operand2

where the operands can be a variable or an expression.

Parameters: or is a binary operator, i.e. it requires a variable or expression on the left side and on the right side.

Return Value: Result of the or operation as follows.

Operand 1	Operand 2	Result
0 – False	0 – False	0 – False
0 – False	1 – True	1 – True
1 – True	0 – False	1 – True
1 – True	1 – True	1 – True

Example: local br1

```
local br2
```

```
brl = orion.GetPoint("Breaker 1 @Subl")
br2 = orion.GetPoint("Breaker 2 @Subl")
if br1 or br2 then
    orion.PrintDiag("At least one breaker closed")
else
```

orion.PrintDiag("No breaker closed")
end

# Advanced Math & Logic for Orion

### Math Operations

Perform a mathematical operation.
operand1 + operand2
operand1 - operand2
operand1 * operand2
operand1 / operand2
operand1 % operand2
operand1 ^ operand2
operandi operandz
where the operands can be a variable or an expression.
+, -, *, /, %, and ^ are binary operators, i.e. each requires a variable o expression on the left side and on the right side.
Result of the mathematical operation.
local i
<pre>i = 5 + 8i is now 13 (add) i = i - 3i is now 10 (subtract) i = 4 * ii is now 40 (multiply) i = 120 / ii is now 3 (divide) i = 5 % ii is now 2 (remainder of division) i = i ^ 8i is now 256 (2 to the power of 8)</pre>

math.abs		Returns the absolute value of the specified number.	
	Syntax:	math.abs(Number)	
	Parameters:	Number	Number or expression whose absolute value will be returned.
	Return Value:	Absolute value of	the given number.
	Example:	local i	
		i = math.abs(	-5)Returns 5
math.asin math.acos math.atan		Perform an invers	e trigonometric operation.
	Syntax:	math.asin(Num math.acos(Num math.atan(Num	ber) ber) ber)
	Parameters:	Number	Ratio from which angle in radians is computed.
	Return Value:	Angle in radians.	
	Example:	local i	
		i = math.asin	(0) <i>Returns 0</i>

orion.Fix		Returns the integer (fixed) portion of a number. If the number is negative, orion.Fix returns the negative number greater than or equal to the number.	
Syn	itax:	orion.Fix(Number)	
Paramet	ters:	Number or expression for which the fixed portion of its value will be returned.	
Return Va	lue:	Fixed portion of the number.	
Exam	ple:	local i	
		Positive Number i = orion.Fix(3 * 0.9)Result i = 2	
		Negative Num i = orion.Fix(	ber -2.2)Result i = -2
orion.Int		Returns the intege orion.Int return number.	r portion of a number. If the number is negative, s the negative number less than or equal to the
Syn	itax:	orion.Int(Numb	er)

Parameters:	Number	Number or expression for which the integer value of its value will be returned.	
Return Value:	Integer value of the	e number.	
Example:	local i		
	Positive Num i = orion.Int	nber (3 * 0.9)Result i = 2	
	Negative Num i = orion.Int	nber (-2.2)Result i = -3	
math.log math.log10		Compute the natural logarithmic value (math.log) or base-10 logarithmic value (math.log10) for supplied number.	
------------------------------	---------------	---	---
Syntax:		math.log(Number	c)
		<pre>math.log10(Number)</pre>	
	Parameters:	Number	Number for which to compute logarithmic value.
F	Return Value:	Natural or base-10 logarithmic value for supplied number.	
	Example:		
		i = math.log10	(100)Returns 2
math.pow		Returns the result of by the exponent part	of taking the base parameter to the power specified ameter.
	Syntax:	<pre>math.pow(Base, Exponent)</pre>	
	Parameters:	Base	Number to which the power will be applied.
		Exponent	Power to apply to the base
F	Return Value:	Result of given base taken to given power.	
	Example:	local i	
		i = math.pow(2,	5)Returns 32
<pre>math.sin math.cos</pre>		Perform a trigonom	etric operation on an angle.
math.tan			
	Syntax:	<pre>math.sin(Angle) math.sin(Angle)</pre>	
		math.tan(Angle)	
	Parameters:	Angle	Angle in radians
r			
ŀ	kelurn value:	Ralio (sine, cosine,	or langent).
	Example:	local i	
		i = math.sin(45	5)Returns 0.7071

math.sqrt Returns the square root of the specified number.		Returns the square root of the specified number.	
	Syntax:	<pre>math.sqrt(Number)</pre>	
	Parameters:	Number Number for which the square root is needed.	
	Return Value:	Square root of the specified number	
	Example:	local i	
		i = math.sqrt(49)Returns 7	
math.rad		Converts an angle from degrees to radians.	
	Syntax:	<pre>math.rad(Angle) Angle The angle in degrees that is to be converted. The value of the angle in radians. local i</pre>	
	Parameters:		
	Return Value:		
	Example:		
		i = math.rad(180)Returns pi	
math.deg		Converts an angle from radians to degrees.	
	Syntax:	<pre>math.deg(Angle)</pre>	
	Parameters:	Angle The angle in radians that is to be converted.	
	Return Value:	The value of the angle in degrees.	
	Example:	local i	
		<pre>i = math.deg(math.pi())Returns 180</pre>	

math.ceil	Returns the smallest integer larger than or equal to the specified number.		
Syntax:	<pre>math.ceil(Number)</pre>		
Parameters:	Number Number for which the ceiling is needed.		
Return Value:	Ceiling of the specified number		
Example:	local i		
	i = math.ceil(3.142)Returns 4		
math.floor	Returns the largest integer smaller than or equal to the specified number.		
Syntax:	<pre>math.floor(Angle)</pre>		
Parameters:	Angle Number for which the floor is needed.		
Return Value:	Floor of the specified number.		
Example:	<pre>local i i = math.floor(3.142)Returns 3</pre>		
math.min	Returns the minimum value among its arguments.		
Syntax:	<pre>math.min(x,)</pre>		
Parameters:	x, The numbers for which the minimum is needed.		
Return Value:	The minimum value of the given numbers.		
Example:	local i		
	i = math.min(3, 4)Returns 3		

### **NovaTech**

### Advanced Math & Logic for Orion

math.max		Returns the maximum value among its arguments.	
	Syntax:	<pre>math.max(x,)</pre>	
	Parameters:	<ul><li>x, The numbers for which the maximum is needed.</li><li>The maximum value of the given numbers.</li></ul>	
	Return Value:		
	Example:	local i	
		i = math.max(3, 4)Returns 4	

math.exp

Returns the value of e to the specified power.

Syntax:	<pre>math.exp(x)</pre>		
Parameters:	x The power to apply to <i>e</i> .		
Return Value:	The value of <i>e</i> to the specified power.		
Example:	local i		
	i = math.exp(1)Returns e		

math.pi Returns the value of pi.
Syntax: math.pi()
Parameters: None.
Return Value: The value of pi.
Example: local i
i = math.pi() --Returns pi

bit.band		Performs a bitwise conjunction of two or more expressions.	
	Syntax:	<pre>bit.band(x1 [,x2])</pre>	
	Parameters:	x1 $[, x2]$ Numbers for which a bitwise conjunction is needed.	
	Return Value:	Bitwise conjunction of the specified numbers.	
	Example:	local i	
		i = bit.band(1, 3)Returns 1	
bit.bor		Performs a bitwise disjunction of two or more expressions.	
	Syntax:	bit.bor(x1 [,x2])	
	Parameters:	x1 [, x2] Numbers for which a bitwise disjunction is needed.	
	Return Value:	Bitwise disjunction of the specified numbers.	
	Example:	local i	
		i = bit.bor(1, 3)Returns 3	
bit.bxor		Performs a bitwise exclusion of two or more expressions.	
	Syntax:	bit.bxor(x1 [,x2])	
	Parameters:	x1 [, x2] Numbers for which a bitwise exclusion is needed.	
	Return Value:	Bitwise exclusion of the specified numbers.	
	Example:	local i	
		i = bit.bxor(1, 3)Returns 2	

