















OOPic Objects

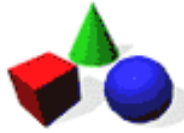


Hardware Objects.

Hardware Objects are Objects that encapsulate the functionality of the physical hardware circuits within the OOPic.

All Hardware Objects have a [Value](#) property that reflects the electrical state of the hardware that they encapsulate. Most can be turned on and off by setting an [Operate](#) property that specifies if the hardware function is active. Hardware Objects that do not have an [Operate](#) property can still be enabled and disabled by their other properties.













	<u>Object</u>	<u>Description</u>
	oA2D	An Object that provides a numerical measurement of a voltage.
	oDIO1	An Object that provides a 1 bit digital I/O.
	oDIO16	An Object that provides a 16-bit digital I/O.
	oDIO16x	An Object that provides a 16-bit digital I/O.
	oDIO4	An Object that provides a 4-bit digital I/O.
	oDIO8	An Object that provides an 8-bit digital I/O.
	oI2C	An Object that provides access to an I2C device.
	oKeypad	An Object that reads a 4 x 4 Keypad matrix.
	oPWM	An Object that provides a Pulse-Width-Modulated output.
	oSerial	An Object that provides an asynchronous serial I/O port.
	oServo	An Object that controls a servo-motor
	oTimer	An Object that provides a 16-bit high speed counter.

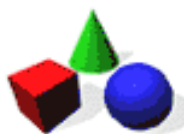


Processing Objects.

Processing Objects are Objects that encapsulate various mathematical, logical and other data manipulation functions.

Processing Objects perform various data manipulation functions. Where-as Variable and Hardware Objects maintain a default **Value** property, Processing Objects do not. Instead, they have Object and Flag pointers that specify which other Objects hold the **Value** properties that will be used for the function's input and output values. All Processing Objects can be turned on and off by setting an **Operate** property that specifies if the function is enabled..



	<u>Object</u>	<u>Description</u>
	<u>oConverter</u>	An Object that provides conversion functions.
	<u>oCounter</u>	An Object that provides counting functions.
	<u>oDataStrobe</u>	An Object that provides a Data-Strobe in response to a value being written to it.
	<u>oDDELink</u>	An Object that provides a Dynamic-Data-Exchange link over the I2C network.
	<u>oDebounce</u>	An Object that provides logic-state debouncing functions.
	<u>oEvent</u>	An Object that runs program code in response to an event.
	<u>oFanOut</u>	An Object that takes an input value and copies it to other objects.
	<u>oGate</u>	An Object that provides logic-gate functions.
	<u>oIndex</u>	An Object that provides indexing functions.
	<u>oMath</u>	An Object that provides mathematical functions.
	<u>oOneShot</u>	An Object that produces a one-pulse output in response to logic transition.
	<u>oRandomizer</u>	An Object that provides a random number.
	<u>oRTC</u>	An Object that maintains a Real Time Clock.




Variable Objects.

Variable Objects are Objects that store values.

All Variable Objects have a **Value** property that is used to store its value. In addition to the **Value** property, various other properties and methods are available to manipulate the variable.

	<u>Object</u>	<u>Description</u>
1	<u>oBit</u>	An Object that maintains a 1-bit variable.
88...	<u>oBuffer</u>	An Object that maintains a variable size data-buffer/string variable.
8	<u>oByte</u>	An Object that maintains an 8-bit (1-byte) variable.
	<u>oEEProm</u>	An Object that provides access to the non-volatile EEPROM memory.
4	<u>oNibble</u>	An Object that maintains a 4-bit variable.
	<u>oRAM</u>	An Object that provides access to the system's Random-Access-Memory.
16	<u>oWord</u>	An Object that maintains a 16-bit (2-byte) variable.



System Objects.

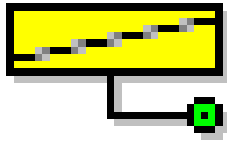
System Objects are Objects that encapsulate the functions of the OOPic's internal system.

One System Object is provided that controls the internal workings of the OOPic. The OOPic Object is pre-created at the time that the OOPic chip is powered on.

	<u>Object</u>	<u>Description</u>
	<u>OOPic</u>	An Object that provides control of and maintains information about the OOPic.

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oA2D Object

An Object that provides a numerical measurement of a voltage.

Description: A Hardware Object that takes an analog voltage presented to one of the OOPic's I/O lines and converts it to a digital value.

Operation: An **oA2D** Object takes the voltage present on the I/O Line specified by the **IOLine** property and compares it to a reference voltage. It then generates a digital value which represents what percentage the voltage is of the reference voltage and updates its **Value** property with the generated value.

The **IOLine** property specifies which one of four analog enabled Input Lines on the OOPic to use. If the **IOLine** property is set to 0, then I/O Line 4 is used for the analog input.

An **Operate** property is provided to activate the **oA2D** Object. When set to 1, the **oA2D** Object continuously performs an analog-to-digital conversion of the voltage present on the I/O Line specified by the **IOLine** property and updates its **Value** property with the result. When the **Operate** property is cleared to 0, the **oA2D** Object enters a dormant state, thus leaving the **Value** property at the value of the last conversion.

The **ExtVRef** property of the **OOPic** Object determines the analog reference voltage, which is selectable to either +5 volts or the voltage level supplied on the OOPic's I/O line 4. If the **ExtVRef** property is set to 0, then +5 volts is used as the reference voltage. If the **ExtVRef** property is set to 1, then the voltage applied to the OOPic's I/O Line 4 is used as the maximum voltage in the voltage conversion range. **The maximum voltage allowed on I/O Line 4 is 14-Volts DC.**

Each dimensioned instance of an **oA2D** Object shares the analog-to-digital converter (A/D) module which converts the analog input signals to a corresponding 8-bit digital number. The A/D module has four analog inputs available on the OOPic's I/O lines 1 - 4, which are multiplexed into a single sample and hold register. The sample and hold register is the input into the A/D module, which generates the result via successive approximation.

The analog-to-digital conversion process takes approximately 14 μ s. Each dimensioned instance of an **oA2D** Object with an **Operate** property set to 1, will, in turn, request an analog-to-digital conversion be done on the voltage present on the I/O line specified in the **oA2D** Object's **IOLine** property. When the conversion is completed, the resulting value will be placed in the Objects **Value** property and the next **A2D** Object instance will follow the same until all active **A2D** Object instances have been addressed. After the last **oA2D** Object instance has been addressed, the process begins again with the first **oA2D** Object instance. With all 4 possible **A2D** Object instances dimensioned and all 4 **Operate** properties set to 1, the entire conversion process takes 56 μ s.

Remarks: When any **oA2D** Object is in use, all four IOLines, 1, 2, 3 and 4 are set to read an analog value and any Digital-I/O Object using these IOLines will read 0.

Size: 3 Bytes

Properties: The following table lists the Properties of the **oA2D** Object

<u>Property</u>	<u>Description</u>						
Address	Returns a pointer to the address of the oA2D Object instance. Data-Type: <i>Number-Pointer, ReadOnly</i> Data-Range: 0 - 127						
IOLine	The physical I/O Line number to use. Data-Type: <i>Nibble</i> Data-Range: 0 - 3 The following table list the values of the IOLine Property.						
	<table border="1"> <thead> <tr> <th><u>IOLine</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>The Analog signal on IOLine 4 is evaluated. 4 can be assigned to the IOLine in place of 0, but it will read back as 0.</td> </tr> </tbody> </table>	<u>IOLine</u>	<u>Constant</u>	<u>Description</u>	0		The Analog signal on IOLine 4 is evaluated. 4 can be assigned to the IOLine in place of 0, but it will read back as 0.
<u>IOLine</u>	<u>Constant</u>	<u>Description</u>					
0		The Analog signal on IOLine 4 is evaluated. 4 can be assigned to the IOLine in place of 0, but it will read back as 0.					

	1 - 3	The Analog signal on the specified IOLine is evaluated.									
MSB	<p>The Most-Significant-Bit of the Value property.</p> <p>Data-Type: <i>Bit, Flag, ReadOnly</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the MSB Property.</p> <table border="1" data-bbox="544 555 1437 824"> <thead> <tr> <th><u>MSB</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>The input voltage is less than 1/2 the reference voltage.</td> </tr> <tr> <td>1</td> <td></td> <td>The input voltage is more than or equal to 1/2 the reference voltage.</td> </tr> </tbody> </table>		<u>MSB</u>	<u>Constant</u>	<u>Description</u>	0		The input voltage is less than 1/2 the reference voltage.	1		The input voltage is more than or equal to 1/2 the reference voltage.
<u>MSB</u>	<u>Constant</u>	<u>Description</u>									
0		The input voltage is less than 1/2 the reference voltage.									
1		The input voltage is more than or equal to 1/2 the reference voltage.									
Operate	<p>A value that specifies whether or not the conversion is performed.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p> <table border="1" data-bbox="544 1229 1437 1413"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The Value property is not updated.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The Value property is updated.</td> </tr> </tbody> </table>		<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The Value property is not updated.	1	cvTrue	The Value property is updated.
<u>Operate</u>	<u>Constant</u>	<u>Description</u>									
0	cvFalse	The Value property is not updated.									
1	cvTrue	The Value property is updated.									
String	<p>The Value property represented as a string.</p> <p>Note: The String property is available in OOPic Basic Version 2</p> <p>Data-Type: <i>String, ReadOnly</i></p>										
Value	<p>The value returned by the Analog-To-Digital conversion for the specified I/O Line.</p> <p>Data-Type: <i>Byte, ReadOnly, Default</i></p> <p>Data-Range: 0 - 255</p> <p>The following table list the values of the Value Property.</p>										

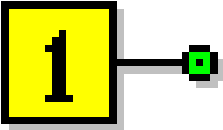
<u>Value</u>	<u>Constant</u>	<u>Description</u>
0		The input voltage is 0 volts.
1 - 254		The input voltage is a percentage of the reference voltage.
255		The input voltage = the reference voltage.

Caution:

The maximum voltage allowed on I/O Line 4 is 14-Volts DC.

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oDIO1 Object

An Object that provides a 1 bit digital I/O.

Description: A Hardware Object that consists of one of the OOPic's I/O Lines.

Operation: An **oDio1** Object's [Value](#) property represents the electrical state of the I/O line specified by the [IOLine](#) property. The [Direction](#) property specifies if the I/O Line is an input or an output.

When the [IOLine](#) property is set, the [Direction](#) property is updated to reflect the current state of the specified I/O Line.

Remarks: Thirty-one physical 1-bit I/O Lines are implemented within the OOPic.

Size: 1 Byte

Properties: The following table lists the Properties of the **oDIO1** Object

<u>Property</u>	<u>Description</u>
-----------------	--------------------

Address	<p>Returns a pointer to the address of the oDIO1 Object instance.</p> <p>Data-Type: <i>Number-Pointer, ReadOnly</i></p> <p>Data-Range: 0 - 127</p>									
Direction	<p>Determines if the I/O Line is an input or an output.</p> <p>Data-Type: <i>Bit</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Direction Property.</p> <table border="1" data-bbox="560 748 1437 1111"> <thead> <tr> <th><u>Direction</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvOutput</td> <td>The specified I/O Line is an output and the Value property is written to it.</td> </tr> <tr> <td>1</td> <td>cvInput</td> <td>The specified I/O Line is High-Z and the Value property is read from it.</td> </tr> </tbody> </table>	<u>Direction</u>	<u>Constant</u>	<u>Description</u>	0	cvOutput	The specified I/O Line is an output and the Value property is written to it.	1	cvInput	The specified I/O Line is High-Z and the Value property is read from it.
<u>Direction</u>	<u>Constant</u>	<u>Description</u>								
0	cvOutput	The specified I/O Line is an output and the Value property is written to it.								
1	cvInput	The specified I/O Line is High-Z and the Value property is read from it.								
IOLine	<p>The physical I/O Line to use.</p> <p>Data-Type: <i>Byte</i></p> <p>Data-Range: 0 - 31</p> <p>The following table list the values of the IOLine Property.</p> <table border="1" data-bbox="560 1469 1437 1742"> <thead> <tr> <th><u>IOLine</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvOff</td> <td>The instance of the DIO Object is inactive.</td> </tr> <tr> <td>1 - 31</td> <td></td> <td>The oDio1 Object uses the pin specified for its I/O</td> </tr> </tbody> </table>	<u>IOLine</u>	<u>Constant</u>	<u>Description</u>	0	cvOff	The instance of the DIO Object is inactive.	1 - 31		The oDio1 Object uses the pin specified for its I/O
<u>IOLine</u>	<u>Constant</u>	<u>Description</u>								
0	cvOff	The instance of the DIO Object is inactive.								
1 - 31		The oDio1 Object uses the pin specified for its I/O								
NonZero	<p>Set to 1 when the Value property is not zero and cleared to 0 when the Value property is zero.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p>									

The following table list the values of the **NonZero** Property.

<u>NonZero</u>	<u>Constant</u>	<u>Description</u>
0	cvFalse	The Value property is zero.
1	cvTrue	The Value property is not zero.

String

The [Value](#) property represented as a string.

Note: The [String](#) property is available in OOPic Basic Version 2

Data-Type: *String, Flag*

Value

The electrical connection to the I/O Line.

Data-Type: *Bit, Flag, Default*

Data-Range: 0 - 1

The following table list the values of the **Value** Property.

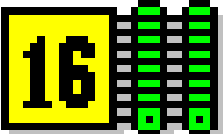
<u>Value</u>	<u>Constant</u>	<u>Description</u>
0	cvOff	The specified I/O line is at 0 Volts.
0 (Logic)	cvLow	The specified I/O line is at 0 Volts.
0 (Voltage)	cvPressed	A Switch connected to the oDio1 Object instance is pressed, connecting the I/O line to ground.
1	cvOn	The specified I/O line is at +5 Volts.
1 (Logic)	cvHigh	The specified I/O line is at +5 Volts.
1 (Voltage)	cvUnpressed	A Switch connected to the oDio1 Object instance is not pressed, leaving the I/O line at +5 Volts.

Methods: The following table lists the Methods of the **oDIO1** Object

<u>Methods</u>	<u>Description</u>
Clear	Clears the Value property to 0.
Invert	Inverts the bits in the Value property
Set	Sets the Value property to 1.

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oDIO16 Object

An Object that provides a 16-bit digital I/O.

Description: A Hardware Object that consists of 16 of the OOPic's I/O lines.

Operation: An **oDIO16** Object's [Value](#) property represents the binary value of the electrical state of the I/O lines 8 - 15 and 24 - 31. The [Direction](#) property specifies if the I/O Lines are inputs or outputs.

When the [IOGroup](#) property is set, the [Direction](#) property is updated to reflect the current state of the specified I/O Lines.

Remarks: 1 Physical 16-bit I/O group is implemented within the OOPic.

Size: 1 Byte

Properties: The following table lists the Properties of the **oDIO16** Object

<u>Property</u>	<u>Description</u>
Address	Returns a pointer to the address of the oDIO16 Object instance. Data-Type: <i>Number-Pointer, ReadOnly</i>

	Data-Range: 0 - 127									
Direction	Determines if the I/O Lines are inputs or outputs. Data-Type: <i>Byte</i> Data-Range: 0 - 1									
IOGroup	Data-Type: <i>Bit, Flag</i> Data-Range: 0 - 1 The following table list the values of the IOGroup Property. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;"><u>IOGroup</u></th> <th style="text-align: left;"><u>Constant</u></th> <th style="text-align: left;"><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	<u>IOGroup</u>	<u>Constant</u>	<u>Description</u>	0			1		
<u>IOGroup</u>	<u>Constant</u>	<u>Description</u>								
0										
1										
String	The Value property represented as a string. Note: The String property is available in OOPic Basic Version 2 Data-Type: <i>String</i>									
Value	The electrical connection to the I/O Lines. Data-Type: <i>Byte, Default</i> Data-Range: 0 - 65535									

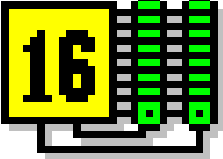
Methods:

The following table lists the Methods of the **oDIO16** Object

<u>Methods</u>	<u>Description</u>
Clear	Clears the Value property to 0.
Dec	Decrements the Value property by 1.
Inc	Increments the Value property by 1.
Invert	Inverts the bits in the Value property
LShift	Shifts the bits in the Value property left.
RShift	Shifts the bits in the Value property Right.
Set	Sets the Value property to 65535.

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•

	<h2 style="margin: 0;">oDIO16x Object</h2> <p style="margin: 0;">An Object that provides a 16-bit digital I/O.</p>
---	--

Description: A Hardware Object that provides expanded I/O by multiplexing IOLines 8 - 15 by IOLines 5 - 7.