### Advanced Math & Logic for Orion

### String Operations

string.byte	Returns the character code of the characters $s[i]$ , $s[i+1]$ ,, $s[j]$ . The default value of i is 1; the default value of j is i.
Syntax:	<pre>string.byte(s [, i [, j]])</pre>
Parameters:	s [, i [, j]] The string, starting position, and ending position.
Return Value:	The character code for the characters s[i], s[i+1], , s[j].
Example:	local i
	<pre>i = string.byte("A")Returns 65</pre>

string.char	Returns a string containing the character associated with the specified character code.	
Syntax:	<pre>string.char(charcode)</pre>	
Parameters:	charcode	Number for which the corresponding character is to be obtained.
Return Value:	ASCII character for supplied number.	
Example:	local sVar sVar = string.char(66) <i>Returns "B"</i>	

string.find	Returns the start and end position of the first occurrence of a search string within a base string. A value of true as a fourth, optional argument Plain turns off the pattern matching facilities.	
Syntax:	<pre>string.find(BaseStr, SrchStr [, StartPos [, Plain]])</pre>	
Parameters:	BaseStr	The string which will be searched for SrchStr.
	SrchStr	The string that will be searched for in ${\tt BaseStr}.$
	StartPos	Numeric value of the character position at which to begin the search. 1 is the first character and default value.
	Plain	Boolean value that turns off pattern matching if true. Default value is false.
Return Value:	nil if no occurrence of SrchStr was found.	
	If found, character position of the first and last matching charact	
Example:	local i	
	Returns 4 i = string.find	d("My String", "S", 1, true)

""	Concatenates two strings and returns the resulting new string.	
Syntax:	String1String2	
Parameters:	String1 Base string.	
	String2	This string will be added to the end of the base string.
Return Value:	The combined string	g.
Example:	<pre>local sVar sVar = "The Point is ""ON"Output to Screen would be "The point is ON" orion.PrintDiag(sVar)</pre>	

# Advanced Math & Logic for Orion

#	Length Operator.	
Syntax:	# String	
Parameters:	String	Base string for which length is needed.
Return Value:	Length of the specified string.	
Example:	local i	
	i = # "OrionLX	"Returns 7
string.len	Returns the length	of the specified string.
Syntax:	string.len(String)	
Parameters:	String	The length of this string will be returned.
Return Value: Length of the specified string.		fied string.

Example:

local i

i = string.len("OrionLX and IO") --Returns 14

string.sub	Returns the substring of the specified string at the specified start position and until the specified end position.	
Syntax:	<pre>string.sub(String, Start [, End])</pre>	
Parameters:	String	The string from which to extract a portion.
	Start	Number indicating the starting position in String at which the substring will be extracted. The first character is at position 1.
	End	Number indicating the ending position in String at which the substring will be extracted. If omitted, defaults to length of String.
Return Value:	The extracted string.	
Example:	<pre>local sVar sVar = string.sub("The Point is ON", 5, 12)Output to Screen would be "Point is" orion.PrintDiag(sVar) sVar = string.sub("The Point is ON", 5)Output to Screen would be "Point is ON" orion.PrintDiag(sVar)</pre>	
string.lower	Returns a copy of the to lowercase.	e specified string with all uppercase letters changed
Syntax:	string.lower(St	ring)
Parameters:	String	Base string to convert to lowercase.
Return Value:	String with all uppercase letters changed to lowercase.	
Example:	local sVar	
	sVar = string.l	ower("OrionLX")Returns "orionlx"

string.upper	Returns a copy of the specified string with all lowercase letters changed to uppercase.	
Syntax:	<pre>string.upper(String)</pre>	
Parameters:	String Base string to convert to uppercase.	
Return Value:	String with all lowercase letters changed to uppercase.	
Example:	local sVar	
	sVar = string.u	<pre>upper("OrionLX")Returns "ORIONLX"</pre>

orion.StrComp	Compares two strings case-insensitively.	
Syntax:	orion.StrComp(String1, String2)	
Parameters:	String1First string to compare.String2Second string to compare.	
Return Value:	<pre>-1 if String1 is less than String2. Example: "ABC" &lt; "ABD" 1 if String1 is greater than String2. Example: "ABD" &gt; "ABC" 0 if String1 is equal to String2. Example: "abc" = "abc"</pre>	
Example:	<pre>local iReturns 1 i = orion.StrComp("My String2", "My String")</pre>	

orion.Trim	Returns the specified string with leading and lagging spaces removed.
Syntax:	orion.Trim(String)
Parameters:	String The string to be trimmed
Return Value:	The specified string minus leading and trailing spaces.
Example:	local sVar
	Returns "My String" sVar = orion.Trim(" My String ")



orion.InStr	Returns the position of the first occurrence of a search string within a base string.		
Syntax:	orion.InStr(StartPosition, BaseString, SearchString)		
Parameters:	StartPosition	Numeric value of the character position at which to begin the search. 1 is the first character.	
	BaseString	The string which will be searched for SearchString.	
	SearchString	The string which will be searched for in BaseString.	
Return Value:	0 if no occurrence o If found, character p	f SearchString <b>was found.</b> position of the first matching character.	
Example:	local i		
	i = orion.InSt	r(1, "My String", "S")Returns 4	
orion.Join	Concatenates two s	trings and returns the resulting new string.	

Syntax:	orion.Join(Str.	ing1, String2)
Parameters:	String1	Base string.
	String2	This string will be added to the end of the base string.
Return Value:	The combined strin	g.
Example:	local sVar	
	<pre>sVar = orion.Join("The Point is ", "ON")Output to Screen would be "The point is ON" orion.PrintDiag(sVar)</pre>	

orion.Left	Returns the specified numbers of leftmost characters.		
Syntax:	orion.Left(String, Length)		
Parameters:	String Base string.		
	Length	Number of characters that are to be returned, starting from the beginning of the string.	
Return Value:	String with the specified number of leftmost characters.		
Example:	local sVar		
	<pre>sVar = orion.Left("OrionLX", 4)Returns "Orio"</pre>		

orion.Mid	Returns a specified portion of a string.		
Syntax:	<pre>orion.Mid(String, Start[, Length])</pre>		
Parameters:	String The string from which to extract a portion.		
	Start	Number indicating the starting position in $String$ at which the substring will be extracted. The first character is at position 1.	
	Length	The length of the substring to be extracted. If this optional parameter is omitted, the remainder of String is returned.	
Return Value:	The extracted string.		
Example:	local sVar		
	<pre>sVar = orion.Mid("The Point is ON", 5, 8)Output to Screen would be "Point is" orion.PrintDiag(sVar)</pre>		
	<pre>sVar = orion.Mid("The Point is ON", 5)Output to Screen would be "Point is ON" orion.PrintDiag(sVar)</pre>		

# **NovaTech**

orion.Right	Returns the specified numbers of rightmost characters, counting from the end of the string.		
Syntax:	orion.Right(Str	ing, Length)	
Parameters:	String Base string.		
	Length	Number of characters that are to be returned, counting from the last character of the string.	
Return Value:	String with the specified number of rightmost characters.		
Example:	local sVar		
	sVar = orion.Ri	ght("OrionLX", 4)Returns "onLX"	