Operation: An **oDIO16** Object expands the available I/O lines by multiplexing IOLines 8 - 15 with IOLines 5 - 7.

Remarks: 1 Physical 16-bit I/O expander is implemented within the OOPic.

Size: 5 Bytes

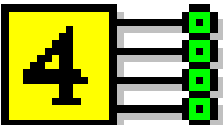
Properties: The following table lists the Properties of the **oDIO16x** Object

<u>Property</u>	<u>Description</u>
Address	<p>Returns a pointer to the address of the oDIO16x Object instance.</p> <p>Data-Type: <i>Number-Pointer, ReadOnly</i></p> <p>Data-Range: 0 - 127</p>
Operate	<p>A value that specifies whether or not the oDIO16x Object scans the multiplexed I/O lines.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p>

	<u>Operate</u>	<u>Constant</u>	<u>Description</u>
	0	cvFalse	The Value property is not updated/output.
	1	cvTrue	The Value property is updated/output.
String	The Value property represented as a string. Note: The String property is available in OOPic Basic Version 2 Data-Type: <i>String</i>		
Value	The electrical connection to the I/O Lines. Data-Type: <i>Byte, Default</i> Data-Range: 0 - 65535		

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oDIO4 Object

An Object that provides a 4-bit digital I/O.

Description: A Hardware Object that consists of 4 of the OOPic's I/O Lines.

Operation: An oDIO4 Object's [Value](#) property represents the binary value of the electrical state of the I/O lines specified by the [IOGroup](#) & the [Nibble](#) properties. The I/O groups are 4 contiguous I/O Lines beginning at I/O line 8, 12, 16, 20, 24 or 28. The [Direction](#) property specifies if the I/O Lines are inputs or outputs.

When either the [IOGroup](#) property or the [Nibble](#) property is set, the [Direction](#) property is updated to reflect the current state of the specified I/O Lines.

Remarks: Six physical 4-bit I/O Groups are implemented within the OOPic.

Size: 1 Byte

Properties: The following table lists the Properties of the **oDIO4** Object

<u>Property</u>	<u>Description</u>									
Address	<p>Returns a pointer to the address of the oDIO4 Object instance.</p> <p>Data-Type: <i>Number-Pointer, ReadOnly</i></p> <p>Data-Range: 0 - 127</p>									
Direction	<p>Determines if the I/O Lines are inputs or outputs.</p> <p>Data-Type: <i>Bit</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Direction Property.</p> <table border="1"> <thead> <tr> <th><u>Direction</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>The specified I/O Lines are outputs and the Value property is written to them.</td> </tr> <tr> <td>1</td> <td></td> <td>The specified I/O Lines are High-Z and the Value property is read from them.</td> </tr> </tbody> </table>	<u>Direction</u>	<u>Constant</u>	<u>Description</u>	0		The specified I/O Lines are outputs and the Value property is written to them.	1		The specified I/O Lines are High-Z and the Value property is read from them.
<u>Direction</u>	<u>Constant</u>	<u>Description</u>								
0		The specified I/O Lines are outputs and the Value property is written to them.								
1		The specified I/O Lines are High-Z and the Value property is read from them.								
IOGroup	<p>The physical I/O Group to use.</p> <p>Data-Type: <i>Nibble</i></p> <p>Data-Range: 0 - 3</p> <p>The following table list the values of the IOGroup Property.</p> <table border="1"> <thead> <tr> <th><u>IOGroup</u></th> <th><u>Nibble</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td></td> <td>The instance of the oDIO8 Object is inactive.</td> </tr> </tbody> </table>	<u>IOGroup</u>	<u>Nibble</u>	<u>Constant</u>	<u>Description</u>	0	0		The instance of the oDIO8 Object is inactive.	
<u>IOGroup</u>	<u>Nibble</u>	<u>Constant</u>	<u>Description</u>							
0	0		The instance of the oDIO8 Object is inactive.							

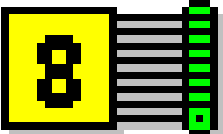
	<table border="1"> <tr><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td></tr> <tr><td>1</td><td>1</td></tr> <tr><td>2</td><td>0</td></tr> <tr><td>2</td><td>1</td></tr> <tr><td>3</td><td>0</td></tr> <tr><td>3</td><td>1</td></tr> </table>	0	1	1	0	1	1	2	0	2	1	3	0	3	1	<p>The instance of the oDIO8 Object is inactive.</p> <p>The oDIO4 Object uses pins 8 - 11 for its I/O.</p> <p>The oDIO4 Object uses pins 12 - 15 for its I/O.</p> <p>The oDIO4 Object uses pins 16 - 19 for its I/O.</p> <p>The oDIO4 Object uses pins 20 - 23 for its I/O.</p> <p>The oDIO4 Object uses pins 24 - 27 for its I/O.</p> <p>The oDIO4 Object uses pins 28 - 31 for its I/O.</p>
0	1															
1	0															
1	1															
2	0															
2	1															
3	0															
3	1															
Nibble	<p>Specifies which side of an IOGroup to use.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Nibble Property.</p> <table border="1"> <thead> <tr> <th><u>Nibble</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvLow</td> <td>The oDio4 Object uses the lower 4 bits of the specified IOGroup</td> </tr> <tr> <td>1</td> <td>cvHigh</td> <td>The oDio4 Object uses the upper 4 bits of the specified IOGroup</td> </tr> </tbody> </table>		<u>Nibble</u>	<u>Constant</u>	<u>Description</u>	0	cvLow	The oDio4 Object uses the lower 4 bits of the specified IOGroup	1	cvHigh	The oDio4 Object uses the upper 4 bits of the specified IOGroup					
<u>Nibble</u>	<u>Constant</u>	<u>Description</u>														
0	cvLow	The oDio4 Object uses the lower 4 bits of the specified IOGroup														
1	cvHigh	The oDio4 Object uses the upper 4 bits of the specified IOGroup														
String	<p>The Value property represented as a string.</p> <p>Note: The String property is available in OOPic Basic Version 2</p> <p>Data-Type: <i>String</i></p>															
Value	<p>The electrical connection to the I/O Lines.</p> <p>Data-Type: <i>Nibble, Default</i></p> <p>Data-Range: 0 - 15</p>															

Methods: The following table lists the Methods of the **oDIO4** Object

Methods	Description
Clear	Clears the Value property to 0.
Dec	Decrements the Value property by 1.
Inc	Increments the Value property by 1.
Invert	Inverts the bits in the Value property
LShift	Shifts the bits in the Value property left.
RShift	Shifts the bits in the Value property Right.
Set	Sets the Value property to 15.

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oDIO8 Object

An Object that provides an 8-bit digital I/O.

Description: A Hardware Object that consists of 8 of the OOPic's I/O Lines.

Operation: An **oDIO8** Object's [Value](#) property represents the binary value of the electrical state of the I/O lines specified by the [IOGroup](#) property. The I/O groups are 8 contiguous I/O Lines beginning at I/O line 8, 16 or 24. The [Direction](#) property specifies if the I/O Lines are inputs or outputs.

When the [IOGroup](#) property is set, the [Direction](#) property is updated to reflect the current state of the specified I/O Lines.

Remarks: Three physical 8-bit I/O Groups are implemented within the OOPic.

Size: 1 Byte

Properties: The following table lists the Properties of the **oDIO8** Object

<u>Property</u>	<u>Description</u>												
Address	<p>Returns a pointer to the address of the oDIO8 Object instance.</p> <p>Data-Type: <i>Number-Pointer, ReadOnly</i></p> <p>Data-Range: 0 - 127</p>												
Direction	<p>Determines if the I/O Lines are inputs or outputs.</p> <p>Data-Type: <i>Bit</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Direction Property.</p> <table border="1"> <thead> <tr> <th><u>Direction</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvOutput</td> <td>The specified I/O Lines are outputs and the Value property is written to them.</td> </tr> <tr> <td>1</td> <td>cvInput</td> <td>The specified I/O Lines are High-Z and the Value property is read from them.</td> </tr> </tbody> </table>	<u>Direction</u>	<u>Constant</u>	<u>Description</u>	0	cvOutput	The specified I/O Lines are outputs and the Value property is written to them.	1	cvInput	The specified I/O Lines are High-Z and the Value property is read from them.			
<u>Direction</u>	<u>Constant</u>	<u>Description</u>											
0	cvOutput	The specified I/O Lines are outputs and the Value property is written to them.											
1	cvInput	The specified I/O Lines are High-Z and the Value property is read from them.											
IOGroup	<p>The physical I/O Group to use.</p> <p>Data-Type: <i>Nibble</i></p> <p>Data-Range: 0 - 3</p> <p>The following table list the values of the IOGroup Property.</p> <table border="1"> <thead> <tr> <th><u>IOGroup</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvOff</td> <td>The instance of the oDIO8 Object is inactive.</td> </tr> <tr> <td>1</td> <td></td> <td>The oDIO8 Object uses pins 8 - 15 for its I/O.</td> </tr> <tr> <td>2</td> <td></td> <td>The oDIO8 Object uses pins 16 - 23 for its I/O.</td> </tr> </tbody> </table>	<u>IOGroup</u>	<u>Constant</u>	<u>Description</u>	0	cvOff	The instance of the oDIO8 Object is inactive.	1		The oDIO8 Object uses pins 8 - 15 for its I/O.	2		The oDIO8 Object uses pins 16 - 23 for its I/O.
<u>IOGroup</u>	<u>Constant</u>	<u>Description</u>											
0	cvOff	The instance of the oDIO8 Object is inactive.											
1		The oDIO8 Object uses pins 8 - 15 for its I/O.											
2		The oDIO8 Object uses pins 16 - 23 for its I/O.											

	3	The oDIO8 Object uses pins 24 - 31 for its I/O
String	The Value property represented as a string. Note: The String property is available in OOPic Basic Version 2 Data-Type: <i>String</i>	
Value	The electrical connection to the I/O Lines. Data-Type: <i>Byte, Default</i> Data-Range: 0 - 255	

Methods: The following table lists the Methods of the **oDIO8** Object

Methods	Description
Clear	Clears the Value property to 0.
Dec	Decrements the Value property by 1.
Inc	Increments the Value property by 1.
Invert	Inverts the bits in the Value property
LShift	Shifts the bits in the Value property left.
RShift	Shifts the bits in the Value property Right.
Set	Sets the Value property to 255.

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oI2C Object

An Object that provides access to an I2C device.

Description: An Object that provides access to an I2C device.

Operation: The **oI2C** Object reads/writes a byte from/to an I2C device connected to the Local I2C Bus. The Local I2C Bus consists of the Local I2C Clock Line, the Local I2C Data Line and Ground.

The **Node** property specifies the 7-Bit address of the I2C device to communicate with. The 7-Bit address is the I2C address that is assigned by the device's manufacturer and is 'hard-wired' into the I2C device.

Three different I2C addressing modes are supported as well as either an 8-Bit or 16-Bit data transfer. If the I2C device does not use an internal memory/register location, then the **Mode** property can be set to 2 (cv7bit) which specifies to send only the 7-Bit I2C address. If the I2C device does use an internal memory/register location, then the **Mode** property can be set to 0 (cv23bit), or 1 (cv10bit) which specifies to send the 7-Bit I2C address followed by a memory/register location of either 16-Bits or 8-Bits respectively.

The **Location** property specifies the I2C device's memory/register location and is only used when the **Mode** property is set to 0 or 1.

The **NoInc** property specifies if the **Location** property is increased each time the I2C device is read from or written to. If the **NoInc** property is set to 0, then the **Location** property will be increased by the number of bytes specified by the **Width** property. If the **NoInc** property is set to 1, then the **Location** property is left unchanged.

The **Width** property specifies how many bytes get transferred. If the **Width** property is set to 0, then 1 byte is transferred and if the **Width** property is set to 1, then 2 bytes are transferred when the I2C device is read/written.

The **oI2C** Object has no **Operate** property. Instead, when an **oI2C** Object's **Value** property is read from or written to, the **oI2C** Object performs all the necessary operations to transfer data to/from the specified I2C device. Once the data has been transferred, the **oI2C** Object returns to a dormant state.

For more information on the specifications of individual I2C devices, consult the manufactures technical information sheets.

Remarks: The default property of the **oI2C** Object cannot be assigned to a pointer.

Size: 5 Bytes

Properties: The following table lists the Properties of the **oI2C** Object

<u>Property</u>	<u>Description</u>															
Address	Returns a pointer to the address of the oI2C Object instance. Data-Type: <i>Number-Pointer, ReadOnly</i> Data-Range: 0 - 127															
Location	A value indicating the current Location-Address to use when accessing the I2C device. Data-Type: <i>Word</i> Data-Range: 0 - 2047															
Mode	The I2C mode Data-Type: <i>Nibble</i> Data-Range: 0 - 3 The following table list the values of the Mode Property. <table border="1" data-bbox="550 1344 1436 1780"> <thead> <tr> <th><u>Mode</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cv23Bit</td> <td>The oI2C Object will use the 23-Bit I2C address mode.</td> </tr> <tr> <td>1</td> <td>cv10Bit</td> <td>The oI2C Object will use the 10-Bit I2C address mode.</td> </tr> <tr> <td>2</td> <td>cv7Bit</td> <td>The oI2C Object will use the 7-Bit I2C address mode.</td> </tr> <tr> <td>3</td> <td></td> <td>Unused</td> </tr> </tbody> </table>	<u>Mode</u>	<u>Constant</u>	<u>Description</u>	0	cv23Bit	The oI2C Object will use the 23-Bit I2C address mode.	1	cv10Bit	The oI2C Object will use the 10-Bit I2C address mode.	2	cv7Bit	The oI2C Object will use the 7-Bit I2C address mode.	3		Unused
<u>Mode</u>	<u>Constant</u>	<u>Description</u>														
0	cv23Bit	The oI2C Object will use the 23-Bit I2C address mode.														
1	cv10Bit	The oI2C Object will use the 10-Bit I2C address mode.														
2	cv7Bit	The oI2C Object will use the 7-Bit I2C address mode.														
3		Unused														
Node	The I2C address to be used. Data-Type: <i>Byte</i> Data-Range: 0 - 127															

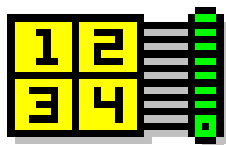
<p>NoInc</p>	<p>A value that specifies if the Location property will be incremented each time the I2C Object's Value property is read or written.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the NoInc Property.</p> <table border="1" data-bbox="555 521 1442 891"> <thead> <tr> <th><u>NoInc</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The oI2C Object does not increment the Location property each time the Value property is read or written</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The oI2C Object increments the Location property each time the Value property is read or written</td> </tr> </tbody> </table>	<u>NoInc</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The oI2C Object does not increment the Location property each time the Value property is read or written	1	cvTrue	The oI2C Object increments the Location property each time the Value property is read or written
<u>NoInc</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	The oI2C Object does not increment the Location property each time the Value property is read or written								
1	cvTrue	The oI2C Object increments the Location property each time the Value property is read or written								
<p>String</p>	<p>The Value property represented as a string.</p> <p>Note: The String property is available in OOPic Basic Version 2</p> <p>Data-Type: <i>String</i></p>									
<p>Value</p>	<p>A value indicating the contents of the I2C device. When read, the value within the I2C device will be returned. When written, the value is stored into the I2C device.</p> <p>Data-Type: <i>Byte, Default</i></p> <p>Data-Range: 0 - 255</p>									
<p>Width</p>	<p>The number of Bytes to read/write</p> <p>Data-Type: <i>Bit</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Width Property.</p> <table border="1" data-bbox="555 1843 1442 2004"> <thead> <tr> <th><u>Width</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cv8Bit</td> <td>The oI2C Object will read and write 8 bits at a time</td> </tr> </tbody> </table>	<u>Width</u>	<u>Constant</u>	<u>Description</u>	0	cv8Bit	The oI2C Object will read and write 8 bits at a time			
<u>Width</u>	<u>Constant</u>	<u>Description</u>								
0	cv8Bit	The oI2C Object will read and write 8 bits at a time								

1	cv16Bit	The oI2C Object will read and write 16 bits at a time
---	----------------	--

See Also: **oEEProm** Object

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oKeypad Object

An Object that reads a 4 x 4 Keypad matrix.

Description: A Hardware Object that scans a 4 row by 4 column set of switches and determines which switch is pressed.