### Advanced Math & Logic for Orion

### **Timer Operations**

orion.DisableTimer	Disables the specified timer.		
Syntax:	orion.DisableTimer(String)		
Parameters:	String	Name of the timer that is to be disabled.	
Return Value:	None		
Example:	orion.DisableT	Timer("Timer1")Disables Timer1	
orion.EnableTimer	Enables the specified timer.		
Syntax:	orion.EnableTi	<pre>mer(String [, Interval])</pre>	
Parameters:	String Name of the timer that is to be enabled.		
	Interval	Optional argument to specify timer interval in milliseconds. If omitted, interval specified at time of creation will be used.	
Return Value:	None		
Example:	orion.EnableTi	mer("Timer2") Enables Timer2	
orion.CreateTimer	Creates the specifi	ed timer (can be called in LOAD function only).	
Syntax:	orion.CreateTi	<pre>mer(String, Interval [, Enable])</pre>	
Parameters:	String	Name of the timer that is to be created.	
	Interval	How often the timer executes in milliseconds.	
	Enable	How often the timer executes in milliseconds. Optional boolean value specifying whether timer is initially enabled or disabled. If omitted, timer will be enabled.	
Return Value:	Enable None	How often the timer executes in milliseconds. Optional boolean value specifying whether timer is initially enabled or disabled. If omitted, timer will be enabled.	

orion.DisableTimer2	Disables the specified timer.		
Syntax:	orion.DisableTimer2(Handle)		
Parameters:	Handle	Handle of the timer that is to be disabled.	
Return Value:	None		
Example:	orion.Disable	<pre>Fimer2(timer1_handle)Disables Timer1</pre>	
orion.EnableTimer2	Enables the speci	fied timer.	
Syntax:	orion.EnableT:	imer2(Handle [, Interval])	
Parameters:	Handle	Handle of the timer that is to be enabled.	
	Interval	Optional argument to specify timer interval in milliseconds. If omitted, interval specified when timer was created will be used.	
Return Value:	None		
Example:	<pre>orion.EnableTimer2(timer2_handle)Enables Timer2</pre>		
orion.CreateTimer2	Creates the specif	ied timer (must be called in LOAD function only).	
Syntax:	orion.CreateT: Enable])	<pre>imer2(Function, Object, Interval [,</pre>	
Parameters:	Function	Function to be called when a timer event occurs based on the specified interval.	
	Object	Table containing the function. The default is a normal, global function if this argument is nil.	
	Interval	How often the timer executes in milliseconds.	
	Enable	Optional boolean value specifying whether timer is initially enabled or disabled. If omitted, timer will be enabled.	
Return Value:	Handle	Timer handle for use in orion.DisableTimer2 and orion.EnableTimer2 function calls.	

### Advanced Math & Logic for Orion

### **Time Operations**

orion.GetDay	Extracts the day fr	om a date/time string.		
Syntax:	orion.GetDay(	String)		
Parameters:	String	Date/time string MM/DD/YYYY HH:MM:SS.	with	format
Return Value:	Returns the day.			
Example:	local i			
	Returns 23 i = orion.Get	Day("11/23/2007 12:14:20	0")	
orion.GetHour	Extracts the hour	from a date/time string.		
Syntax:	orion.GetHour	(String)		
Parameters:	String	Date/time string MM/DD/YYYY HH:MM:SS.	with	format
Return Value:	Returns the hour.			
Example:	local i			
	<i>Returns 12</i> i = orion.Get	Hour("11/23/2007 12:14:2	20")	
orion.GetMinute	Extracts the minut	e from a date/time string.		
Syntax:	orion.GetMinu	te(String)		
Parameters:	String	Date/time string MM/DD/YYYY HH:MM:SS.	with	format
Return Value:	Returns the minut	e.		
Example:	local i			
	<i>Returns 14</i> i = orion.Getl	Minute("11/23/2007 12:1	4:20")	

orion.GetMonth	Extracts the month	n from a date/time string.		
Syntax:	orion.GetMont	n(String)		
Parameters:	String	Date/time string MM/DD/YYYY HH:MM:SS.	with	format
Return Value:	Returns the month	1.		
Example:	local i			
	Returns 11 i = orion.GetN	Month("11/23/2007 12:14	:20")	
orion.GetSecond	Extracts the secor	nd from a date/time string.		
Syntax:	orion.GetSeco	nd (String)		
Parameters:	String	Date/time string MM/DD/YYYY HH:MM:SS.	with	format
Return Value:	Returns the secon	d.		
Example:	local i			
	<i>Returns 20</i> i = orion.Get	Second("11/23/2007 12:14	4:20")	
orion.GetYear	Extracts the year f	rom a date/time string.		
Syntax:	orion.GetYear	(String)		
Parameters:	String	Date/time string MM/DD/YYYY HH:MM:SS.	with	format
Return Value:	Returns the year.			
Example:	local i			
	Returns 200 i = orion.Get	7 Year("11/23/2007 12:14:2	20 <b>")</b>	

orion.GetDOW	Extracts the day of the week from a date/time string – 0=Sunday, 1=Monday, , 6=Saturday		
Syntax:	orion.GetDOW(String)		
Parameters:	String Date/time string with format MM/DD/YYYY HH:MM:SS.		
Return Value:	Returns the day of the week.		
Example:	local i		
	Returns 5 for Friday i = orion.GetDOW("11/23/2007 12:14:20")		
orion.LocalTime	Returns the local date/time at the time of the call.		
Syntax:	orion.LocalTime()		
Parameters:	None.		
Return Value:	Returns a string with the current local date/time.		
Example:	local sVar		
	Returns "11/23/2007 12:14:20" sVar = orion.LocalTime()		
orion.SystemTime	Returns the UTC date/time at the time of the call.		
Syntax:	orion.SystemTime()		
Parameters:	None.		
Return Value:	Returns a string with the current UTC date/time.		
Example:	local sVar		
	Returns "11/23/2007 18:14:20" sVar = orion.SystemTime()		

# Advanced Math & Logic for Orion

### **Alarming Operations**

orion.GetUnacknowledge dAlarmCount	Retrieves the current number of unacknowledged alarms on the Orion. Unless the Return to Normal parameter is used, this operation only returns unacknowledged alarms in an alarm state.	
Syntax:	<pre>orion.GetUnacknowledgedAlarmCount(Zone, [Return to Normal])</pre>	
Parameters:	Zone	Optional filter. The number corresponding to an alarm zone. This parameter filters results to show only alarms from the specified zone.
	Return to Normal	Optional boolean value. If true, returns unacknowledged alarm points that have returned to normal in addition to the alarm points in an alarm state. If false or left blank, the operation only returns unacknowledged alarms in an alarm state.
Return Value:	Returns the current number of unacknowledged alarms or -1 if the information is not available.	
Example:	local i	
	Returns count of i = orion.GetUnac	of all unacknowledged alarms knowledgedAlarmCount()
	Returns count of all unacknowledged alarms for zone 1 including Return to Normal alarms i = orion.GetUnacknowledgedAlarmCount(1, true)	

orion.AcknowledgeAllAlarms	Acknowledges all unacknowledged alarms.		
Syntax:	orion.AcknowledgeAllAlarms(Zone)		
Parameters:	Zone Optional filter. The number corresponding to an alarm zone. This parameter only acknowledges alarms for the specified zone if the point is configured with no Alarm Acknowledge Group in the Alarm/Archive/Retentive module. Returns true on success or false on failure.		
Return Value:	Returns true on suc	ccess or false on failure.	
Return Value: Example:	Returns true on suc	ccess or false on failure.	
Return Value: Example:	Returns true on sur local ok Acknowledges ok = orion.Ack	ccess or false on failure. all unacknowledged alarms nowledgeAllAlarms()	



# Advanced Math & Logic for Orion

### Port Operations

orion.ResetPortCounters	Resets the current number of polls/responses for the specified port.	
Syntax:	orion.ResetPortCounters(Port)	
Parameters:	Port	String containing number or letter corresponding to NCD port.
Return Value:	Returns true on success or false on failure.	
Example:	local ok	
	Resets polls ok = orion.Res	<pre>s/responsee for port 1 etPortCounters(``1")</pre>