Operation: An **oKeypad** Object splits I/O Group 1 (I/O lines 8 - 15) into 2 sets of 4 I/O lines. The least significant 4 I/O Lines are used for the keypad's 4 columns and the most significant 4 I/O Lines are used for the keypad's 4 rows. The 4 row lines are individually and sequentially set low (0-Volts) while the 4 column lines are used to read which switch is pressed within that row. A switch-press is detected when any one of the 4 column lines is shorted to any one of the row lines. The **oKeypad** Object handles all the required I/O logic an the keypad switch matrix requires no isolation diodes or pull up resistors for operation.

When the [Operate](#) property is set to 1, the internal pull-up resistors on I/O lines 8 - 15 are activated and the keypad switch matrix is scanned for a switch-press, If any switch is pressed, the [Value](#) property is updated with the value of the switch and the [Received](#) property is set to 1. The value of the switch is calculated by $((\text{Row} - 1) * 4) + (\text{Column} - 1)$. If two switches are presses simultaneously, the switch with the highest value will be used. The [Received](#) property will remain at 1 so long as at least 1 switch is depressed. If a second key is pressed before the originally pressed key is released, the [Value](#) and [Received](#) property will remain unchanged from the

values set at the time of the first keys depression. Once all keys are released, the [Received](#) property will be cleared to 0.

When the [Operate](#) property is set to 0, the switch matrix scanning is suspended until the [Operate](#) property is set back to 1.

Size: 4 Bytes

Properties: The following table lists the Properties of the **oKeypad** Object

<u>Property</u>	<u>Description</u>									
Address	<p>Returns a pointer to the address of the oKeypad Object instance.</p> <p>Data-Type: <i>Number-Pointer, ReadOnly</i></p> <p>Data-Range: 0 - 127</p>									
Operate	<p>A value that specifies whether or not the Object is operating and updating its properties.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p> <table border="1"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The Value property is not updated.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The Value property is updated.</td> </tr> </tbody> </table>	<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The Value property is not updated.	1	cvTrue	The Value property is updated.
<u>Operate</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	The Value property is not updated.								
1	cvTrue	The Value property is updated.								
Received	<p>A value that indicates that a new key-value has been received and that the Value property was updated with new information.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Received Property.</p> <table border="1"> <thead> <tr> <th><u>Received</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> </tbody> </table>	<u>Received</u>	<u>Constant</u>	<u>Description</u>						
<u>Received</u>	<u>Constant</u>	<u>Description</u>								

	0	cvFalse	The Value property has not been updated with new data.
	1	cvTrue	The Value property has been updated with new data.
String	<p>The Value property represented as a string.</p> <p>Note: The String property is available in OOPic Basic Version 2</p> <p>Data-Type: <i>String, ReadOnly</i></p>		
Value	<p>A value that is set to the switch number of the last switch that was pressed.</p> <p>Data-Type: <i>Nibble, ReadOnly, Default</i></p> <p>Data-Range: 0 - 15</p>		

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oPWM Object

An Object that provides a Pulse-Width-Modulated output.

Description: A Hardware Object that provides a clocked output with a variable sized pulse width.

Operation: An **oPWM** Object outputs a Pulse-Width-Modulated clock cycle on the I/O line specified by the [IOLine](#) property.

The combination of the [Period](#) and the [PreScale](#) properties specify the PWM clock cycle's frequency. The [PreScale](#) property specifies how many times a 5-Mhz clock is divided. This scaled down 5-Mhz clock is used to increment a Period-Duration Counter. Once the Period-Duration Counter reaches the value specified by the [Period](#)

property, a full PWM clock cycle has been reached. The [PreScale](#) property can be set to divide the 5-Mhz clock by 1, 4 or 16 giving frequencies of 5-Mhz, 1.25-Mhz and 312.5-Khz and the [Period](#) property can be set to any value from 1 to 255, thus giving 765 possible frequencies.

The Pulse-Width of the logic-high portion of the cycle is determined by the [Value](#) property. At the beginning of each PWM clock cycle, the I/O Line specified by the IOLine property is set to 5-Volts. Then, each time the Period-Duration Counter is incremented, it is compared to the value specified by the [Value](#) property and once the Period-Duration Counter reaches the value specified by the [Value](#) property, the I/O Line is set to 0-Volts.

The [Period](#) property also dictates the resolution of the PWM pulse. If the [Period](#) property is set to 255 then the [Value](#) property's range is 0 - 255. Likewise, if the [Period](#) property is set to 10 then the [Value](#) property's effective range is 0 - 10 because any value above 10 will never be reached by the Period-Duration Counter.

The [Operate](#) property specifies if the High going PWM pulse is generated. When the [Operate](#) property is set to 1, the **oPWM** Object starts generating PWM pulses. When the [Operate](#) property is set to 0, the PWM pulses are discontinued and the specified I/O Line is set to 0-Volts. If a current PWM pulse was in progress, it is cut short and the I/O line is immediately set to 0-Volts.

Remarks: Two physical Pulse-Width-Modulators are implemented within the OOPic.

Size: 3 Bytes

Properties: The following table lists the Properties of the **oPWM** Object

<u>Property</u>	<u>Description</u>
Address	Returns a pointer to the address of the oPWM Object instance. Data-Type: <i>Number-Pointer, ReadOnly</i> Data-Range: 0 - 127
IOLine	The I/O Line of one of the two PWM hardware modules.

Data-Type: *Bit*

Data-Range: 0 - 1

The following table list the values of the **IOLine** Property.

<u>IOLine</u>	<u>Constant</u>	<u>Description</u>
0		The PWM signal is outputted on IOLine 18. 18 can be assigned to the IOLine in place of 0, but it will read back as 0.
1		The PWM signal is outputted on IOLine 17. 17 can be assigned to the IOLine in place of 1, but it will read back as 1.

Operate

A value that specifies whether or not the **oPWM** Object outputs the duty cycle.

Data-Type: *Bit, Flag*

Data-Range: 0 - 1

The following table list the values of the **Operate** Property.

<u>Operate</u>	<u>Constant</u>	<u>Description</u>
0	cvFalse	The PWM signal remains Low.
1	cvTrue	The PWM signal outputs a high pulse with a Duty Cycle determined by the Value property and the Period property.

Period

A value that specifies the frequency of the PWM cycle.

Data-Type: *Byte*

Data-Range: 0 - 255

PreScale

A value that specifies the base frequency of the [Period](#) property.

Data-Type: *Nibble*

	<p>Data-Range: 0 - 3</p> <p>The following table list the values of the PreScale Property.</p> <table border="1"> <thead> <tr> <th><u>PreScale</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>The base frequency is 5-Mhz</td> </tr> <tr> <td>1</td> <td></td> <td>The base frequency of 5-Mhz is divided by 4 to get 1.25Mhz</td> </tr> <tr> <td>2</td> <td></td> <td>The base frequency of 5-Mhz is divided by 16 to get 312,500hz</td> </tr> <tr> <td>3</td> <td></td> <td>The base frequency of 5-Mhz is divided by 16 to get 312,500hz (same as 2)</td> </tr> </tbody> </table>	<u>PreScale</u>	<u>Constant</u>	<u>Description</u>	0		The base frequency is 5-Mhz	1		The base frequency of 5-Mhz is divided by 4 to get 1.25Mhz	2		The base frequency of 5-Mhz is divided by 16 to get 312,500hz	3		The base frequency of 5-Mhz is divided by 16 to get 312,500hz (same as 2)
<u>PreScale</u>	<u>Constant</u>	<u>Description</u>														
0		The base frequency is 5-Mhz														
1		The base frequency of 5-Mhz is divided by 4 to get 1.25Mhz														
2		The base frequency of 5-Mhz is divided by 16 to get 312,500hz														
3		The base frequency of 5-Mhz is divided by 16 to get 312,500hz (same as 2)														
String	<p>The Value property represented as a string.</p> <p>Note: The String property is available in OOPic Basic Version 2</p> <p>Data-Type: <i>String</i></p>															
Value	<p>The duty cycle value for the oPWM Object.</p> <p>Data-Type: <i>Byte, Default</i></p> <p>Data-Range: 0 - 255</p>															

Methods: The following table lists the Methods of the **oPWM** Object

<u>Methods</u>	<u>Description</u>
Clear	Clears the Value property to 0.
Dec	Decrements the Value property by 1.
Inc	Increments the Value property by 1.
LShift	Shifts the bits in the Value property left.
RShift	Shifts the bits in the Value property Right.
Set	Sets the Value property to 255.

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oSerial Object

An Object that provides an asynchronous serial I/O port.

Description: A Hardware Object that uses two I/O lines to communicate serially with other devices.

Operation: An **oSerial** Object transmits and receives serial data at a baud rate specified by the [Baud](#) property.

The [Baud](#) property specifies a baud rate of 31500, 1200, 2400 or 9600 Baud. When a value is written to the [Value](#) property, it is sent serially out I/O Line 22 and when a value is received serially from I/O line 23, it is stored in the [Value](#) property and the [Received](#) property is set to [cvTrue](#). After the [Received](#) property is set to [cvTrue](#), reading the [Value](#) property will clear the [Received](#) property to [cvFalse](#).

The serial input and output signals are TTL level signals providing 0 and 5 volts. Conversion to RS232 signals can be done with a TTL to RS232 signal converter chip such as the SN75188 or the MAX203 which will provide the voltage conversion to +/- 12 Volts as well as providing the required signal inversion.

The **oSerial** Object class defines an Objects that uses 1 Input line and 1 Output line of the OOPic's thirty-one Input/Output Lines and encapsulates the OOPic's serial communication (SC) module.

The serial communication (SC) module provides a high-speed serial communication port capable of communicating at 1200, 2400, 9600 and 31500 BPS. It can be configured as full duplex asynchronous or synchronous and can communicate with serial devices such as CRT terminals and personal computers. It uses the standard nonreturn-to-zero (NRZ) format asynchronous mode, (one start bit, eight data bits and one stop bit). The SC module transmits and receives the lowest significant bit first. The SC module's transmitters

and receiver are functionally independent but use the same data format and baud rate.

Each dimensioned instance of the oSerial Object internally uses this single serial communications module and thus only one instance may be active at any given time. Setting the Baud property to a value other than 0 activates an instance of the Serial Object and specifies which baud rate the SC module is to communicate at. Setting the Baud property on any oSerial Object will deactivate any other oSerial Object that was previously activated.

Remarks: One physical serial port is implemented within the OOPic.

Size: 4 Bytes

Properties: The following table lists the Properties of the **oSerial** Object

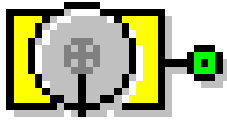
<u>Property</u>	<u>Description</u>															
Address	Returns a pointer to the address of the oSerial Object instance. Data-Type: <i>Number-Pointer, ReadOnly</i> Data-Range: 0 - 127															
Baud	A value that specifies the baud rate of the serial port. Data-Type: <i>Nibble</i> Data-Range: 0 - 3 The following table list the values of the Baud Property.															
	<table border="1"> <thead> <tr> <th><u>Baud</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvMidi</td> <td>The UART communicates at 31250 Baud. (Midi)</td> </tr> <tr> <td>1</td> <td>cv1200</td> <td>The UART communicates at 1200 Baud.</td> </tr> <tr> <td>2</td> <td>cv2400</td> <td>The UART communicates at 2400 Baud.</td> </tr> <tr> <td>3</td> <td>cv9600</td> <td>The UART communicates at 9600 Baud.</td> </tr> </tbody> </table>	<u>Baud</u>	<u>Constant</u>	<u>Description</u>	0	cvMidi	The UART communicates at 31250 Baud. (Midi)	1	cv1200	The UART communicates at 1200 Baud.	2	cv2400	The UART communicates at 2400 Baud.	3	cv9600	The UART communicates at 9600 Baud.
<u>Baud</u>	<u>Constant</u>	<u>Description</u>														
0	cvMidi	The UART communicates at 31250 Baud. (Midi)														
1	cv1200	The UART communicates at 1200 Baud.														
2	cv2400	The UART communicates at 2400 Baud.														
3	cv9600	The UART communicates at 9600 Baud.														

<p>Mode</p>	<p>A value that specifies if the serial port operates synchronous or asynchronous.</p> <p>Data-Type: <i>Bit</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Mode Property.</p> <table border="1" data-bbox="627 519 1437 698"> <thead> <tr> <th><u>Mode</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	<u>Mode</u>	<u>Constant</u>	<u>Description</u>	0			1		
<u>Mode</u>	<u>Constant</u>	<u>Description</u>								
0										
1										
<p>Operate</p>	<p>A value that specifies whether or not the oSerial Object will receive or send data.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p> <table border="1" data-bbox="627 1104 1437 1283"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0			1		
<u>Operate</u>	<u>Constant</u>	<u>Description</u>								
0										
1										
<p>Received</p>	<p>A value specifying if a byte of data has been received. Set when 8 bits of data have been successfully received. Cleared when the Value property is read.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Received Property.</p> <table border="1" data-bbox="627 1780 1437 2004"> <thead> <tr> <th><u>Received</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>No new data has been received.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>New data has been received.</td> </tr> </tbody> </table>	<u>Received</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	No new data has been received.	1	cvTrue	New data has been received.
<u>Received</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	No new data has been received.								
1	cvTrue	New data has been received.								

<p>String</p>	<p>The Value property represented as a string.</p> <p>Note: The String property is available in OOPic Basic Version 2</p> <p>Data-Type: <i>String</i></p>									
<p>Transmitting</p>	<p>A value specifying if the serial port's transmit buffer is full. Set when the Value property is written to. Cleared when the 8 bits of data has begun to transmit. Any write to the Value property when Transmitting is 1, will be ignored.</p> <p>Data-Type: <i>Bit, Flag, ReadOnly</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Transmitting Property.</p> <table border="1" data-bbox="627 931 1439 1205"> <thead> <tr> <th><u>Transmitting</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>No data is currently being sent.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>Data is currently being sent.</td> </tr> </tbody> </table>	<u>Transmitting</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	No data is currently being sent.	1	cvTrue	Data is currently being sent.
<u>Transmitting</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	No data is currently being sent.								
1	cvTrue	Data is currently being sent.								
<p>Value</p>	<p>The value of the received and transmitted data. If written to, the data is sent serially through the UART's transmit line located on I/O line 22. If read, it returns the last data received by the UART through the receive line located on I/O line 23.</p> <p>Data-Type: <i>Byte, Default</i></p> <p>Data-Range: 0 - 255</p>									

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

-



oServo Object

An Object that controls a servo-motor

Description: A Hardware Object that outputs a PWM pulse that is timed for RC servo-motor control.

Operation: An **oServo** Object positions a RC servo motor connected to the I/O Line specified by the [IOLine](#) property to a position specified by the [Value](#), [Center](#) and [InvertOut](#) properties by outputting a PWM servo control pulse.

The [IOLine](#) property specifies which one of the OOPic's 31 digital I/O Lines is to be use. When set to 0, the **oServo** object enters a dormant state. When set to a value other than 0, the **oServo** Object will, when activated, set that I/O line to an Output and use that I/O line for controlling the servo.

An [Operate](#) property is provided to activate the **oServo** Object. When set to 1, the **oServo** Object will continuously output a PWM servo control pulse to the I/O Line specified by the [IOLine](#) property. When the [Operate](#) property is cleared to 0, the **oServo** Object sets the specified I/O line to 0 volts.

The **oServo** Object's PWM servo control pulse is tailored to control a standard RC Servo and is capable of generating a Logical High-Going pulse from 0 to 3ms in duration in 1/36ms increments. A typical servo with a rotational range of 180 degrees, is positioned by the duration of a 5 volt Logical high-going pulse in the range of .61ms to 2.36ms. The duration of the **oServo** Object's PWM servo control pulse is configured by setting the [Value](#), [Center](#) and [InvertOut](#) properties. The [Value](#) property specifies the desired position of the servo represented as a value. The [Center](#) property adjusts the servo's position by offsetting the [Value](#) property and must be adjusted for the mechanical differences of each different servo motor connected to the OOPic. An [InvertOut](#) property is used to reverse the direction that the servo turns in response to the [Value](#) and [Center](#) properties. If the [InvertOut](#) property is set to 0, then the pulse time is calculated as $((\text{Value} + \text{Center}) * (1/36))$ If the [InvertOut](#) property is set to 1, then the pulse time is calculated as $((107 - (\text{Value} + \text{Center})) * (1/36))$. Note that each unit of

magnitude of the control pulse time is equal to 1/36ms

PWM Servo Control Pulse Duration Examples:

In the case of positioning a servo at 0 degrees, set the [Value](#) property to 0 and the [Center](#) property to 22.

$$((0+22) * (1/36)) = .61$$

In the case of positioning a servo at 180 degrees, set the [Value](#) property to 63 and the [Center](#) property to 22.

$$((63+22) * (1/36)) = 2.36$$

In the case of positioning a servo at 180 degrees while turning in reverse, set the [Value](#) property to 0, the [Center](#) property to 22 and the [InvertOut](#) property to 1.

$$((107-(0+22)) * (1/36)) = 2.36$$

Each dimensioned instance of the **oServo** Object shares the Servo Motor Control (SM) module which controls one servo at a time. The SM module provides the PWM servo control pulse that is required to electrically refresh a servo motor's position. The SM control module can output the PWM control pulse on any of the OOPic's 31 I/O Lines.