# Advanced Math & Logic for Orion

### **Objects and Object Operations**

orion.point	Returns a point object given a point name.		
Syntax:	<pre>orion.point(String)</pre>		
Parameters:	String Name of the point whose object is to be obtained. Example: Breaker Status @Feeder1		
Return Value:	Returns a point object corresponding to the specified name.		
Example:	local pt1		
	<pre>pt1 = orion.point("Breaker Status @Feeder1")</pre>		
:get	Returns a table of point attributes for the specified point object. These attributes include "name", "index", "value", "changes", "online" status, and associated "string", if applicable.		
Syntax:	:get()		
Parameters:	None.		
Return Value:	A table containing the current name, index, value, number of changes, and online/offline status, as well as string, if applicable, of the specified point.		
Example:	<pre>local pt1 local i = {}</pre>		
	<pre>pt1 = orion.point("Breaker Status @Feeder1")</pre>		
	Returns the current attributes of point Breaker Status @Feeder1 i.online or i["online"] contains the online/offline status i = ptl:get()		

:set	Sets the "value" "offtime", and/or "p will also automatic interested modules	, "changes", "online" status, "string", "duration", ulses" of the specified database point. This operation cally generate a change notification to inform other s of the point's change.	
Syntax:	:set (PointAttributes)		
Parameters:	PointAttributes table entries:		
	value	New value that the point will be set to. This value should be within the set scaling range for the point. However, setting the value below or beyond the set scaling is not restricted	
	changes	This parameter is optional and sets the number of changes for the point	
	online	This parameter is optional and sets the point's online status if desired. False sets the point to offline.	
	duration	This parameter is optional and sets the point's control duration in milliseconds. This parameter will be used in conjunction with a Trip/Close point	
	offtime	This parameter is optional and sets the point's time between pulses in milliseconds. This parameter will be used in conjunction with a Trip/Close point operation	
	pulses	This parameter is optional and sets the point's number of pulses. This parameter will be used in conjunction with a Trip/Close point operation	
	string	This parameter is optional and sets the point's associated text.	
Return Value:	None.		
Example:	local pt1		
	pt1 = orion.pc	pint("Breaker Status @Feeder1")	
	Sets the Breaker Status @Feeder1 point online and sets its value to 1 pt1:set({value=1, online=true})		

### Advanced Math & Logic for Orion

orion.timer	Returns a timer object given a timer name.	
Syntax:	orion.timer(String)	
Parameters:	String	Name of the timer whose object is to be obtained. Example: Timer1
Return Value:	Returns a timer object corresponding to the specified name.	
Example:	local tmr1	
	tmr1 = orion.t	<pre>imer("Timer1")</pre>

:disable

#### Disables the specified timer.

Syntax:	:disable()		
Parameters:	None		
Return Value:	None		
Example:	local tmr1		
	<pre>tmr1 = orion.timer("Timer1")</pre>		
	<pre>tmr1:disable()Disables Timer1</pre>		

:enable	Enables the specified timer.	
Syntax:	:enable([Interval])	
Parameters:	Interval	Optional argument that specifies the interval in milliseconds at which the timer executes. If omitted, the interval specified at timer creation will be used.
Return Value:	None	
Example:	local tmr1	
	tmr1 = orion.t	<pre>imer("Timer1")</pre>
	<pre>tmr1:enable()</pre>	Enables Timer1



orion.device	Returns a device object given a device name. This object is currently only available for DNP, Modbus and SEL client (serial and network) devices.		
Syntax:	orion.device(String)		
Parameters:	String Name of the device whose object is to be obtained. Example: "Feeder1"		
Return Value:	Returns a device object corresponding to the specified name.		
Example:	local dev1		
	<pre>dev1 = orion.device("Feeder1")</pre>		
:get	Returns a table of device attributes for the specified device object. These attributes include "online" status, "scan" status, and all device configuration parameters. The "scan" status attribute will currently only be available on active DNP, Modbus and SEL client (serial and network) ports.		
Syntax:	:get()		
Parameters:	None.		
Return Value:	The current online/offline status, on-scan/off-scan status, as well as all configured custom device parameters.		
Example:	<pre>local dev1 local i = {}</pre>		
	<pre>dev1 = orion.device("Feeder1")</pre>		
	Returns the current attributes of device Feeder1 i.online or i["online"] contains the online/offline status i = dev1:get()		

:set	Sets the "online" status, "scan" status, polls/responses "counter-reset", "points-online" status, and any other device configuration parameter that is allowed to be changed for the specified device object. This operation will currently affect only active DNP, Modbus and SEL client (serial and network) devices. Only DNP client devices support the "Event Polls" configuration parameter.		
Syntax:	:set(DeviceAttributes)		
Parameters:	DeviceAttributes table entries:		
	online	This parameter is optional and sets the device's online status if desired. False sets the device to offline True sets the device to online	
	scan	This parameter is optional and sets the device's scan/poll status if desired. False sets the device off-scan True sets the device on-scan	
	counter-reset	This parameter is optional and resets the device's polls and responses counters if desired. False does nothing. True resets the device polls/responses to zero.	
	points-online	This parameter is optional and sets the device's point online status if desired. False sets the device points to offline True sets the device points to online	
	Event Polls	This parameter is applicable to a DNP client device only and sets the device's event poll frequency in milliseconds.	
Return Value:	None.		
Example:	local dev1		
	<pre>dev1 = orion.device("Feeder1")</pre>		
	<pre>Takes the Feederl device off-scan, offline, and sets all of the device's points offline dev1:set({scan=false, online=false, ["points- online"]=false}) Sets the Feederl device on-scan, online, and sets the device's polls/responses to zero dev1:set({scan=true, online=true, ["counter- reset"]=true})</pre>		

orion.rtu	Returns an rtu object given an rtu name. This object is currently only available for DNP server (serial and network) and Modbus server (serial and network) RTUs.	
Syntax:	orion.rtu(String)	
Parameters:	String Name of the rtu whose object is to be obtained. Example: "To_EMS_DNP"	
Return Value:	Returns an rtu object corresponding to the specified name.	
Example:	local rtul	
	rtul = orion.r	tu("To_EMS_DNP")
:get	Returns a table of attributes include parameters.	f rtu attributes for the specified rtu object. These "polls", "responses", and all rtu configuration
Syntax:	:get()	
Parameters:	None.	
Return Value:	The current polls, parameters.	responses, as well as all configured custom rtu
Example:	<pre>local rtu1 local i = {}</pre>	
	rtul = orion.r	tu("To_EMS_DNP")
	Returns the rtu To_EMS_L i.polls or i polls counte i = rtul:get()	current attributes of NP ["polls"] contains the r

orion.pollgroup	Returns a poll grou	p object, given a port and poll group name.
Syntax:	orion.pollgrou	p(Port, String)
Parameters:	Port	Number or letter corresponding to an NCD port.
	String	Name of the poll group whose object is to be obtained. Example: "Default Poll Group"
Return Value:	Returns a poll grou	p object corresponding to the specified name.
Example:	local grp1	
	<pre>grp1 = orion.p</pre>	ollgroup("1", "Default Poll Group")
:get	Returns a table of p These attributes ind and "milliseconds".	oll group attributes for the specified poll group object. clude "name", "days", "hours", "minutes", "seconds",
Syntax:	:get()	
Parameters:	None.	
Return Value:	The current numbe of the specified poll	r of days, hours, minutes, seconds, and milliseconds group.
Example:	<pre>local grp1 local i = {}</pre>	
	<pre>grp1 = orion.p</pre>	ollgroup("1", "Default Poll Group")
	Returns the Default Poll i.seconds or of seconds i = grpl:get()	current attributes of Group on Port 1 i["seconds"] contains the number

:set	Sets the "days", "h the specified poll g active DNP, Modb	ours", "minutes", "seconds", and/or "milliseconds" for group object. Currently, this operation will only affect us and SEL client (serial and network) ports.
Syntax:	:set(PollGroup	DAttributes)
Parameters:	PollGroupAttri	ibutes table entries:
	days hours minutes seconds milliseconds	Number of days for the poll group. Number of hours for the poll group. Number of minutes for the poll group. Number of seconds for the poll group. Number of milliseconds for the poll group.
Return Value:	None.	
Example:	local grp1	
	grp1 = orion.p	<pre>bollgroup("1", "Default Poll Group")</pre>
	Sets the num Default Poli to 30 grpl:set({seco	mber of seconds for 1 Group on Port 1 onds=30})

# Advanced Math & Logic for Orion

### Wake Operations

orion.wake	This command reactivates the Orion display screen from rest mode. This command only applies to OrionLX-CPX and OrionLX+ models.				
Syntax:	<pre>orion.wake(":[Display.Screen]")</pre>				
	or				
	orion.wake()				
Parameters:	Display Screen	This value refers to a particular display screen by number. The wake function assumes 0 if no number is specified. Add a colon in front of the display screen number as seen in the examples.			
Return Value:	Returns 0 on success,	and can return either 1, 2, or 3 on failure.			
Example:	local I				
	Returns "0" i = orion.wake(":	0.0")			

### Appendix B – Tools and Buttons on the Input and Output Menus

For easy configuration, numerous tools and buttons are available as shown in <u>Figure 47</u> and described in the sections below. The tools and buttons described in this appendix are available for the <u>Inputs Menu</u>, <u>Outputs Menu</u>, <u>Logic Inputs Menu</u>, and <u>Logic Outputs Menu</u>. These menus are very similar, so one sample screenshot is provided for each feature.



Figure 47: Tools on Input, Output, Logic Input and Logic Output Menus

### 1. Display/Hide Tag Name List

Clicking the  $\bigcirc$  button under the Input/Outputs lists hides the list, thus providing more usable screen space. Clicking the  $\bigcirc$  button displays the list again.

### 2. Display/Hide Menu Tree

Clicking the ③ button to the left of the Standard/Functions tree hides the tree, thus providing more usable screen space. Clicking the ③ button displays the tree again.



#### 3. Search Point Name Lists

The Point Name list and the Orion Point Name list can be searched using free text. Enter a free text word such as volts in the search field. In this case, all point names containing "volts" will be filtered and displayed.

Inputs			Ing	puts (4 Item(s) - Filter By: All Points)	Font 72
🔎 volts 🗸 🗸			P	volts 🚺 💽 👔	Configured Inputs
Point Name	Alias			Orion Point Name	Alias
Orion 12 Volts @Orion Internal	Orion Internal_Orion 12 Volts		t.	Orion 12 Volts @Orion Internal	Orion Internal_Orion 12 Volts ^
Orion 3A Volts @Orion Internal	Orion Internal_Orion 3A Volts		1+	Orion 3A Volts @Orion Internal	Orion Internal_Orion 3A Volts
Orion 3B Volts @Orion Internal	Orion Internal_Orion 3B Volts	<<	14	Orion 3B Volts @Orion Internal	Orion Internal_Orion 3B Volts
Orion 5 Volts @Orion Internal	Orion Internal_Orion 5 Volts		14	Orion 5 Volts @Orion Internal	Orion Internal Orion 5 Volts
			<		>

Figure 48: Filter Point Name List

The search field can be cleared by clicking on the Clear Search button, 🔟 to the right of the search field. Then the complete points list will be displayed again.

#### 4. Toggle Show Point Details

To view configuration details of one or more points, highlight the point(s) in the Point Name list (left column) or the Orion Point Name list (right column) and click the Toggle Show Details for Selected Rows button . An expanded window appears below the selected point and lists detailed information related to that point (Figure 49).

Logic Client Port 124 - Advanced M	ath and Logic (OrionLX_Demo.ncd)						
Port Options: Delete Port Close Port	0 0 0 🖻 🖉 🗇 🖫 📮 🔎	🔀 🛛 🔭 Setting:	Simulate	Line: 4 Cha	racter: 1		
	function DefaultStart: end	Function()					۹ ۲
<ul> <li>Define</li> <li>Evaluation</li> </ul>	<						>
Loop	<ul> <li>No code errors detected</li> </ul>						
▷ Math b Orion	Inputs			Inputs (4 Ite	em(s) - Filter By: All Poin	ts) Font	12
String	🔎 volts 🗸 🗸			🔎 volts	× 0	👔 📝 🧮 Configured Inputs	
	Point Name	Alias		Orion F	Point Name	Alias	
	Orion 3A Volts @Orion Internal	Orion Internal_Orion 3A Volts	^ >>	Orion 3B	Volts @Orion Internal	Orion Internal_Orion 3B Volts	^
	Orion 38 Volts @Orion Internal           Port 21           Port Name Client           Device Company, NevaTach           Device Model OrionL/, Pluss           Point Name: Orion 38 Volts           Point Name: Orion 38 Volts           Point Name: Orion 38 Volts           Max Value: 25	Onion Internal_Orion 3B Volts		Port: 21 Port Name: S Device: Orior Device Comp Device Comp Device Mode Point Name: Point Type: v Min Value: 0 Max Value: 2	Sensor Client n Internal any: NovaTech I: Orion JZ Volts roc 255		
	Orion 5 Volts @Orion Internal	Orion Internal_Orion 5 Volts		Orion 5 V	/olts @Orion Internal	Orion Internal_Orion 5 Volts	
	$\odot$		v	<			> ×

Figure 49: Show Point Details

To close the point details display, click the Toggle Show Details for Selected Rows button.



#### 5. Display/Hide Alias Name

If the Add Ons option Alias Module has been configured, the Toggle Alias Column Visibility button is displayed above the Orion Point Name list in the right pane. Clicking this button will toggle the display of the Alias points name column.

#### 6. View Report

The View Report button generates a printable report for the points which are currently displayed in the Orion Point Name list. This report can be printed, previewed, or saved as an HTML Web Page or as a CSV file on the PC.

Report - All Points							
Orion Point Name	Alias	Event Event Type	Deadband	Min	Max	Queue Size	$\sim$
Orion 12 Volts @Orion Internal	Orion Internal_Orion 12 Volts		25.5	0	255	0	
Orion 3A Volts @Orion Internal	Orion Internal_Orion 3A Volts		25.5	0	255	0	
Orion 3B Volts @Orion Internal	Orion Internal_Orion 3B Volts		25.5	0	255	0	
Orion 5 Volts @Orion Internal	Orion Internal_Orion 5 Volts		25.5	0	255	0	
							$\sim$
			Print	Print	Preview	Save As	Close

Figure 50: View Report

#### 7. Edit Common Attributes

Note that all settings made in the Edit Common Attributes window can also be set directly with each individual data point.

Attributes which should be identical for multiple points configured for logic can be edited in a single window. First, highlight all points for which the attributes shall be set up identically. Then click on the Edit Common

Attributes button 🔟.

At this point, the Edit Common Attributes window opens (Figure 51). Only attributes which are common to all selected points are displayed, select the attribute to be changed by clicking its checkbox, make the desired changes and click Apply. Any changes made in this window are applied to all selected points (Figure 52).