When the SM control module is activated, it looks for the first active **oServo** Object. (The first **oServo** Object with its [Operate](#) property set to 1). That servo's I/O line is set to 1 and then timed. When the time specified by the **oServo** Object's [Value](#), [Center](#) & [InvertOut](#) properties is met, the I/O line is set to 0 and the SM control module moves to the next active **oServo** Object. The SM control module will systematically continue through each active **oServo** Object until all been completed.

The servo's refresh cycle is a result of the SM control module being activated 30 times each second, which, in turn, outputs the PWM servo control pulse for each active **oServo** Object 30 times per second. If the sum of the PWM pulse time from all the active **oServo** Objects totals more than 16.67ms, then the SM control module is activated 15 times each second which, in turn, refreshes each active **oServo** Object 15 times per second.

Refresh Cycle Examples:

In the case of 7 or fewer servos, the servo refresh cycle will always remain at 30 refreshes per second because the maximum total pulse time will never exceed 16.67ms.

$$(2.36ms [Pulse\ width\ at\ full\ scale] * 7 [number\ of\ servos] = 16.52ms.)$$

However, in the case of 8 or more servos, the servo refresh cycle can change depending on the position of the servos.

With all 8 servos at full scale, the refresh rate is reduced to 15 times per second because the total pulse time exceeds 16.67ms.

$$(2.36ms \text{ [Pulse width at full scale]} * 8 \text{ [number of servos]} = 18.88ms.)$$

With all 8 servos at lowest scale, the refresh cycle remains at 30 times per second because the total pulse time is less than 16.67ms.

$$(0.61ms \text{ [Pulse width at lowest scale]} * 8 \text{ [number of servos]} = 4.88ms.)$$

Remarks: After the [IOLine](#) property is set, the specified IOLine will remain in its former logic state of either High-Z, High or Low, until the operate property is set to 1.

Size: 4 Bytes

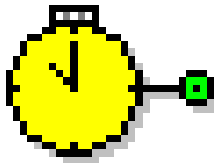
Properties: The following table lists the Properties of the **oServo** Object

<u>Property</u>	<u>Description</u>						
Address	Returns a pointer to the address of the oServo Object instance. Data-Type: Number-Pointer, ReadOnly Data-Range: 0 - 127						
Center	Adjusts the value for the servo's mechanical center Data-Type: Byte Data-Range: 0 - 63						
InvertOut	A value that specifies if the output pulse is reversed. Data-Type: Bit, Flag Data-Range: 0 - 1 The following table list the values of the InvertOut Property.						
<table border="1"> <thead> <tr> <th><u>InvertOut</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The Servo control signal is normal</td> </tr> </tbody> </table>		<u>InvertOut</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The Servo control signal is normal
<u>InvertOut</u>	<u>Constant</u>	<u>Description</u>					
0	cvFalse	The Servo control signal is normal					

	1	cvTrue	The Servo control signal is reversed.									
IOLine	<p>The physical I/O Line to use.</p> <p>Data-Type: <i>Byte</i></p> <p>Data-Range: 0 - 31</p>											
Operate	<p>A value that specifies whether or not the pulse is outputted.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p> <table border="1"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The Servo control signal is not outputted.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The Servo control signal is outputted.</td> </tr> </tbody> </table>			<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The Servo control signal is not outputted.	1	cvTrue	The Servo control signal is outputted.
<u>Operate</u>	<u>Constant</u>	<u>Description</u>										
0	cvFalse	The Servo control signal is not outputted.										
1	cvTrue	The Servo control signal is outputted.										
String	<p>The Value property represented as a string.</p> <p>Note: The String property is available in OOPic Basic Version 2</p> <p>Data-Type: <i>String</i></p>											
Value	<p>The value specifying the position of the servo-motor</p> <p>Data-Type: <i>Byte, Default</i></p> <p>Data-Range: 0 - 127</p>											

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oTimer Object

An Object that provides a 16-bit high speed counter.

Description: A Hardware Object that provides a 16-bit high speed counter with optional Crystal controlled frequency and input scaling.

Operation: The **oTimer** Object counts each cycle of a clock source by incrementing the 16 bit [Value](#) property each time the clock source cycles. An [MSB](#) Flag property reflects the 16th bit of the [Value](#) property and can be used as the input to a **oCounter** Object to extend the count. The [ExtClock](#) property specifies if the programmable-clock counts the cycles of an internal 5Mhz clock or a external clock supplied on I/O line 16. An [ExtXtal](#) property specifies if I/O Line 16 is used in conjunction with I/O Line 17 for a crystal to be connected. A [PreScale](#) property specifies a value which divides the clock source frequency before it is counted.

The [Operate](#) property specifies if the [Value](#) property will increase each time the clock source cycles.

When the [Value](#) property reaches a count of 65535, the next clock source cycle will roll the [Value](#) back to 0.

Remarks: Since only one physical 16-bit timer is implemented within the OOPic, the properties of all instances of the **oTimer** Object will read the same.

Size: 1 Byte

Properties: The following table lists the Properties of the **oTimer** Object

<u>Property</u>	<u>Description</u>
Address	Returns a pointer to the address of the oTimer Object instance. Data-Type: <i>Number-Pointer, ReadOnly</i> Data-Range: 0 - 127

<p>ExtClock</p>	<p>Determines if the oTimer Object uses I/O Lines or an internal clock for its clock source.</p> <p>Data-Type: Bit</p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the ExtClock Property.</p> <table border="1" data-bbox="560 521 1439 880"> <thead> <tr> <th><u>ExtClock</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvOff</td> <td>5Mhz internal clock is used as the input frequency to the oTimer Object.</td> </tr> <tr> <td>1</td> <td>cvOn</td> <td>I/O Lines 16 & 17 are used for the source of the input frequency to the oTimer Object.</td> </tr> </tbody> </table>	<u>ExtClock</u>	<u>Constant</u>	<u>Description</u>	0	cvOff	5Mhz internal clock is used as the input frequency to the oTimer Object.	1	cvOn	I/O Lines 16 & 17 are used for the source of the input frequency to the oTimer Object.
<u>ExtClock</u>	<u>Constant</u>	<u>Description</u>								
0	cvOff	5Mhz internal clock is used as the input frequency to the oTimer Object.								
1	cvOn	I/O Lines 16 & 17 are used for the source of the input frequency to the oTimer Object.								
<p>ExtXtal</p>	<p>A value that specifies if the external crystal driver is enabled on the I/O lines.</p> <p>Data-Type: Bit</p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the ExtXtal Property.</p> <table border="1" data-bbox="560 1290 1439 1648"> <thead> <tr> <th><u>ExtXtal</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvOff</td> <td>The external crystal drive circuitry is disabled. The oTimer Object uses only I/O line 16.</td> </tr> <tr> <td>1</td> <td>cvOn</td> <td>The external crystal drive circuitry is enabled. The oTimer Object uses both I/O Lines 16 & 17.</td> </tr> </tbody> </table>	<u>ExtXtal</u>	<u>Constant</u>	<u>Description</u>	0	cvOff	The external crystal drive circuitry is disabled. The oTimer Object uses only I/O line 16.	1	cvOn	The external crystal drive circuitry is enabled. The oTimer Object uses both I/O Lines 16 & 17.
<u>ExtXtal</u>	<u>Constant</u>	<u>Description</u>								
0	cvOff	The external crystal drive circuitry is disabled. The oTimer Object uses only I/O line 16.								
1	cvOn	The external crystal drive circuitry is enabled. The oTimer Object uses both I/O Lines 16 & 17.								
<p>MSB</p>	<p>The Most-Significant-Bit of the Value property. Cycles each time the oTimer Object counts to 65535.</p> <p>Data-Type: Bit, Flag</p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the MSB Property.</p> <table border="1" data-bbox="560 2011 1439 2063"> <thead> <tr> <th><u>MSB</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> </tbody> </table>	<u>MSB</u>	<u>Constant</u>	<u>Description</u>						
<u>MSB</u>	<u>Constant</u>	<u>Description</u>								

	0	Value property is ≤ 32767															
	1	Value property is ≥ 32768															
Operate	<p>A value that specifies whether or not the oTimer Object counts the cycles of the clock source .</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p> <table border="1"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The Value property is not updated.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The Value property is updated.</td> </tr> </tbody> </table>		<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The Value property is not updated.	1	cvTrue	The Value property is updated.						
<u>Operate</u>	<u>Constant</u>	<u>Description</u>															
0	cvFalse	The Value property is not updated.															
1	cvTrue	The Value property is updated.															
PreScale	<p>A value that specifies what the input clock of the oTimer Object will be divided by.</p> <p>Data-Type: <i>Nibble</i></p> <p>Data-Range: 0 - 3</p> <p>The following table list the values of the PreScale Property.</p> <table border="1"> <thead> <tr> <th><u>PreScale</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>1:1 - The input clock is not changed.</td> </tr> <tr> <td>1</td> <td></td> <td>1:2 - The input clock is divided by 2.</td> </tr> <tr> <td>2</td> <td></td> <td>1:4 - The input clock is divided by 4.</td> </tr> <tr> <td>3</td> <td></td> <td>1:8 - This input clock is divided by 8.</td> </tr> </tbody> </table>		<u>PreScale</u>	<u>Constant</u>	<u>Description</u>	0		1:1 - The input clock is not changed.	1		1:2 - The input clock is divided by 2.	2		1:4 - The input clock is divided by 4.	3		1:8 - This input clock is divided by 8.
<u>PreScale</u>	<u>Constant</u>	<u>Description</u>															
0		1:1 - The input clock is not changed.															
1		1:2 - The input clock is divided by 2.															
2		1:4 - The input clock is divided by 4.															
3		1:8 - This input clock is divided by 8.															
String	<p>The Value property represented as a string.</p> <p>Note: The String property is available in OOPic Basic Version 2</p> <p>Data-Type: <i>String</i></p>																

Value	The current cycle count value. Data-Type: <i>Word, Default</i> Data-Range: 0 - 65535
--------------	--

See Also: **Hz1** property, **hz60** property.

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•

	<h2 style="margin: 0;"><u>oConverter Object</u></h2> <p style="margin: 0;">An Object that provides conversion functions.</p>
--	--

Description: A Processing Object that converts one value to another in one of 4 predefined, commonly used ways. Some of the value conversions emulate the functions of common integrated circuits.

Operation: An **oConverter** Object takes the [Value](#) property from the Object pointed to by [Input](#) property and converts it to another value in a manner specified by the [Mode](#) property then stores that value in the [Value](#) property of the Object pointed to by the [Output](#) property.

The conversions include: binary to decimal, binary to 7-Segment display, binary to Sine and binary to stepper-motor phase.

Size: 3 Bytes

Properties: The following table lists the Properties of the **oConverter** Object

<u>Property</u>	<u>Description</u>
-----------------	--------------------

<p>Address</p>	<p>Returns a pointer to the address of the oConverter Object instance.</p> <p>Data-Type: <i>Number-Pointer, ReadOnly</i></p> <p>Data-Range: 0 - 127</p>									
<p>Blank</p>	<p>A value that specifies if the bits of the converted value are cleared before stored in the Output Object.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Blank Property.</p> <table border="1" data-bbox="576 792 1437 1061"> <thead> <tr> <th><u>Blank</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>The Output Object is updated with the converted information.</td> </tr> <tr> <td>1</td> <td></td> <td>The Output Object's bits are all set to 0.</td> </tr> </tbody> </table>	<u>Blank</u>	<u>Constant</u>	<u>Description</u>	0		The Output Object is updated with the converted information.	1		The Output Object's bits are all set to 0.
<u>Blank</u>	<u>Constant</u>	<u>Description</u>								
0		The Output Object is updated with the converted information.								
1		The Output Object's bits are all set to 0.								
<p>Input</p>	<p>A pointer to an Object that holds the value to be converted.</p> <p>Data-Type: <i>Number-Pointer</i></p> <p>Data-Range: Any Object with a Value property</p>									
<p>InvertOut</p>	<p>A value that specifies if the bits of the converted value are inverted before stored in the Output Object.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the InvertOut Property.</p> <table border="1" data-bbox="576 1742 1437 2056"> <thead> <tr> <th><u>InvertOut</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>The converted value is written to the Output Object.</td> </tr> <tr> <td>1</td> <td></td> <td>The bits of the converted value are inverted before written to the Output Object.</td> </tr> </tbody> </table>	<u>InvertOut</u>	<u>Constant</u>	<u>Description</u>	0		The converted value is written to the Output Object.	1		The bits of the converted value are inverted before written to the Output Object.
<u>InvertOut</u>	<u>Constant</u>	<u>Description</u>								
0		The converted value is written to the Output Object.								
1		The bits of the converted value are inverted before written to the Output Object.								

<p>Mode</p>	<p>A value that specifies which one of 4 conversion types to performed.</p> <p>Data-Type: <i>Nibble</i></p> <p>Data-Range: 0 - 3</p> <p>The following table list the values of the Mode Property.</p> <table border="1" data-bbox="576 517 1437 1167"> <thead> <tr> <th><u>Mode</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cv7Seg</td> <td>Binary to/from 7 Segment display.</td> </tr> <tr> <td>1</td> <td>cvPhase</td> <td>Binary to 8-Phase Stepper-Motor.</td> </tr> <tr> <td>2</td> <td>cvSin</td> <td>Binary to/from Sine. This can also be done in program code with the Bsin Function.</td> </tr> <tr> <td>3</td> <td>cvDecimal</td> <td>Binary to/from Decimal. Same as (74150 IC). I.E. Binary:011 = Descimal:00000100 -and- Binary:111 = Descimal:10000000</td> </tr> </tbody> </table>	<u>Mode</u>	<u>Constant</u>	<u>Description</u>	0	cv7Seg	Binary to/from 7 Segment display.	1	cvPhase	Binary to 8-Phase Stepper-Motor.	2	cvSin	Binary to/from Sine. This can also be done in program code with the Bsin Function.	3	cvDecimal	Binary to/from Decimal. Same as (74150 IC). I.E. Binary:011 = Descimal:00000100 -and- Binary:111 = Descimal:10000000
<u>Mode</u>	<u>Constant</u>	<u>Description</u>														
0	cv7Seg	Binary to/from 7 Segment display.														
1	cvPhase	Binary to 8-Phase Stepper-Motor.														
2	cvSin	Binary to/from Sine. This can also be done in program code with the Bsin Function.														
3	cvDecimal	Binary to/from Decimal. Same as (74150 IC). I.E. Binary:011 = Descimal:00000100 -and- Binary:111 = Descimal:10000000														
<p>Operate</p>	<p>A value that specifies whether or not the values are updated.</p> <p>Data-Type: <i>Bit, Flag, Default</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p> <table border="1" data-bbox="576 1570 1437 1843"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The Output Objects Value property is not updated.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The Output Objects Value property is updated.</td> </tr> </tbody> </table>	<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The Output Objects Value property is not updated.	1	cvTrue	The Output Objects Value property is updated.						
<u>Operate</u>	<u>Constant</u>	<u>Description</u>														
0	cvFalse	The Output Objects Value property is not updated.														
1	cvTrue	The Output Objects Value property is updated.														
<p>Output</p>	<p>A pointer to an Object whose Value property will be updated with the converted value.</p> <p>Data-Type: <i>Number-Pointer</i></p>															

Data-Range: Any Object with a Value property
--

See Also: **Bsin**

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oCounter Object

An Object that provides counting functions.

Description: A Processing Object that performs counting operations by increasing or decreasing another Object's [Value](#) property for each "Clock-Tick"

Operation: The **oCounter** Object will increment or decrement the [Value](#) of the Object pointed to by the [Output](#) property for each "Clock-Tick".

A [Mode](#) property specifies how the **oCounter** Object determines when a "Clock-Tick" has occurred. If the [Mode](#) property is set to cvCount or 0, each cycle of the Flag property pointed to by the [ClockIn1](#) property is considered to be a "Clock-Tick" and the [Value](#) of the [Output](#) Object is incremented or decremented in the direction specified by the [Direction](#) Property . If the [Mode](#) property is set to cvPhase or 1, the Flags pointed to by the [ClockIn1](#) and [Clockin2](#) properties are Quadrant-Encoded and the [Value](#) of the [Output](#) Object is incremented or decremented in the direction specified by Quadrant change. The [Direction](#) Property is then updated to reflect the direction of the change.