### **NovaTech**

### Advanced Math & Logic for Orion

Logic Client Po	rt 124 - Advanced Math and Logic (OrionLX_De				
Port Options: Del	A Edit Common Attributes	× × Settings	Simulate Line: 4 Character: 1		
Standard     Inputs     Outputs     Logic Inputs     Logic Inputs     Logic Outputs     Events     Commands     P All     D Control     D Define     E Evaluat     D Loop     Math	the fall common Attributes:         Only velected attributes will be changed.         fall common Attributer from the AII Pelets tabs         point by provide attributes for that give         Vent:         V	Details isplays a subset of attributes. Individual v	Inputs (4 Item(s) - Filter By, A)	IPoints) Fort	*
₀ Onon ₀ String	Queue Size: 0	12 bit usinged min 16 bit usinged min 16 bit unsigned min 32 bit unsigned min 32 bit unsigned min Apply Cancel ternal Orion Internal_Orion 5 Volta	vola     vola	Configured Inputs Configured Inputs Configured Inputs Configured C	Event Type

Figure 51: Edit Common Attributes

 $\label{eq:clicking on Details in the Edit Common Attributes window lists the points for which any settings made in that window will be applied to.$ 

Logic Client	Port 124 - Advanced M	Nath and Logic (OrionLX_Dem							×
Port Options:	Dele 🍁 Edit Commo	n Attributes	×	📉 🔀 🛛 🚰 Settings 🖌 Simu	ilate	Line: 4 Character: 1			
<ul> <li>Standard</li> </ul>									^
Inputs	Only selected a	ttributes will be changed.	Details						
Outputs Logic Inp	Edit Common Attr point type tabs pri	ibutes from the <b>All Points</b> tab di wide attributes for that type.	Points to be changed: (4 Ite	m(s)) ×					
Events	Event:	SampleEvent	Orion Point Name	Alias					
⊿ Comman	nds		Orion 12 Volts @Orion Interr	nal Orion Internal_Orion 12 Volts					
D All	Event Type:	DeadBand	Orion 3A Volts @Orion Interr	nal Orion Internal_Orion 3A Volts					
Defi	Deadband:	25.5	Orion 3B Volts @Orion Interr	nal Orion Internal_Orion 3B Volts					~
Eval	uati	0	Orion 5 Volts @Orion Interna	al Orion Internal_Orion 5 Volts					>
Loop	Min:	0							
Math Original	Max	255				Inputs (4 Item(s) - Filter By: A	Il Points)	Font 🕖	12
5 Strin	9 Oueue Size	0				🔎 volts 🗙	💽 🖹 📝 📃 Configured	lanuts	
					7	Orion Paint Nama	Aline	Event	Event Tune
				Orion 34 Volte	_	Orion 12 Volte @Orion Interne	Orion Internal Orion 12 Volte	Lven	Event type
				Orion 3B Volts	· >>	Orion 34 Volte @Orion Interna	Orion Internal Orion 34 Volte		
					~	Orion 3B Volte @Orion Interna	Orion Internal, Orion 3B Volte		
			Apply Cancel			Orion 5 Volts @Orion Internal	Orion Internal, Orion 5 Volts		
			Apply Califer						
		Point Type: voc Min Value: 0 Max Value: 255 Orion 5 Volts @Orion Int	ernal Orion Internal	L_Orion 5 Volts					
					-	<			>
		$\overline{\mathbf{v}}$			_				

Figure 52: Edit Common Attributes - Details



### 8. Font Size

The font size of the  $\,$  Orion  $\,$  Point  $\,$  Name  $\,$  list can be adjusted by moving the slider.

Logic Client Port 124 - Advan	ed Math and Logic (OrionLX_Demo.ncd)*						- 0 %
Port Options: Delete Port Clos	2 Port 🗋 📔 🗇 😰 🖄 🖫 🏪 🛛 🔎			🗡 🛛 🚰 Settings 🎽 Simulate	Line:	4 Character: 1	
Standard Punctions     Inputs     Outputs     Logic Inputs     Logic Inputs     Logic Vutputs     Events     Commands     P All     Control     Defen	<ul> <li>function DefaultStartFunction</li> <li>end</li> <li></li> <li>No code errors detected</li> </ul>	D					^ >
beine ► Evaluation ► Loop	Inputs		Inputs	(4 Item(s) - Filter By: All Point	s)	Font	14
⊳ Math	🔑 volts 🗸 🔀 💽		p vol	s 🔀 🖸		Configured Inp	outs
⊳ Orion	D. LIN				Event	F . T	
► String	Point Name		C	rion Point Name	Event	Event Type	Deadba
b String	Orion 12 Volts @Orion Internal	>>	0 🕁	rion Point Name rion 12 Volts @Orion Internal	Event	Event Type	Deadba 25.5
b String	Orion 12 Volts @Orion Internal Orion 3A Volts @Orion Internal Orion 3A Volts @Orion Internal Orion 3B Volts @Orion Internal	>>	0	rion Point Name rion 12 Volts @Orion Internal rion 3A Volts @Orion Internal	Event	Event Type	25.5 25.5
b String	Orion 12 Volts @Orion Internal     Orion 3A Volts @Orion Internal     Orion 3A Volts @Orion Internal     Orion 5 Volts @Orion Internal	>> <<		rion Point Name rion 12 Volts @Orion Internal rion 3A Volts @Orion Internal rion 3B Volts @Orion Internal	Event	Event Type	25.5 25.5 25.5
b String	Voits @Orion Internal     Orion 12 Volts @Orion Internal     Orion 38 Volts @Orion Internal     Orion 5 Volts @Orion Internal	»» «		rion Point Name rion 12 Volts @Orion Internal rion 3A Volts @Orion Internal rion 3B Volts @Orion Internal rion 5 Volts @Orion Internal	Event	Event Type	Deadba 25.5 25.5 25.5 25.5 25.5

Figure 53: Font Size Slider



### Advanced Math & Logic for Orion

#### Appendix C – Logic in .ncd and .lua File

Below is the logic section from a .ncd file. The configuration is separated into six headings.

The [Logic General] heading includes a comma-separated list of .lua source files and, if applicable, the name of the function executed at startup.

The [Logic Inputs Master] heading contains a list of input points configured for use with the logic sub system. Each point has a name, min scaling value, max scaling value, deadband, change function, and change type. The deadband value is used for detecting data changes outside of the deadband range, while the change function is optional. The change type can be either Refresh for every database update or Deadband for every database change outside of the deadband range. These points are configured under the Logic Inputs Menu.

The [Logic Outputs Master] heading contains a list of output points and has identical parameters to the [Logic Inputs Master] heading. These points are configured under the Logic Outputs.

[Logic Inputs Slave] and [Logic Outputs Slave] are identical to [Logic Inputs Maser] and [Logic Outputs Master] respectively, with the exception that points listed in the server headings are pseudo points. Points are configured under the <u>Inputs Menu</u> and the <u>Outputs Menu</u>, respectively.

The [Logic Timers] heading contains a list of defined timers. Each entry has a timer name, an interval in ms, and an initial enabled state.

[Logic General] SourceFiles=logic2.lua OnLoad=MyStartFunction [Logic Inputs Master] Point001 @Device 1, 0, 65535,1, MyFirstChangeEvent, Refresh Point003 @Device 1, 0, 65535,1, MySecondChangeEvent, Refresh Point003 @Device 1, 0, 65535,1, MySecondChangeEvent, Refresh BinaryOutput0 @Device 1, 0, 1, 1, DisableMyTimer, Refresh BinaryOutput1 @Device 1, 0, 1, 1, EnableMyTimer, Refresh [Logic Inputs Slave] Pseudol @Device 1, 0, 65535,1, MyThirdChangeEvent, DeadBand [Logic Outputs Slave] [Logic Outputs Slave] Commented [PR1]: Do we need this?