Each time a "Clock-Tick" increments the [Output](#) Object's value, that value is tested against the [Value](#) of the Object pointed to by the [Input](#) property. If it is greater, the [Value](#) of the [Output](#) Object is cleared to 0. Each time a "Clock-Tick" decrements the [Output](#) Object's value, it is tested against 0. If it is lower, the [Value](#) of the [Output](#) Object is set to the [Value](#) of the Object pointed to by the [Input](#) property

Size: 5 Bytes

Properties: The following table lists the Properties of the **oCounter** Object

<u>Property</u>	<u>Description</u>																
Address	Returns a pointer to the address of the oCounter Object instance. Data-Type: <i>Number-Pointer, ReadOnly</i> Data-Range: 0 - 127																
ClockIn1	A pointer to an Object's Flag property whose value is used as a clocking signal to determine when to increment or decrement the Value of the Object pointed to by the Output property. Data-Type: <i>Flag-Pointer</i> Data-Range: Any Object's Flag property																
ClockIn2	A pointer to an Object's Flag property whose value is used as the ClockIn1 's complement when the Mode property is set to cvPhase . Data-Type: <i>Flag-Pointer</i> Data-Range: Any Object's Flag property																
Direction	Specifies the direction of the count. Data-Type: <i>Bit, Flag</i> Data-Range: 0 - 1 The following table list the values of the Direction Property.																
<table border="1"> <thead> <tr> <th><u>Direction</u></th> <th><u>Mode</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>cvPositive</td> <td>Count positively (Increment)</td> </tr> <tr> <td>0</td> <td>1</td> <td>cvPositive</td> <td>Output was incremented</td> </tr> <tr> <td>1</td> <td>0</td> <td>cvNegative</td> <td>Count negatively (Decrement)</td> </tr> </tbody> </table>		<u>Direction</u>	<u>Mode</u>	<u>Constant</u>	<u>Description</u>	0	0	cvPositive	Count positively (Increment)	0	1	cvPositive	Output was incremented	1	0	cvNegative	Count negatively (Decrement)
<u>Direction</u>	<u>Mode</u>	<u>Constant</u>	<u>Description</u>														
0	0	cvPositive	Count positively (Increment)														
0	1	cvPositive	Output was incremented														
1	0	cvNegative	Count negatively (Decrement)														

1	1	cvNegative	Output was decremented
---	---	-------------------	-------------------------------

Input

A pointer to an Object whose **Value** property is used as a count limit value.

Each time a "Clock-Tick" Increments the **Output** Object's **Value**, the new **Value** is tested against the **Input** Object's **Value**. If it is greater, then the **Value** of the **Output** Object is cleared to 0.

Each time a "Clock-Tick" decrements the **Output** Object's **Value**, the new **Value** is tested against 0. If it is lower, then the **Value** of the **Output** Object is set to the **Input** Object's **Value**.

Data-Type: *Number-Pointer*

Data-Range: Any Object with a Value property

Mode

A value that specifies how the counter evaluates the **ClockIn1** and **ClockIn2** properties.

Data-Type: *Nibble*

Data-Range: 0 - 1

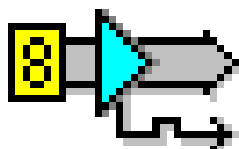
The following table list the values of the **Mode** Property.

Mode	Constant	Description
0	cvCount	Each cycle of the Flag property pointed to by the ClockIn1 property is considered to be a "Clock-Tick" and the Value of the Output Object is incremented or decremented in the direction specified by the Direction Property.
1	cvPhase	If the Mode property is set to cvPhase or 1, the Flags pointed to by the ClockIn1 and Clockin2 properties are Quadrant-Encoded and the Value of the Output Object is incremented or decremented in the direction specified by Quadrant change. The Direction Property is then updated to reflect the

	direction of the change.									
Operate	<p>A value that specifies whether or not the count is updated.</p> <p>Data-Type: <i>Bit, Flag, Default</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p> <table border="1"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The count is not updated.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The count is updated.</td> </tr> </tbody> </table>	<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The count is not updated.	1	cvTrue	The count is updated.
<u>Operate</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	The count is not updated.								
1	cvTrue	The count is updated.								
Output	<p>A pointer to an Object whose Value property is incremented or decremented for each "Clock-Tick".</p> <p>Data-Type: <i>Number-Pointer</i></p> <p>Data-Range: Any Object with a Value property</p>									
Tick	<p>A value that specifies how the oCounter Object values each "Clock-Tick".</p> <p>If the Tick property is set to 1, the Value property of the Output Object is cleared.</p> <p>If the Tick property is cleared to 0, the Value property of the Output Object is incremented or decremented by a value of 1.</p> <p>The Tick property is similar to a clear or reset function with the difference being that the Tick property is only evaluated when a "Clock-Tick" has occurred.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Tick Property.</p> <table border="1"> <thead> <tr> <th><u>Tick</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>Change the Output Object by 1.</td> </tr> <tr> <td>1</td> <td></td> <td>Set the Output Object to 0.</td> </tr> </tbody> </table>	<u>Tick</u>	<u>Constant</u>	<u>Description</u>	0		Change the Output Object by 1.	1		Set the Output Object to 0.
<u>Tick</u>	<u>Constant</u>	<u>Description</u>								
0		Change the Output Object by 1.								
1		Set the Output Object to 0.								

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oDataStrobe Object

An Object that provides a Data-Strobe in response to a value being written to it.

Description:

Size: 5 Bytes

Properties: The following table lists the Properties of the **oDataStrobe** Object

<u>Property</u>	<u>Description</u>						
Address	<p>Returns a pointer to the address of the oDataStrobe Object instance.</p> <p>Data-Type: <i>Number-Pointer, ReadOnly</i></p> <p>Data-Range: 0 - 127</p>						
Mode	<p>A value that indicates if the oDataStrobe Object will use an 8-bit mode or a dual 4-bit mode.</p> <p>Data-Type: <i>Bit</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Mode Property.</p> <table border="1"> <thead> <tr> <th><u>Mode</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cv8Bit</td> <td>The Value property will be transferred as a single 8-bit value and a single data-strobe will be generated when the Value property is written to.</td> </tr> </tbody> </table>	<u>Mode</u>	<u>Constant</u>	<u>Description</u>	0	cv8Bit	The Value property will be transferred as a single 8-bit value and a single data-strobe will be generated when the Value property is written to.
<u>Mode</u>	<u>Constant</u>	<u>Description</u>					
0	cv8Bit	The Value property will be transferred as a single 8-bit value and a single data-strobe will be generated when the Value property is written to.					

1	cv4Bit	The Value property will be transferred as two 4-bit values and a data-strobe will be generated for each when the Value property is written to.
---	---------------	--

Nibble

A value that indicates if the data sent to the [Output](#) Object is the Low-Nibble or the High-Nibble.

Data-Type: *Nibble, Flag*

Data-Range: 0 - 1

The following table list the values of the **Nibble** Property.

<u>Nibble</u>	<u>Constant</u>	<u>Description</u>
0	cvLow	The lowest 4 bits of the Value property is being used for the current data strobe.
1	cvHigh	The Upper 4 bits of the Value property is being used for the current data strobe.

OnChange

A value that specifies if the Data-Strobe is generated only when the value has changed.

Data-Type: *Bit*

Data-Range: 0 - 1

The following table list the values of the **OnChange** Property.

<u>OnChange</u>	<u>Constant</u>	<u>Description</u>
0	cvFalse	The Value Property is transferred and the data strobe is generated each time the Value property is written to.
1	cvTrue	The Value Property is transferred and the data strobe is generated only when the Value property is written to with a value different than the

	last value written.									
Operate	<p>A value that indicates if a Data-Strobe is generated in response to new data.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p> <table border="1"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The Output Objects Value property is not updated.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The Output Objects Value property is updated.</td> </tr> </tbody> </table>	<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The Output Objects Value property is not updated.	1	cvTrue	The Output Objects Value property is updated.
<u>Operate</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	The Output Objects Value property is not updated.								
1	cvTrue	The Output Objects Value property is updated.								
Output	<p>A pointer that specifies which Object receives the data written to the Value property.</p> <p>Data-Type: <i>Number-Pointer</i></p> <p>Data-Range: 0 - 127</p>									
Result	<p>A value that indicates the Data-Strobe is active.</p> <p>Data-Type: <i>Byte, Flag</i></p> <p>Data-Range: 0 - 1</p>									
String	<p>The Value property represented as a string.</p> <p>Note: The String property is available in OOPic Basic Version 2</p> <p>Data-Type: <i>String</i></p>									
Strobe	<p>A pointer that specifies which Flag property is used as a Data-Strobe.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: 0 - 127</p>									

Value	<p>A value that is transferred to the Unit Object.</p> <p>Data-Type: <i>Byte, Default</i></p> <p>Data-Range: 0 - 255</p>
--------------	--

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oDDELink Object

An Object that provides a Dynamic-Data-Exchange link over the I2C network.

Description: A Processing Object that dynamically exchanges data with other OOPics by transmitting information over an I2C network of which both OOPics are connected.

Operation: An **oDDELink** Object uses the I2C network to transfer data to and from a local Object located within the same physical OOPic and a remote Object located in any other physical OOPic connected to the I2C network. A **oDDELink** Object is required in both the Local OOPic and the Remote OOPic to complete a communication link over the I2C network. To create a communication link, three specifications must be made, 1: The local Object is specified by the [Input](#) and the [Output](#) properties of the local **oDDELink** Object. 2: The Remote Object is specified with the [Input](#) and [Output](#) properties of the Remote **oDDELink** Object. 3: The Remote **oDDELink** Object is specified by the [Node](#) and [Location](#) properties of the local **oDDELink** object. (See "Using the oDDELink Object" in the OOPic Programmers guide for more information on the 3 specifications). When the [Operate](#) property of the local **oDDELink** Object is set to [cvTrue](#), it will initiate a communication link between the local **oDDELink** Object and the remote **oDDELink** Object and transfer the data. When a data transfer is successfully initiated, the default properties of the two [Input](#) and [Output](#) Objects will have been copied in the direction specified by the [Direction](#) property and

the [Operate](#) property will be cleared to 0. If the network is busy, or if an I2C arbitration is lost, the transmission will wait until the network is not busy and try again.

Size: 8 Bytes

Properties: The following table lists the Properties of the **oDDELink** Object

<u>Property</u>	<u>Description</u>									
Address	<p>Returns a pointer to the address of the oDDELink Object instance.</p> <p>Data-Type: <i>Number-Pointer, ReadOnly</i></p> <p>Data-Range: 0 - 127</p>									
Direction	<p>A value specifying the direction of the data flow on the I2C network.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Direction Property.</p> <table border="1"> <thead> <tr> <th><u>Direction</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvSend</td> <td>Data is copied from the local Object specified by the Link property to the remote Object specified by the Node and Location properties.</td> </tr> <tr> <td>1</td> <td>cvReceive</td> <td>Data is copied from the remote Object specified by the Node and Location properties to the local Object specified by the Link property.</td> </tr> </tbody> </table>	<u>Direction</u>	<u>Constant</u>	<u>Description</u>	0	cvSend	Data is copied from the local Object specified by the Link property to the remote Object specified by the Node and Location properties.	1	cvReceive	Data is copied from the remote Object specified by the Node and Location properties to the local Object specified by the Link property.
<u>Direction</u>	<u>Constant</u>	<u>Description</u>								
0	cvSend	Data is copied from the local Object specified by the Link property to the remote Object specified by the Node and Location properties.								
1	cvReceive	Data is copied from the remote Object specified by the Node and Location properties to the local Object specified by the Link property.								
Input	<p>A pointer to a local-Object whose default property value will be send to the remote-Object's default property.</p> <p>Data-Type: <i>Number-Pointer</i></p>									

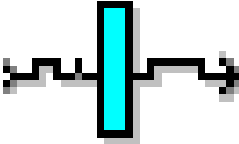
	Data-Range: Any Object with a Value property									
Location	<p>A pointer to a remote oDDELink Object instance whose Link-Object's data will be exchanged.</p> <p>Data-Type: <i>Byte</i></p> <p>Data-Range: 0 - 255</p>									
Node	<p>The Node address of the I2C networked OOPic that contains the remote oDDELink Object.</p> <p>Data-Type: <i>Byte</i></p> <p>Data-Range: 0 - 127</p> <p>The following table list the values of the Node Property.</p> <table border="1"> <thead> <tr> <th><u>Node</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>The oDDELink Object instance is inactive.</td> </tr> <tr> <td>1 - 127</td> <td></td> <td>The oDDELink Object will exchange data with the OOPic on the I2C network that has that Node address.</td> </tr> </tbody> </table>	<u>Node</u>	<u>Constant</u>	<u>Description</u>	0		The oDDELink Object instance is inactive.	1 - 127		The oDDELink Object will exchange data with the OOPic on the I2C network that has that Node address.
<u>Node</u>	<u>Constant</u>	<u>Description</u>								
0		The oDDELink Object instance is inactive.								
1 - 127		The oDDELink Object will exchange data with the OOPic on the I2C network that has that Node address.								
Operate	<p>A value that specifies whether or not the data is updated.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p> <table border="1"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The Value properties are not updated.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The Value properties are updated.</td> </tr> </tbody> </table>	<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The Value properties are not updated.	1	cvTrue	The Value properties are updated.
<u>Operate</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	The Value properties are not updated.								
1	cvTrue	The Value properties are updated.								

<p>Output</p>	<p>A pointer to a local-Object whose Value property will be updated with the remote-Object's default property.</p> <p>Data-Type: <i>Number-Pointer</i></p> <p>Data-Range: Any Object with a Value property</p> <p>The following table list the values of the Output Property.</p> <table border="1" data-bbox="624 566 1441 792"> <thead> <tr> <th><u>Output</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>The oDDELink Object instance is inactive.</td> </tr> <tr> <td>1 - 127</td> <td></td> <td>The local-Object's Address.</td> </tr> </tbody> </table>	<u>Output</u>	<u>Constant</u>	<u>Description</u>	0		The oDDELink Object instance is inactive.	1 - 127		The local-Object's Address .
<u>Output</u>	<u>Constant</u>	<u>Description</u>								
0		The oDDELink Object instance is inactive.								
1 - 127		The local-Object's Address .								
<p>Sync</p>	<p>A value instruct the oDDELink Object to synchronize its data with the remote oDDELink Object's data.</p> <p>Data-Type: <i>Bit, Flag, Default</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Sync Property.</p> <table border="1" data-bbox="624 1245 1441 1518"> <thead> <tr> <th><u>Sync</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The Value properties are not synchronizing.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The Value properties are synchronizing.</td> </tr> </tbody> </table>	<u>Sync</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The Value properties are not synchronizing.	1	cvTrue	The Value properties are synchronizing.
<u>Sync</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	The Value properties are not synchronizing.								
1	cvTrue	The Value properties are synchronizing.								
<p>Transmitting</p>	<p>A Value that specifies if the oDDELink Object is currently transferring data.</p> <p>Data-Type: <i>Bit, Flag, ReadOnly</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Transmitting Property.</p> <table border="1" data-bbox="624 1921 1441 2080"> <thead> <tr> <th><u>Transmitting</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The oDDELink Object instance is idle.</td> </tr> </tbody> </table>	<u>Transmitting</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The oDDELink Object instance is idle.			
<u>Transmitting</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	The oDDELink Object instance is idle.								

	1	cvTrue	The oDDELink Object instance is transferring data.
--	---	--------	---

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

-



oDebounce Object

An Object that provides logic-state debouncing functions.

Description: A Processing Object that stabilizes an input by ignoring it's changes for a period of time after going to a logical High state.

Operation: An **oDebounce** Object takes the Flag property pointed to by the [Input](#) property and extends it by the amount of time specified by the [Period](#) property.

Size: 5 Bytes

Properties: The following table lists the Properties of the **oDebounce** Object

<u>Property</u>	<u>Description</u>
Address	Returns a pointer to the address of the oDebounce Object instance. Data-Type: <i>Number-Pointer, ReadOnly</i> Data-Range: 0 - 127
Input	A pointer to an Object's Flag property that will be used as the value to debounce. Data-Type: <i>Flag-Pointer</i> Data-Range: Any Object's Flag property

<p>InvertIn</p>	<p>A value that specifies if the Input Flag is inverted before used in the Debounce operation.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the InvertIn Property.</p> <table border="1" data-bbox="576 521 1437 1115"> <thead> <tr> <th><u>InvertIn</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>When the value of the Object pointed to by the Input property is set 1, the Value property is set to 1 after an amount of time specified by the Period property has passed.</td> </tr> <tr> <td>1</td> <td></td> <td>When the value of the Object pointed to by the Input property is set 1, the Value property is set to 1 after the amount of time specified by the Period property has elapsed.</td> </tr> </tbody> </table>	<u>InvertIn</u>	<u>Constant</u>	<u>Description</u>	0		When the value of the Object pointed to by the Input property is set 1, the Value property is set to 1 after an amount of time specified by the Period property has passed.	1		When the value of the Object pointed to by the Input property is set 1, the Value property is set to 1 after the amount of time specified by the Period property has elapsed.
<u>InvertIn</u>	<u>Constant</u>	<u>Description</u>								
0		When the value of the Object pointed to by the Input property is set 1, the Value property is set to 1 after an amount of time specified by the Period property has passed.								
1		When the value of the Object pointed to by the Input property is set 1, the Value property is set to 1 after the amount of time specified by the Period property has elapsed.								
<p>InvertOut</p>	<p>A value that specifies if the Boolean Result of the debounce operation is inverted before it is stored into the Output Flag.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the InvertOut Property.</p> <table border="1" data-bbox="576 1563 1437 1883"> <thead> <tr> <th><u>InvertOut</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>The debounced value is written to the Output Object</td> </tr> <tr> <td>1</td> <td></td> <td>The debounced value is inverted before written to the Output Object</td> </tr> </tbody> </table>	<u>InvertOut</u>	<u>Constant</u>	<u>Description</u>	0		The debounced value is written to the Output Object	1		The debounced value is inverted before written to the Output Object
<u>InvertOut</u>	<u>Constant</u>	<u>Description</u>								
0		The debounced value is written to the Output Object								
1		The debounced value is inverted before written to the Output Object								

<p>Operate</p>	<p>A value that specifies whether or not the Value property is updated.</p> <p>Data-Type: <i>Bit, Flag, Default</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p> <table border="1" data-bbox="576 524 1437 703"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The Value property is not updated.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The Value property is updated.</td> </tr> </tbody> </table>	<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The Value property is not updated.	1	cvTrue	The Value property is updated.
<u>Operate</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	The Value property is not updated.								
1	cvTrue	The Value property is updated.								
<p>Output</p>	<p>A pointer to an Object's Flag property that will be updated with the debounced value.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>									
<p>Period</p>	<p>The amount of time in 1/60th of a second increments that the debounce operation ignores the Input Objects value after it has changed to a logical High state.</p> <p>Data-Type: <i>Byte</i></p> <p>Data-Range: 0 - 127</p>									
<p>Result</p>	<p>The Boolean result of the debounce evaluation. Indicates debounced value of the Input Object.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Result Property.</p> <table border="1" data-bbox="576 1704 1437 1973"> <thead> <tr> <th><u>Result</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>The Input value has not gone to a Logical High state.</td> </tr> <tr> <td>1</td> <td></td> <td>The Input value has gone to a logical High state.</td> </tr> </tbody> </table>	<u>Result</u>	<u>Constant</u>	<u>Description</u>	0		The Input value has not gone to a Logical High state.	1		The Input value has gone to a logical High state.
<u>Result</u>	<u>Constant</u>	<u>Description</u>								
0		The Input value has not gone to a Logical High state.								
1		The Input value has gone to a logical High state.								

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oEvent Object

An Object that runs program code in response to an event.

Description: A Processing Object that calls a **Sub** procedure when it's [Operate](#) property is set to 1.

Operation: When an **oEvent** Object's [Operate](#) property transitions from 0 to 1, the program flow is interrupted with a call to a **Sub** procedure specified by the **oEvent** Object's name. While the **Sub** procedure is executing, the **oEvent** Object ignores the [Operate](#) property and waits for the program flow to return from the **Sub** procedure. When the program flow returns from the **Sub** procedure, the **oEvent** Object resumes watching for the [Operate](#) property to transition from 0 to 1.

The name of the **Sub** procedure that the **oEvent** object expects to call is the name of the **oEvent** object followed by "_CODE" I.E. An **oEvent** object named "A" will call a **Sub** procedure named "A_CODE"

Size: 3 Bytes

Properties: The following table lists the Properties of the **oEvent** Object

<u>Property</u>	<u>Description</u>
Address	Returns a pointer to the address of the oEvent Object instance. Data-Type: <i>Number-Pointer, ReadOnly</i> Data-Range: 0 - 127
Operate	A value that specifies whether or not the event is acknowledged.

Data-Type: *Bit, Flag, Default*

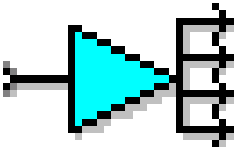
Data-Range: 0 - 1

The following table list the values of the **Operate** Property.

<u>Operate</u>	<u>Constant</u>	<u>Description</u>
0	cvFalse	The oEvent Object's Code is not executed.
1	cvTrue	The oEvent Object's Code is executed.

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oFanOut Object

An Object that takes an input value and copies it to other objects.

Description: A Processing Object

Operation: When an **oFanOut** Object's [Operate](#) property = 1, the Flag property pointed to by the [Input](#) property is copied to the Flag properties pointed to by the [Output](#) properties. An [InvertOut](#) property allows each individual Output to be optionally inverted.

Size: 2 + (number of specified outputs) Bytes

Properties: The following table lists the Properties of the **oFanOut** Object

<u>Property</u>	<u>Description</u>
Address	Returns a pointer to the address of the oFanOut Object instance.

	<p>Data-Type: <i>Number-Pointer, ReadOnly</i></p> <p>Data-Range: 0 - 127</p>									
Input	<p>A pointer to an Object's Flag property whose value will be copied to the Output Objects.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>									
InvertOut1	<p>A value that specifies if the Boolean value is inverted before stored into the Output1 Flag.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the InvertOut1 Property.</p> <table border="1"> <thead> <tr> <th><u>InvertOut1</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	<u>InvertOut1</u>	<u>Constant</u>	<u>Description</u>	0			1		
<u>InvertOut1</u>	<u>Constant</u>	<u>Description</u>								
0										
1										
InvertOut2	<p>A value that specifies if the Boolean value is inverted before stored into the Output2 Flag.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the InvertOut2 Property.</p> <table border="1"> <thead> <tr> <th><u>InvertOut2</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	<u>InvertOut2</u>	<u>Constant</u>	<u>Description</u>	0			1		
<u>InvertOut2</u>	<u>Constant</u>	<u>Description</u>								
0										
1										
InvertOut3	<p>A value that specifies if the Boolean value is inverted before stored into the Output3 Flag.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p>									

	<p>The following table list the values of the InvertOut3 Property.</p> <table border="1"> <thead> <tr> <th><u>InvertOut3</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	<u>InvertOut3</u>	<u>Constant</u>	<u>Description</u>	0			1		
<u>InvertOut3</u>	<u>Constant</u>	<u>Description</u>								
0										
1										
<p>InvertOut4</p>	<p>A value that specifies if the Boolean value is inverted before stored into the Output4 Flag.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the InvertOut4 Property.</p> <table border="1"> <thead> <tr> <th><u>InvertOut4</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	<u>InvertOut4</u>	<u>Constant</u>	<u>Description</u>	0			1		
<u>InvertOut4</u>	<u>Constant</u>	<u>Description</u>								
0										
1										
<p>Operate</p>	<p>A value that specifies whether or not the Output Objects are updated with the result of the logic operation.</p> <p>Data-Type: <i>Bit, Flag, Default</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p> <table border="1"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>The Output Objects are not updated.</td> </tr> <tr> <td>1</td> <td></td> <td>The Output Objects are updated.</td> </tr> </tbody> </table>	<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0		The Output Objects are not updated.	1		The Output Objects are updated.
<u>Operate</u>	<u>Constant</u>	<u>Description</u>								
0		The Output Objects are not updated.								
1		The Output Objects are updated.								
<p>Output1</p>	<p>Pointer to an Object's Flag property where the result of the operation will be stored. The result of the logic operation is not stored if the Output1 property is not set to point to an Objects Flag property or has been reset to Null.</p> <p>Always points to a Flag property.</p>									

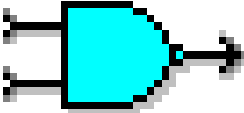
	<p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>
Output2	<p>Pointer to an Object's Flag property where the result of the operation will be stored. The result of the logic operation is not stored if the Output2 property is not set to point to an Objects Flag property or has been reset to Null.</p> <p>Always points to a Flag property.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>
Output3	<p>Pointer to an Object's Flag property where the result of the operation will be stored. The result of the logic operation is not stored if the Output3 property is not set to point to an Objects Flag property or has been reset to Null.</p> <p>Always points to a Flag property.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>
Output4	<p>Pointer to an Object's Flag property where the result of the operation will be stored. The result of the logic operation is not stored if the Output4 property is not set to point to an Objects Flag property or has been reset to Null.</p> <p>Always points to a Flag property.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>
Width	<p>The number of outputs specified when the Object instance was Dimmed.</p> <p>Read-only property.</p> <p>Data-Type: <i>Nibble, ReadOnly</i></p>

Data-Range: 0 - 3

See Also: **oGate, oWire**

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•

	<h2 style="margin: 0;"><u>oGate Object</u></h2> <p style="margin: 0;">An Object that provides logic-gate functions.</p>
---	--

Description: A Processing Object that can be configured to do multi-input logic gate operations like AND, NAND, OR, NOR, XOR, XNOR, NOT, LATCH, Etc.

Operation: An **oGate** Object takes the Flag properties pointed to by one or more [Input](#) pointers and performs a logical OR operation. The logical OR operation can be changed by setting the [InvertIn1-8](#), [InvertOut](#) and the **Exclusive** properties. The properties [InvertIn1](#) - [InvertIn8](#) invert the input values to the logical OR operation while the [Exclusive](#) property determines if the inputs will be evaluated with a Boolean-OR or a Boolean-XOR logic. The property [InvertOut](#) causes the result of the evaluation to be inverted. The [Operate](#) property determines if the [Value](#) property is updated with the final result of the logic operation or if the previous value is latched. If the [Value](#) property is updated, then its value is also stored in the Flag property of the Object pointed to by the [Output](#) property.

Remarks: The number of inputs that an instance of an **oGate** Object will have is specified by the subscript value that is used when the instance is dimensioned with the **Dim** statement.

oGate Objects can be setup to emulate the functionality of several different integrated circuits such as 7400, 7402, 7404, 7408, 7410, 7420, 7428,7430, 7475, etc.

Size: 2 + (number of specified inputs) Bytes

Properties: The following table lists the Properties of the **oGate** Object

<u>Property</u>	<u>Description</u>									
Address	<p>Returns a pointer to the address of the oGate Object instance.</p> <p>Data-Type: <i>Number-Pointer, ReadOnly</i></p> <p>Data-Range: 0 - 127</p>									
Exclusive	<p>Used to determine if the inputs are exclusive.</p> <p>Data-Type: <i>Bit</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Exclusive Property.</p> <table border="1"> <thead> <tr> <th><u>Exclusive</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>Inputs are not exclusive and the gate performs a logical OR.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>Inputs are exclusive and the gate performs a logical XOR.</td> </tr> </tbody> </table>	<u>Exclusive</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	Inputs are not exclusive and the gate performs a logical OR.	1	cvTrue	Inputs are exclusive and the gate performs a logical XOR.
<u>Exclusive</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	Inputs are not exclusive and the gate performs a logical OR.								
1	cvTrue	Inputs are exclusive and the gate performs a logical XOR.								
Input1	<p>A pointer to an Object's Flag property whose value is used as an input to the logic operation. A logic 0 is used if it has not been set to point to an Object's Flag property or has been reset to Null.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>									

Input2	<p>A pointer to an Object's Flag property whose value is used as an input to the logic operation. A logic 0 is used if it has not been set to point to an Object's Flag property or has been reset to Null.</p> <p>Not available if the oGate Object was dimensioned with less than 2 inputs.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>
Input3	<p>A pointer to an Object's Flag property whose value is used as an input to the logic operation. A logic 0 is used if it has not been set to point to an Object's Flag property or has been reset to Null.</p> <p>Not available if the oGate Object was dimensioned with less than 3 inputs.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>
Input4	<p>A pointer to an Object's Flag property whose value is used as an input to the logic operation. A logic 0 is used if it has not been set to point to an Object's Flag property or has been reset to Null.</p> <p>Not available if the oGate Object was dimensioned with less than 4 inputs.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>
Input5	<p>A pointer to an Object's Flag property whose value is used as an input to the logic operation. A logic 0 is used if it has not been set to point to an Object's Flag property or has been reset to Null.</p> <p>Not available if the oGate Object was dimensioned with less than 5 inputs.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>

<p>Input6</p>	<p>A pointer to an Object's Flag property whose value is used as an input to the logic operation. A logic 0 is used if it has not been set to point to an Object's Flag property or has been reset to Null.</p> <p>Not available if the oGate Object was dimensioned with less than 6 inputs.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>
<p>Input7</p>	<p>A pointer to an Object's Flag property whose value is used as an input to the logic operation. A logic 0 is used if it has not been set to point to an Object's Flag property or has been reset to Null.</p> <p>Not available if the oGate Object was dimensioned with less than 7 inputs.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>
<p>Input8</p>	<p>A pointer to an Object's Flag property whose value is used as an input to the logic operation. A logic 0 is used if it has not been set to point to an Object's Flag property or has been reset to Null.</p> <p>Not available if the oGate Object was dimensioned with less than 8 inputs.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>
<p>InvertIn1</p>	<p>A value that specifies if the Boolean value of Input1 is inverted before the logical operation is performed. The InvertIn1 property is ignored if the corresponding Input pointer is set to Null.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the InvertIn1 Property.</p>

<u>InvertIn1</u>	<u>Constant</u>	<u>Description</u>
0	cvFalse	The value of Input1 is not inverted.
1	cvTrue	The value of Input1 is inverted.

InvertIn2 A value that specifies if the Boolean value of [Input2](#) is inverted before the logical operation is performed. The [InvertIn2](#) property is ignored if the corresponding Input pointer is set to Null.

Not available if the **oGate** Object was dimensioned with less than 2 inputs.

Data-Type: *Bit, Flag*

Data-Range: 0 - 1

The following table list the values of the **InvertIn2** Property.

<u>InvertIn2</u>	<u>Constant</u>	<u>Description</u>
0		
1		

InvertIn3 A value that specifies if the Boolean value of [Input3](#) is inverted before the logical operation is performed. The [InvertIn3](#) property is ignored if the corresponding Input pointer is set to Null.

Not available if the **oGate** Object was dimensioned with less than 3 inputs.

Data-Type: *Bit, Flag*

Data-Range: 0 - 1

The following table list the values of the **InvertIn3** Property.

<u>InvertIn3</u>	<u>Constant</u>	<u>Description</u>
0		
1		

InvertIn4 A value that specifies if the Boolean value of [Input4](#) is inverted before the logical operation is performed. The

[InvertIn4](#) property is ignored if the corresponding Input pointer is set to Null.

Not available if the **oGate** Object was dimensioned with less than 4 inputs.

Data-Type: *Bit, Flag*

Data-Range: 0 - 1

The following table list the values of the **InvertIn4** Property.

<u>InvertIn4</u>	<u>Constant</u>	<u>Description</u>
0		
1		

InvertIn5 A value that specifies if the Boolean value of [Input5](#) is inverted before the logical operation is performed. The [InvertIn5](#) property is ignored if the corresponding Input pointer is set to Null.

Not available if the **oGate** Object was dimensioned with less than 5 inputs.

Data-Type: *Bit, Flag*

Data-Range: 0 - 1

The following table list the values of the **InvertIn5** Property.

<u>InvertIn5</u>	<u>Constant</u>	<u>Description</u>
0		
1		

InvertIn6 A value that specifies if the Boolean value of [Input6](#) is inverted before the logical operation is performed. The [InvertIn6](#) property is ignored if the corresponding Input pointer is set to Null.

Not available if the **oGate** Object was dimensioned with less than 6 inputs.

Data-Type: *Bit, Flag*

Data-Range: 0 - 1

The following table list the values of the **InvertIn6** Property.

<u>InvertIn6</u>	<u>Constant</u>	<u>Description</u>
0		
1		

InvertIn7

A value that specifies if the Boolean value of [Input7](#) is inverted before the logical operation is performed. The [InvertIn7](#) property is ignored if the corresponding Input pointer is set to Null.

Not available if the **oGate** Object was dimensioned with less than 7 inputs.

Data-Type: *Bit, Flag*

Data-Range: 0 - 1

The following table list the values of the **InvertIn7** Property.