## Appendix D – Lua Quick Reference Libraries

### The Lua Cheat Sheet

Reserve	d ident	ifiers	and com	ments							
and	break	do	else	elseif	end	false	for	functio	o if	in	
local	nil	not	or	repeat	return	then	true	until	while		
	commer	nt to end	of line	-	[=[	multi line	e comment	(zero or mi	ultiple '=' ar	e valid)	
x is "res	served" (by	y conver	ntion) for cons	stants	#!	usual U	nix sheban	; Lua ignoi	es whole fi	rst line if this starts the lin	e.
(with X b	eing anv s	sequenc	e of uppercas	e letters)				0			
Types (t	he strir	na val	ues are th	ne nossil	ole resu	ults of b	ase libr	arv fund	tion tyr	)) )	
"nil"	"bo	olean"	"number"	"strin	a" "ta	able"	"functi	on" "thre	ad" "u	serdata"	
Note: for	type boole	an. nil :	and false cou	int as false: e	everything	else is tru	e (including	0 and "").			
Operato	rs. deci	reasir	a preced	ence				,			
^ (right	associativ	e. math	library require	ed)							
not		.,	#	# (length of	strings and	tables)	- (u	nary)			
*			/		_		%				
+					-						
. (string	concatena	ation, rig	ht associative	e)							
<		>	<	<=	>=		~=		==		
and (StO	ps on <b>fals</b>	e or nil,	returns last e	valuated val	ue)						
or (stop	s on true	(not fals	e or nil), retu	rns last eval	uated value	e)					
Assignn	nent an	d coe	rcion								
a = 5	b=	S	imple assignr	ment; variabl	es are not	typed and	can hold d	ifferent type	es. Local va	riables are lexically scope	ed; their
"hi"		s	cope begins	after the ful	I declaratio	on (so that	local a = 5	).			
a, b, c	c = 1, 2	,3 n	nultiple assigr	nments are s	upported						
a, b =	b, a	S	wap values: r	right hand sid	de is evalu	ated befor	e assignme	ent takes pla	ace		
a, b =	4, 5, "	6" e	xcess values	on right han	d side ("6"	) are evalu	uated but di	scarded			
a, b =	"there"	fe	or missing val	ues on right	hand side	nil is assu	imed				
a = nil	a = nil destroys a; its contents are eligible for garbage collection if unreferenced.										
a = z	a = z if z is not defined it is nil, so nil is assigned to a (destroying it)										
a = "3'	a = "3" + "2" numbers expected, strings are converted to numbers (a = 5)										
a = 3	a = 3 2 strings expected, numbers are converted to strings (a = "32")										
Control	-4										
Control	structu	res									
do bl	ock end	1			blo	ock; introd	uces local :	scope.			
if exp	b then	block	{elseif e	xp then b	LOCK   CO	conditional execution					
while	exp ac	D10	ck end		100	oop as long as exp is true					
repeat	DIOCK	unti	L exp	1 -1 - 1 - 7 1	ex	numerical for loop: varia local to loop					
for v	ar - sta arc <b>in</b>	itor	ator <b>de</b>	block of	d ito	rator base	d for loop:	vars are loc	op. al to loop		
brook	a13 <b>11</b>	TLEI	acor <b>do</b>	DIOCK EI		avits loop; must be lost statement in block					
DIEak	break					ato 100p, 11		Statement	TI DIOCK.		
Function	a dofini	tion									
Function	i denni	uon									
functio	on name	( args	) body [1	eturn val	ues] de	tines tune	tion and as	signs to glo	Dai Variable	name	
local 1	unction	name	(args) L	ooay [retu	irn de	ennes iunc	tion as loca	i to chunk	richle <b>f</b>		
f = fur	iction (	args	) Doay [re	scurn vall	iesj an	ionymous	iuncion as				
functio	function ( [args, ] ) body [return			va ch	shortcut for t name – function						
functio	function obj:name ( args ) body [return			oh	object function, gets obj as additional first argument self						
runcer	<b>m</b> 005.11	ianic (	arg5 <b>,</b> boo	i feculi	. 05	joorranou	on, geto ob		iai mot argi		
Function	n call										
f (x)			simple cal	I, possibly re	turning on	e or more	values				
f "hell	Lo"		shortcut fo	or f("hello")							
f 'good	ibye '		shortcut fo	or f('goodby	e')						
f [[see	a you so	on]]	shortcut fo	or t([[see yo	u soon]])						
f {x =	3,у=	4}	shortcut fo	or $t({x = 3, y})$	= 4})	10.000					
t.f (x)			calling a fu	unction assig	ned to tiel	ar of table	t				
x:move	(2, -3)		object call	: snortcut for	x.move(x	(, 2, -3)					



### The Mathematical Library [math]

#### Basic operations

math.abs (x)	returns the absolute value of x
math.mod (x, y)	returns the remainder of $\mathbf{x} / \mathbf{y}$ as a rounded-down integer, for $\mathbf{y} \sim = 0$
math.floor (x)	returns x rounded down to the nearest integer
math.ceil (x)	returns x rounded up to the nearest integer
math.min (args)	returns the minimum value from the args received
math.max (args)	returns the maximum value from the args received

#### Exponential and logarithmic

math.sqrt (x)	returns the square root of $\mathbf{x}$ , for $\mathbf{x} \ge 0$
math.pow (x, y)	returns x raised to the power of y, i.e. $x^{y}$ ; if $x < 0$ , y must be integer.
pow (x, y)	global function added by the math library to make operator 'A' work
math.exp (x)	returns e (base of natural logs) raised to the power of x, i.e. e^x
math.log (x)	returns the natural logarithm of $\mathbf{x}$ , for $\mathbf{x} \ge 0$
math.log10 (x)	returns the base-10 logarithm of $\mathbf{x}$ , for $\mathbf{x} \ge 0$

#### Trigonometrical

0	
math.deg (a)	converts angle a from radians to degrees
math.rad (a)	converts angle a from degrees to radians
math.pi	constant containing the value of pi
math.sin (a)	returns the sine of angle <b>a</b> (measured in radians)
math.cos (a)	returns the cosine of angle a (measured in radians)
math.tan (a)	returns the tangent of angle a (measured in radians)
math.asin (x)	returns the arc sine of x in radians, for x in [-1, 1]
math.acos (x)	returns the arc cosine of <b>x</b> in radians, for <b>x</b> in [-1, 1]
math.atan (x)	returns the arc tangent of x in radians
<pre>math.atan2 (y, x)</pre>	similar to math.atan( $y / x$ ) but with quadrant and allowing $x = 0$

#### Splitting on powers of 2

 math.frexp
 (x)
 splits x into normalized fraction and exponent of 2 and returns both

 math.ldexp
 (x, y)
 returns x \* (2 ^ y) with x = normalized fraction, y = exponent of 2

#### Pseudo-random numbers

math.random ([n [, m]) returns a pseudo-random number in range [0, 1] if no arguments given; in range [1, n] if n is given, in range [n, m] if both n and m are passed

#### The String Library [string]

Note: string indexes extend from 1 to #string, or from end of string if negative (index -1 refers to the last character).

Note: the string library sets a metatable for strings where the index field points to the string table. String functions can be used in object-oriented style, e.g. string.len(s) can be written s:len(); literals have to be enclosed in parentheses, e.g. ("xyz"):len().