<u>InvertIn7</u>	<u>Constant</u>	<u>Description</u>
0		
1		

InvertIn8

A value that specifies if the Boolean value of [Input8](#) is inverted before the logical operation is performed. The **InvertIn8** property is ignored if the corresponding Input pointer is set to Null.

Not available if the **oGate** Object was dimensioned with less than 2 inputs.

Data-Type: *Bit, Flag*

Data-Range: 0 - 1

The following table list the values of the **InvertIn8** Property.

<u>InvertIn8</u>	<u>Constant</u>	<u>Description</u>
0		

	1									
InvertOut	<p>A value that specifies if the Boolean Result of the logic-gate operation is inverted before it is stored into the Output Flag.</p> <p>Data-Type: <i>Bit</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the InvertOut Property.</p> <table border="1" data-bbox="576 647 1437 875"> <thead> <tr> <th><u>InvertOut</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>Value and Output is not inverted.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>Value and Output is inverted.</td> </tr> </tbody> </table>	<u>InvertOut</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	Value and Output is not inverted.	1	cvTrue	Value and Output is inverted.
<u>InvertOut</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	Value and Output is not inverted.								
1	cvTrue	Value and Output is inverted.								
Operate	<p>A value that specifies whether or not the Output Object is updated with the result of the logic operation.</p> <p>Data-Type: <i>Bit, Flag, Default</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p> <table border="1" data-bbox="576 1281 1437 1599"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>Value and Output are not updated.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>Value and Output are updated with the result of the logical evaluation.</td> </tr> </tbody> </table>	<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	Value and Output are not updated.	1	cvTrue	Value and Output are updated with the result of the logical evaluation.
<u>Operate</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	Value and Output are not updated.								
1	cvTrue	Value and Output are updated with the result of the logical evaluation.								
Output	<p>A pointer to an Object's Flag property that will be updated with the result of the logic-gate operation.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>									
Result	<p>The Boolean result of the gate evaluation. This is calculated by ORing (or Exclusive-ORing if the Exclusive property is set to True) together the Boolean values of the inputs and then inverting the result if the</p>									

	<p>InvertOut property was set to cvTrue.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Result Property.</p> <table border="1"> <thead> <tr> <th><u>Result</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>The result of the gate evaluation is Logical Low.</td> </tr> <tr> <td>1</td> <td></td> <td>The result of the gate evaluation is Logical High.</td> </tr> </tbody> </table>	<u>Result</u>	<u>Constant</u>	<u>Description</u>	0		The result of the gate evaluation is Logical Low.	1		The result of the gate evaluation is Logical High.
<u>Result</u>	<u>Constant</u>	<u>Description</u>								
0		The result of the gate evaluation is Logical Low.								
1		The result of the gate evaluation is Logical High.								
Width	<p>The number of inputs specified when the Object instance was Dimmed.</p> <p>Read-only property.</p> <p>Data-Type: <i>Nibble, ReadOnly</i></p> <p>Data-Range: 0 - 7</p>									

See Also: **oFanOut, oWire**

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•

oIndex Object

An Object that provides indexing functions.

Description: A Processing Object that stores or retrieves values from an array of values or optionally searches the array for a specified value. it is capable of doing encoder / decoder / multiplexer / demultiplexer and lookup functions.

Operation: An **oIndex** Object points to a **oBuffer** Object that contains an array of information and stores or retrieves the value at the location within the array specified by the **oIndex** Object.

Size: 4 Bytes

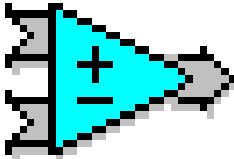
Properties: The following table lists the Properties of the **oIndex** Object

<u>Property</u>	<u>Description</u>						
Address	<p>Returns a pointer to the address of the oIndex Object instance.</p> <p>Data-Type: <i>Number-Pointer, ReadOnly</i></p> <p>Data-Range: 0 - 127</p>						
Array	<p>A pointer to a oBuffer Object from which values are returned from or stored to.</p> <p>Data-Type: <i>Buffer-Pointer</i></p> <p>Data-Range: Any oBuffer Object.</p>						
Direction	<p>In Indexed mode, Direction determines the direction of data flow.</p> <p>In Lookup mode, Direction determines the direction of the search.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Direction Property.</p> <table border="1"> <thead> <tr> <th><u>Direction</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvRetrieve</td> <td> <p>Retrieves the {Index}th value out of the Array Object and places it into the Value property of the Unit Object.</p> <p>This can also be done in code by using the value returned when specifying a oBuffer Object's subscript. I.E. X =</p> </td> </tr> </tbody> </table>	<u>Direction</u>	<u>Constant</u>	<u>Description</u>	0	cvRetrieve	<p>Retrieves the {Index}th value out of the Array Object and places it into the Value property of the Unit Object.</p> <p>This can also be done in code by using the value returned when specifying a oBuffer Object's subscript. I.E. X =</p>
<u>Direction</u>	<u>Constant</u>	<u>Description</u>					
0	cvRetrieve	<p>Retrieves the {Index}th value out of the Array Object and places it into the Value property of the Unit Object.</p> <p>This can also be done in code by using the value returned when specifying a oBuffer Object's subscript. I.E. X =</p>					

		Buffer(3).									
1	cvStore	Stores the Value property of the Unit Object into the { Index }th position of the Array Object. This can also be done in code by assigning a byte by specifying a oBuffer Object's subscript to a value. I.E. Buffer(2) = 13.									
Index	<p>A pointer to an Object whose Value property holds the position within the Array Object at which values are to be stored or retrieved.</p> <p>The offset to the {Index}th position within the Array Object is 0 based and is calculated as follows:</p> <p>(the bit-wide size of the Object pointed to by Unit) * (the Value property of the Object pointed to by Index) + 1.</p> <p>Data-Type: <i>Number-Pointer</i></p> <p>Data-Range: Any Object with a Value property</p>										
Operate	<p>A value that specifies whether or not the values are updated.</p> <p>Data-Type: <i>Bit, Flag, Default</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p> <table border="1"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The Values are not updated.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The Values are updated.</td> </tr> </tbody> </table>		<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The Values are not updated.	1	cvTrue	The Values are updated.
<u>Operate</u>	<u>Constant</u>	<u>Description</u>									
0	cvFalse	The Values are not updated.									
1	cvTrue	The Values are updated.									
Unit	<p>A pointer to an Object whose Value property is used to be stored in or returned from within the Array Object.</p> <p>Data-Type: <i>Number-Pointer</i></p> <p>Data-Range: Any Object with a Value property</p>										

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oMath Object

An Object that provides mathematical functions.

Description: A Processing Object that performs mathematical and Boolean operations.

Operation: An **oMath** Object takes the values of the two Objects pointed to by [Input1](#) & [Input2](#), performs the operation specified by the [Mode](#) property and then stores the resulting value in the Object pointed to by the [Output](#) property. Two evaluations are made on the resulting value and are stored. The [Negative](#) Flag stores the result of a Negative-Value evaluation and the [NonZero](#) flag stores the result of a Not-Zero-Value evaluation.

Size: 4 Bytes

Properties: The following table lists the Properties of the **oMath** Object

<u>Property</u>	<u>Description</u>
Address	Returns a pointer to the address of the oMath Object instance. Data-Type: <i>Number-Pointer, ReadOnly</i> Data-Range: 0 - 127
Input1	A pointer to an Object whose Value property is to be used in a mathematical operation. Data-Type: <i>Number-Pointer</i> Data-Range: Any Object with a Value property

<p>Input2</p>	<p>A pointer to an Object whose Value property is to be used in a mathematical operation.</p> <p>Data-Type: <i>Number-Pointer</i></p> <p>Data-Range: Any Object with a Value property</p>																											
<p>Mode</p>	<p>A value that specifies which one of 8 math operations to performed.</p> <p>Data-Type: <i>Nibble</i></p> <p>Data-Range: 0 - 7</p> <p>The following table list the values of the Mode Property.</p> <table border="1" data-bbox="555 745 1436 1704"> <thead> <tr> <th><u>Mode</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvAdd</td> <td>Adds the values of Input1 and Input2</td> </tr> <tr> <td>1</td> <td>cvSubtract</td> <td>Subtracts the values of Input1 and Input2</td> </tr> <tr> <td>2</td> <td>cvLShift</td> <td>Bit-shifts left the values of Input1 a number of times specified by Input2.</td> </tr> <tr> <td>3</td> <td>cvRShift</td> <td>Bit-shifts right the values of Input1 a number of times specified by Input2.</td> </tr> <tr> <td>4</td> <td>cvAND</td> <td>Input1 is ANDed with Input2.</td> </tr> <tr> <td>5</td> <td>cvOr</td> <td>Input1 is ORed with Input2.</td> </tr> <tr> <td>6</td> <td>cvXOr</td> <td>Input1 is XORed with Input2.</td> </tr> <tr> <td>7</td> <td>cvLatch</td> <td>Input2 is copied to Output when Input1 is Not Zero in value.- If Input2 is not pointing to any Object, then Output is cleared.</td> </tr> </tbody> </table>	<u>Mode</u>	<u>Constant</u>	<u>Description</u>	0	cvAdd	Adds the values of Input1 and Input2	1	cvSubtract	Subtracts the values of Input1 and Input2	2	cvLShift	Bit-shifts left the values of Input1 a number of times specified by Input2 .	3	cvRShift	Bit-shifts right the values of Input1 a number of times specified by Input2 .	4	cvAND	Input1 is ANDed with Input2 .	5	cvOr	Input1 is ORed with Input2 .	6	cvXOr	Input1 is XORed with Input2 .	7	cvLatch	Input2 is copied to Output when Input1 is Not Zero in value.- If Input2 is not pointing to any Object, then Output is cleared.
<u>Mode</u>	<u>Constant</u>	<u>Description</u>																										
0	cvAdd	Adds the values of Input1 and Input2																										
1	cvSubtract	Subtracts the values of Input1 and Input2																										
2	cvLShift	Bit-shifts left the values of Input1 a number of times specified by Input2 .																										
3	cvRShift	Bit-shifts right the values of Input1 a number of times specified by Input2 .																										
4	cvAND	Input1 is ANDed with Input2 .																										
5	cvOr	Input1 is ORed with Input2 .																										
6	cvXOr	Input1 is XORed with Input2 .																										
7	cvLatch	Input2 is copied to Output when Input1 is Not Zero in value.- If Input2 is not pointing to any Object, then Output is cleared.																										
<p>Negative</p>	<p>If Mode is 0-3, A value that indicates the result of the Output-value-is-negative evaluation.</p> <p>If Mode is 4-7, A value that specifies if the bits in the result will be inverted.</p> <p>Data-Type: <i>Bit, Flag</i></p>																											

Data-Range: 0 - 1

The following table list the values of the **Negative** Property.

<u>Negative</u>	<u>Mode</u>	<u>Constant</u>	<u>Description</u>
0	0 - 3	cvPositive	Output is positive in value.
0	4 - 7	cvFalse	Output is unchanged.
1	0 - 3	cvNegative	Output is negative in value.
1	4 - 7	cvTrue	The bits in Output are inverted.

NonZero

A value that indicates the result of the [Output](#)'s value is not zero evaluation.

Data-Type: *Bit, Flag*

Data-Range: 0 - 1

The following table list the values of the **NonZero** Property.

<u>NonZero</u>	<u>Constant</u>	<u>Description</u>
0	cvFalse	Output is zero in value.
1	cvTrue	Output is not zero in value.

Operate

A value that specifies whether or not the [Output](#) Object is updated with the result of the calculation.

Data-Type: *Bit, Flag, Default*

Data-Range: 0 - 1

The following table list the values of the **Operate** Property.

<u>Operate</u>	<u>Constant</u>	<u>Description</u>
0	cvFalse	The data is not updated.
1	cvTrue	The data is updated.

Output	<p>A pointer to an Object whose Value property will be updated with the resulting of the mathematical operation.</p> <p>Data-Type: <i>Number-Pointer</i></p> <p>Data-Range: Any Object with a Value property</p>
---------------	--

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•

	<h2 style="margin: 0;"><u>oOneShot Object</u></h2> <p style="margin: 0;">An Object that produces a one-pulse output in response to logic transition.</p>
---	--

Description: A Processing Object that produces a one-pulse output in response to an [Input](#) Flag transitioning from 0 to 1.

Operation: An **oOneShot** Object stays idle until the [Input](#) Flag has changed from a logical Low state to a logical High state. When then this happens, it activates the one-shot logic and the [Result](#) property is set to 1 for an amount of time determined by the OOPic's Object-Loop. The OOPic's Object-Loop is calculated so that all of the other objects can process the a transition one and only one time. In addition to setting the [Result](#) property, the Flag pointed to by [Output](#) is also set to 1. After the OOPic's Object-Loop time has expired, the [Result](#) property is reset back to 0 and the Flag pointed to by the [Output](#) pointer is also set to 0. The **oOneShot** Object will wait until the [Input](#) Flag is set to 0 before it can be activated again.

If the [Operate](#) property is set to 0 the transitions are ignored.

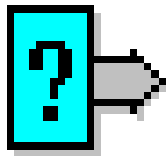
Size: 3 Bytes

Properties: The following table lists the Properties of the **oOneShot** Object

<u>Property</u>	<u>Description</u>
-----------------	--------------------

<p>Address</p>	<p>Returns a pointer to the address of the oOneShot Object instance.</p> <p>Data-Type: <i>Number-Pointer, ReadOnly</i></p> <p>Data-Range: 0 - 127</p>									
<p>Input</p>	<p>A pointer to an Object's Flag property whose value will be used to trigger the "One-Shot" pulse.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>									
<p>InvertIn</p>	<p>A value that specifies if the Input Flag is inverted before used in the One-Shot operation.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the InvertIn Property.</p> <table border="1" data-bbox="576 1066 1437 1249"> <thead> <tr> <th><u>InvertIn</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	<u>InvertIn</u>	<u>Constant</u>	<u>Description</u>	0			1		
<u>InvertIn</u>	<u>Constant</u>	<u>Description</u>								
0										
1										
<p>InvertOut</p>	<p>A value that specifies if the Boolean value of the One-Shot operation is inverted before it is stored into the Output Flag.</p> <p>Data-Type: <i>Bit</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the InvertOut Property.</p> <table border="1" data-bbox="576 1697 1437 1881"> <thead> <tr> <th><u>InvertOut</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	<u>InvertOut</u>	<u>Constant</u>	<u>Description</u>	0			1		
<u>InvertOut</u>	<u>Constant</u>	<u>Description</u>								
0										
1										

<p>Operate</p>	<p>A value that specifies if the Result property is pulsed once.</p> <p>Data-Type: <i>Bit, Flag, Default</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p> <table border="1" data-bbox="576 524 1437 792"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The Result property is not changed.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The Result property is set to 1 for 1 Object-Loop, then returned to 0.</td> </tr> </tbody> </table>	<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The Result property is not changed.	1	cvTrue	The Result property is set to 1 for 1 Object-Loop, then returned to 0.
<u>Operate</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	The Result property is not changed.								
1	cvTrue	The Result property is set to 1 for 1 Object-Loop, then returned to 0.								
<p>Output</p>	<p>A pointer to an Object's Flag property that will be updated with the "One-Shot" pulse.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>									
<p>Result</p>	<p>The Boolean result of the one-shot evaluation. Indicates that the Operate Flag has transitioned from a logical Low to a logical High.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Result Property.</p> <table border="1" data-bbox="576 1518 1437 1742"> <thead> <tr> <th><u>Result</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The oOneShot is not in its active state.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The oOneShot is in its active state.</td> </tr> </tbody> </table>	<u>Result</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The oOneShot is not in its active state.	1	cvTrue	The oOneShot is in its active state.
<u>Result</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	The oOneShot is not in its active state.								
1	cvTrue	The oOneShot is in its active state.								



oRandomizer Object

An Object that provides a random number.

Description: A Processing Object that sets the value of another Object to a random number each time the Object-Loop is performed.

Operation: An **oRandomizer** Object applies the gaussian distributed random number formula to the [Value](#) of the Object pointed to by the [Output](#) property. When the [Operate](#) property is 1, the value is randomized each Object-Loop iteration. If the [Operate](#) property is 0, the value is left unchanged. For any given successive random number, the resulting value cannot be predicted.