Basic operations				
string.len (s)	returns the length of string s, including embedded zeros (see also # operator)			
string.sub (s, i [, returns the substring of s from position i to j [default: -1] inclusive				
string.rep (s, n) returns a string made of <b>n</b> concatenated copies of string <b>s</b>				
<pre>string.upper (s)</pre>	returns a copy of <b>s</b> converted to uppercase according to locale			
<pre>string.lower (s)</pre>	returns a copy of <b>s</b> converted to lowercase according to locale			
Character codes				
string.byte (s [, i returns the platform-dependent numerical code (e.g. ASCII) of characters s[i], s[i+1],, s[j]. The default value for j is i.				
<pre>string.char (args)</pre>	returns a string made of the characters whose platform-dependent numerical codes are passed as args			
Function storage				
string.dump (f)	returns a binary representation of function f(), for later use with loadstring() (f() must be a Lua function with no upvalues)			
Formatting				
<pre>string.format (s [, args])</pre>	returns a copy of <b>s</b> where formatting directives beginning with '%' are replaced by the value of arguments args in the given order (see Formatting directives below)			
arg5]/	argo, in the given order (dee normaling uncerves below)			



### The IO Library

#### Complete I/O

<pre>io.open (fn [, m])</pre>	opens file with name fn in mode m: "r" = read [default], "w" = write", "a" = append, "r+" = update-preserve, "w+" = update-erase, "a+" = update-append (add trailing "b" for binary mode on some systems); returns a file object (a userdata with a C handle).
file:close ()	closes file
<pre>file:read (formats)</pre>	returns a value from file for each of the passed formats: "'n" = reads a number, "*a" = reads the whole file as a string from current position (returns "" at end of file), "t" = reads a line (nil at end of file) [default], $n$ = reads a string of up to $n$ characters (nil at end of file)
file:lines ()	returns an iterator function for reading file line by line; the iterator does not close the file when finished.
file:write (values)	writes each of the values (strings or numbers) to file, with no added separators. Numbers are written as text, strings can contain binary data (in this case, file may need to be opened in binary mode on some systems).
<pre>file:seek ([p] [,     of])</pre>	sets the current position in file relative to p ("set" = start of file [default], "cur" = current, "end" = end of file) adding offset
file:flush ()	flushes any data still held in buffers to file
iiic.iidan ()	

#### Simple I/O

<pre>io.input ([file])</pre>	sets file as default input file; file can be either an open file object or a file name; in the latter case the file is opened for reading in text mode. Returns a file object, the current one if no file given; raises error on
<pre>io.output ([file])</pre>	sets file as default output file (the current output file is not closed); file can be either an open file object or a file name; in the latter case the file is opened for writing in text mode. Returns a file object, the current one if no file given; raises error on failure.
<pre>io.close ([file])</pre>	closes file (a file object) [default: closes the default output file]
<pre>io.read (formats)</pre>	reads from the default input file, usage as file:read()
<pre>io.lines ([fn])</pre>	opens the file with name fn for reading and returns an iterator function to read line by line; the iterator closes the
	lie when inished. If no fin is given, returns an iterator reading lines from the default input life.
io.write (values)	writes to the default output file, usage as file:write()
io.flush ()	flushes any data still held in buffers to the default output file

#### Standard files and utility functions

io.stdin, io.stdout,	predefined file objects for stdin, stdout and stderr streams
io.popen ([prog [,	starts program prog in a separate process and returns a file handle that you can use to read data from (if
mode]])	mode is "r", default) or to write data to (if mode is "w")
io.type (x)	returns the string "file" if x is an open file, "closed file" if x is a closed file or nil if x is not a file object
io.tmpfile ()	returns a file object for a temporary file (deleted when program ends)

### The Operating System Library [os]

#### System interaction

os.execute (cmd)	calls a system shell to execute the string cmd as a command; returns a system-dependent status code.
os.exit ([code])	terminates the program returning code [default: success]
os.getenv (var)	returns a string with the value of the environment variable var or nil if no such variable exists
os.setlocale (s [,	sets the locale described by string s for category c: "all", "collate", "ctype", "monetary", "numeric" or "time"
c])	[default: "all"]; returns the name of the locale or <b>nil</b> if it can't be set.
os.remove (fn)	deletes the file fn; in case of error returns nil and error description.
os.rename (of, nf)	renames file of to nf; in case of error returns nil and error description.
os.tmpname ()	returns a string usable as name for a temporary file; subject to name conflicts, use io.tmpfile() instead.

#### Date/time

os.clock ()	returns an approximation of the amount in seconds of CPU time used by the program
os.time ([tt])	returns a system-dependent number representing date/time described by table tt [default: current]. tt must have
	fields year, month, day; can have fields hour, min, sec, isdst (daylight saving, boolean). On many systems
	the returned value is the number of seconds since a fixed point in time (the "epoch").
os.date ([fmt [,	returns a table or a string describing date/time t (should be a value returned by os.time() [default: current
t]])	date/time]), according to the format string fmt [default: date/time according to locale settings]; if fmt is "*t" or
	"!*t", returns a table with fields year (yyyy), month (112), day (131), hour (023), min (059), sec (061),
	wday (17, Sunday = 1), yday (1366), isdst (true = daylight saving), else returns the fmt string with formatting
	directives beginning with "%' replaced according to Time formatting directives (see below). In either case a
	leading "!" requests UTC (Coordinated Universal Time).
os.difftime (t2, t1)	returns the difference between two values returned by os.time()



## The Orion Library [orion]

Database Operations	
orion.GetPoint (name)	returns either a specific point attribute or a table of point attributes from the real-time database name point.
orion.SetPoint	sets point attribute(s) att of the real-time database name point.
(name, fn, [obj,	associates the <b>name</b> point with the specified data change function <b>fn</b> .
typ, db]) orion.GetQueuedPoint	returns a table of point attributes from the <b>name</b> point's change event. Returns <b>nil</b> if no events are available.
(name)	
Logging Operations	
orion.PrintDiag (s)	prints the string <b>s</b> to the Orion MMI View Communications menu or the Logic Simulator Diagnostic Output.
orion.PrintLog (s)	prints the string <b>s</b> to the Orion Event Log or the Logic Simulator Log Output.
Math Operations	returns the integer portion of number <b>num</b> with pagative numbers returning the pagative number greater than
	or equal to the number.
orion.int (num)	returns the integer portion of number <b>num</b> , with negative numbers returning the negative number less than or equal to the number.
Timer Operations	
<pre>orion.DisableTimer (name)</pre>	disables the specified <b>name</b> timer.
<pre>orion.EnableTimer (name)</pre>	enables the specified name timer.
<pre>orion.CreateTimer (name, intvl, [enbl])</pre>	creates the <b>name</b> timer using interval <b>intvl</b> and optionally enables/disables <b>enbl</b> it.
orion.DisableTimer2 (h)	disables the timer specified by handle h.
orion.EnableTimer2 (h)	enables the timer specified by handle h.
<pre>orion.CreateTimer2 (fn, obj, intvl, [enbl])</pre>	links function <b>fn</b> with a timer using interval <b>intvl</b> and optionally enables/disables <b>enbl</b> it.
Time Operations	
orion.GetDay (s)	returns the day from the date/time string s.
orion.GetHour (s)	returns the hour from the date/time string s.
orion.GetMinute (s)	returns the <b>minute</b> from the date/time string <b>s</b> .
orion.GetMonth (s)	returns the month from the date/lime string s.
orion.GetSecond s)	returns the second from the date/time string s.
orion.GetYear (s)	returns the vear from the date/time string s
orion.GetDOW (s)	returns the day of the week using the date/time string s
orion.LocalTime ()	returns a string with local data/time
orion.SystemTime ()	returns a string with LITC date/time



### Appendix E – Additional References

Document Title	File Name
OrionLX/OrionLX+ User Manual	OrionLX_User_Manual.pdf
OrionLXm User Manual	OrionLXm_User_Manual.pdf
OrionMX User Manual	OrionMX_User_Manual.pdf
Orion I/O User Manual	OrionIO_User_Manual.pdf
Analog/Accumulator Scaling	TechNote_Scaling.pdf

#### **Third-Party Lua References**

- Lua 5.3 Reference Manual http://www.lua.org/manual/5.3/manual.html
- Lua database connectivity: <u>https://keplerproject.github.io/luasgl/manual.html</u>
- Luars232 object: <u>http://lua-users.org/lists/lua-l/2012-09/msg00554.html</u> and <u>https://github.com/ynezz/librs232/blob/master/doc/example.lua</u>

Note that these references are not controlled by NovaTech and may be changed by the respective authors at any time.

# A NovaTech

# Advanced Math & Logic for Orion

Revision	Date	Changes
A	03/01/21	Initial release with logic Simulator and added NCD functionality.