Size: 4 Bytes


Properties: The following table lists the Properties of the **oRandomizer** Object

<u>Property</u>	<u>Description</u>									
Address	Data-Type: <i>Number-Pointer, ReadOnly</i> Data-Range: 0 - 127									
Operate	A value that specifies whether or not the Output Object is randomized Data-Type: <i>Bit, Flag, Default</i> Data-Range: 0 - 1 The following table list the values of the Operate Property.									
	<table border="1"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0			1		
<u>Operate</u>	<u>Constant</u>	<u>Description</u>								
0										
1										
Output	A pointer to an Object whose Value property will be randomized. Data-Type: <i>Number-Pointer</i>									

	Data-Range: Any Object with a Value property									
Result	<p>A random One-Bit Boolean Value.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Result Property.</p> <table border="1"> <thead> <tr> <th><u>Result</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	<u>Result</u>	<u>Constant</u>	<u>Description</u>	0			1		
<u>Result</u>	<u>Constant</u>	<u>Description</u>								
0										
1										

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

-



oRTC Object

An Object that maintains a Real Time Clock.

Description: A Processing Object that performs a Real Time Clock format counting operation by increasing or decreasing an **oBuffer** Object's array for each "Clock-Tick"

Size: 4 Bytes

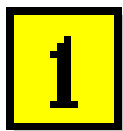
Properties: The following table lists the Properties of the **oRTC** Object

<u>Property</u>	<u>Description</u>
Address	<p>Returns a pointer to the address of the oRTC Object instance.</p> <p>Data-Type: <i>Number-Pointer, ReadOnly</i></p> <p>Data-Range: 0 - 127</p>

<p>ClockIn1</p>	<p>A pointer to an Object's Flag property whose value, When cycled, is used to increment and decrement the Value of the oBuffer Object pointed to by the Output property.</p> <p>A "Clock Tick" is counted when it changes from High to Low in value.</p> <p>Data-Type: <i>Flag-Pointer</i></p> <p>Data-Range: Any Object's Flag property</p>									
<p>Direction</p>	<p>Specifies the direction of the count.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Direction Property.</p> <table border="1" data-bbox="560 965 1433 1144"> <thead> <tr> <th><u>Direction</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvPositive</td> <td>Count positively (Increment)</td> </tr> <tr> <td>1</td> <td>cvNegative</td> <td>Count negatively (Decrement)</td> </tr> </tbody> </table>	<u>Direction</u>	<u>Constant</u>	<u>Description</u>	0	cvPositive	Count positively (Increment)	1	cvNegative	Count negatively (Decrement)
<u>Direction</u>	<u>Constant</u>	<u>Description</u>								
0	cvPositive	Count positively (Increment)								
1	cvNegative	Count negatively (Decrement)								
<p>Operate</p>	<p>A value that specifies whether or not the count is updated.</p> <p>Data-Type: <i>Bit, Flag, Default</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p> <table border="1" data-bbox="560 1552 1433 1731"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The count is not updated.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The count is updated.</td> </tr> </tbody> </table>	<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The count is not updated.	1	cvTrue	The count is updated.
<u>Operate</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	The count is not updated.								
1	cvTrue	The count is updated.								
<p>Output</p>	<p>A pointer to an oBuffer Object whose byte-array will be incremented or decremented in an 24-Hour clock fashion.</p> <p>Data-Type: <i>Buffer-Pointer</i></p> <p>Data-Range: Any oBuffer Object.</p>									

<p>PM</p>	<p>A value that indicates that the time is afternoon.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the PM Property.</p> <table border="1" data-bbox="560 432 1441 701"> <thead> <tr> <th><u>PM</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>The Real-Time-Clock is less than "12:00:00:00"</td> </tr> <tr> <td>1</td> <td></td> <td>The Real-Time-Clock is more than or equal to "12:00:00:00"</td> </tr> </tbody> </table>	<u>PM</u>	<u>Constant</u>	<u>Description</u>	0		The Real-Time-Clock is less than "12:00:00:00"	1		The Real-Time-Clock is more than or equal to "12:00:00:00"
<u>PM</u>	<u>Constant</u>	<u>Description</u>								
0		The Real-Time-Clock is less than "12:00:00:00"								
1		The Real-Time-Clock is more than or equal to "12:00:00:00"								
<p>Tick</p>	<p>A value that specifies how the oRTC Object values each "Clock-Tick".</p> <p>If the Tick property is set to 1, the 100ths second byte of the Output Object is incremented or decremented by 1 each clock-tick. If Tick property is cleared to 0, the seconds byte of the Output Object is incremented or decremented by 1 each clock-tick.</p> <p>Data-Type: <i>Bit</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Tick Property.</p> <table border="1" data-bbox="560 1328 1441 1686"> <thead> <tr> <th><u>Tick</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>Each "Tick" detected by the oRTC Object increments or decrements the ones column of the Seconds.</td> </tr> <tr> <td>1</td> <td></td> <td>Each "Tick" detected by the oRTC Object increments or decrements the ones column of the 1/60 Seconds.</td> </tr> </tbody> </table>	<u>Tick</u>	<u>Constant</u>	<u>Description</u>	0		Each "Tick" detected by the oRTC Object increments or decrements the ones column of the Seconds.	1		Each "Tick" detected by the oRTC Object increments or decrements the ones column of the 1/60 Seconds.
<u>Tick</u>	<u>Constant</u>	<u>Description</u>								
0		Each "Tick" detected by the oRTC Object increments or decrements the ones column of the Seconds.								
1		Each "Tick" detected by the oRTC Object increments or decrements the ones column of the 1/60 Seconds.								





oBit Object

An Object that maintains a 1-bit variable.

Description: A Variable Object whose [Value](#) property consists of 1 bit.

Operation: An **oBit** Object variable stores a 1 bit value in its [Value](#) property and maintains other values that represent the status of that 1 bit. A [NonZero](#) property indicates if the value of the 1 bit is not zero.

Remarks: An **oBit** Object uses only one byte of storage space. Therefore, it is one of the fastest & most efficient numeric variables.

Since the [Value](#) property of the **oBit** Object is also a Flag property, A Virtual Circuit connection may be made to the [Value](#) property by either a Object pointer or a Flag pointer.

Size: 1 Byte

Properties: The following table lists the Properties of the **oBit** Object

<u>Property</u>	<u>Description</u>						
Address	Returns a pointer to the address of the oBit Object instance. Data-Type: <i>Number-Pointer, ReadOnly</i> Data-Range: 0 - 127						
NonZero	Set to 1 when the Value property is not zero and cleared to 0 when the Value property is zero. Data-Type: <i>Bit, Flag</i> Data-Range: 0 - 1 The following table list the values of the NonZero Property.						
	<table border="1"> <thead> <tr> <th><u>NonZero</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The Value property is zero.</td> </tr> </tbody> </table>	<u>NonZero</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The Value property is zero.
<u>NonZero</u>	<u>Constant</u>	<u>Description</u>					
0	cvFalse	The Value property is zero.					

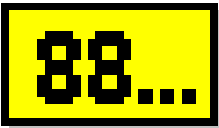
	1	cvTrue	The Value property is not zero.
String	The Value property represented as a string. Note: The String property is available in OOPic Basic Version 2 Data-Type: <i>String, Flag</i>		
Value	The value of the 1 bit. Data-Type: <i>Bit, Flag, Default</i> Data-Range: 0 - 1		

Methods: The following table lists the Methods of the **oBit** Object

<u>Methods</u>	<u>Description</u>
Clear	Clears the Value property to 0.
Invert	Inverts the bits in the Value property
Set	Sets the Value property to 1.

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•

	<h2><u>oBuffer Object</u></h2> <p>An Object that maintains a variable size data-buffer/string variable.</p>
---	---

Description: A Variable Object that consists of an array of up to 32 Bytes.

Operation: An **oBuffer** Object maintains an array of bytes. Access to the array of bytes can be done as a string of characters or by specifying one byte within the array.

When accessing the array as a string of characters, The [String](#)

property is used to read or write a series of characters.

When reading as a string of characters, each character within the string will be sequentially read each time the **String** property is accessed. The first character that is read will be the character pointed to by the **Location** property. Each subsequent character will be in the direction of the **Direction** property. The last character read will be determined by one of the following methods; A. If the **Direction** property is 0, the character at the last position in the array. B. If the **Direction** property is 1, the character at the first position in the array. C. If the character at the **Location** position is equal to 0 or 13 then the previously read character will be the last one read. After all characters have been read, the **Location** property will be at the position of the last character read.

When writing as a string of characters. each character written will be sequentially stored starting at the position specified by the **Location** property. and each subsequent character written will be stored at the next position in the direction of the **Direction** property. After all characters are written, the **Location** property will be located at the next character position.

When accessing the array by specifying a single byte, the **Value** property is used to read or write a specified byte within the array and the location of the byte that gets read from or written to is specified by the **Location** property.

Remarks: The number of bytes that an instance of an **oBuffer** Object will have in it's array is specified by the subscript value that is used when the instance is dimensioned with the **Dim** statement.

An **oBuffer** Object is the closest thing that the OOPic language has to a string.

Size: 4 + (number specified in Width) Bytes.

Properties: The following table lists the Properties of the **oBuffer** Object

<u>Property</u>	<u>Description</u>
-----------------	--------------------

Address	<p>Returns a pointer to the address of the oBuffer Object instance.</p> <p>Data-Type: <i>Number-Pointer, ReadOnly</i></p> <p>Data-Range: 0 - 127</p>									
Location	<p>A number that is used to specify a specific byte within the oBuffer Object's array. The Location property is moved to the next array position in the direction specified by the Direction property each time the Value property is read or written.</p> <p>Data-Type: <i>Byte</i></p> <p>Data-Range: 0 - 31</p>									
NonZero	<p>Set to 1 when any byte in the oBuffer Object's array is not zero and cleared to 0 when each byte in the oBuffer Object's array is 0.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the NonZero Property.</p> <table border="1" data-bbox="592 1249 1437 1429"> <thead> <tr> <th><u>NonZero</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	<u>NonZero</u>	<u>Constant</u>	<u>Description</u>	0			1		
<u>NonZero</u>	<u>Constant</u>	<u>Description</u>								
0										
1										
RTCString	<p>The entire contents of the oBuffer Object's array represented as an RTC string.</p> <p>Data-Type: <i>String</i></p>									
String	<p>The entire contents of the oBuffer Object's array represented as a string.</p> <p>Note: The String property is available in OOPic Basic Version 2</p> <p>Data-Type: <i>String</i></p>									
Value	<p>The byte within the oBuffer Object's array pointed to by the Location property</p>									

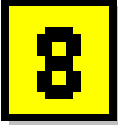
	Data-Type: <i>Byte, Default</i> Data-Range: 0 - 255
Width	The number of elements declared when the Object instance was dimensioned by the class size parameter. (see DIM statement) Data-Type: <i>Byte, ReadOnly</i> Data-Range: 1 - 32

Methods: The following table lists the Methods of the **oBuffer** Object

<u>Methods</u>	<u>Description</u>
Clear	Clears the Value property to 0.
Dec	Decrements the Value property by 1.
Inc	Increments the Value property by 1.
Invert	Inverts the bits in the Value property
LShift	Shifts the bits in the Value property left.
RShift	Shifts the bits in the Value property Right.
Set	Sets the Value property to 255.

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•

	<h2 style="margin: 0;"><u>oByte Object</u></h2> <p style="margin: 0;">An Object that maintains an 8-bit (1-byte) variable.</p>
---	--

Description: A Variable Object whose [Value](#) property consists of 8 bits.

Operation: An **oByte** Object variable stores an 8 bit value in its **Value** property and maintains other values that represent the status of those 8 bits. A **MSB** property indicates the Most-Significant-Bit of the value and a **NonZero** property indicates if the value of the 8 bits are not zero.

Size: 2 Bytes

Properties: The following table lists the Properties of the **oByte** Object

<u>Property</u>	<u>Description</u>									
Address	Returns a pointer to the address of the oByte Object instance. Data-Type: <i>Number-Pointer, ReadOnly</i> Data-Range: 0 - 127									
MSB	The MSB property sets and returns the most significant bit (MSB) of the oByte Object's Value property. Data-Type: <i>Bit, Flag</i> Data-Range: 0 - 1 The following table list the values of the MSB Property. <table border="1" style="margin-left: 20px;"> <thead> <tr> <th><u>MSB</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvPositive</td> <td>The Most-Significant-Bit of the Value property is 0.</td> </tr> <tr> <td>1</td> <td>cvNegative</td> <td>The Most-Significant-Bit of the Value property is 1.</td> </tr> </tbody> </table>	<u>MSB</u>	<u>Constant</u>	<u>Description</u>	0	cvPositive	The Most-Significant-Bit of the Value property is 0.	1	cvNegative	The Most-Significant-Bit of the Value property is 1.
<u>MSB</u>	<u>Constant</u>	<u>Description</u>								
0	cvPositive	The Most-Significant-Bit of the Value property is 0.								
1	cvNegative	The Most-Significant-Bit of the Value property is 1.								
NonZero	Set to 1 when the Value property is not zero and cleared to 0 when the Value property is zero. Set to 1 when the Value property is not zero and cleared to 0 when the Value property is zero. Data-Type: <i>Bit, Flag</i> Data-Range: 0 - 1 The following table list the values of the NonZero Property. <table border="1" style="margin-left: 20px;"> <thead> <tr> <th><u>NonZero</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> </tbody> </table>	<u>NonZero</u>	<u>Constant</u>	<u>Description</u>						
<u>NonZero</u>	<u>Constant</u>	<u>Description</u>								

	0	cvFalse	The Value property is zero.
	1	cvTrue	The Value property is not zero.
String	The Value property represented as a string. Note: The String property is available in OOPic Basic Version 2 Data-Type: <i>String</i>		
Value	The value of the 8 bits. Data-Type: <i>Byte, Default</i> Data-Range: 0 - 255		

Methods: The following table lists the Methods of the **oByte** Object

<u>Methods</u>	<u>Description</u>
Clear	Clears the Value property to 0.
Dec	Decrements the Value property by 1.
Inc	Increments the Value property by 1.
Invert	Inverts the bits in the Value property
LShift	Shifts the bits in the Value property left.
RShift	Shifts the bits in the Value property Right.
Set	Sets the Value property to 255.

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oEEProm Object

An Object that provides access to the non-volatile EEPROM memory.

Description: A Variable Object whose [Value](#) property is stored in the removable EEPROM memory chip located in the OOPic E0 and E1 EEPROM sockets.

Operation: The **oEEProm** Object reads/writes a byte from/to the EEPROM memory chip located in socket E0 or E1. The memory location that gets read from or written to is specified by the [Location](#) property. Each time the [Value](#) property is read or written to, the [Location](#) property will be increment by 1. The [Location](#) property can be assigned to the location of specified data by using the **Data** method.

The **oEEProm** Object's properties are identical in function to the **oI2C** Object's properties. The advantage to using the **oEEProm** Object is that, when the **Data** method is used, All of the properties are automatically setup for the EEPROM that is plugged into the E0 socket.

Remarks: The default property of the **oEEProm** Object cannot be assigned to a pointer.

Size: 5 Bytes

Properties: The following table lists the Properties of the **oEEProm** Object

<u>Property</u>	<u>Description</u>
Address	Returns a pointer to the address of the oEEProm Object instance. Data-Type: <i>Number-Pointer, ReadOnly</i> Data-Range: 0 - 127
Location	A value indicating the current read/write location in the EEPROM memory chip. Data-Type: <i>Word</i> Data-Range: 0 - 2047

<p>Mode</p>	<p>The I2C mode</p> <p>Data-Type: <i>Nibble</i></p> <p>Data-Range: 0 - 3</p> <p>The following table list the values of the Mode Property.</p> <table border="1" data-bbox="555 432 1437 862"> <thead> <tr> <th><u>Mode</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cv23Bit</td> <td>The oEEProm Object will use the 23-Bit I2C address mode.</td> </tr> <tr> <td>1</td> <td>cv10Bit</td> <td>The oEEProm Object will use the 10-Bit I2C address mode.</td> </tr> <tr> <td>2</td> <td>cv7Bit</td> <td>The oEEProm Object will use the 7-Bit I2C address mode.</td> </tr> <tr> <td>3</td> <td></td> <td>None</td> </tr> </tbody> </table>	<u>Mode</u>	<u>Constant</u>	<u>Description</u>	0	cv23Bit	The oEEProm Object will use the 23-Bit I2C address mode.	1	cv10Bit	The oEEProm Object will use the 10-Bit I2C address mode.	2	cv7Bit	The oEEProm Object will use the 7-Bit I2C address mode.	3		None
<u>Mode</u>	<u>Constant</u>	<u>Description</u>														
0	cv23Bit	The oEEProm Object will use the 23-Bit I2C address mode.														
1	cv10Bit	The oEEProm Object will use the 10-Bit I2C address mode.														
2	cv7Bit	The oEEProm Object will use the 7-Bit I2C address mode.														
3		None														
<p>Node</p>	<p>The I2C address to be used.</p> <p>Data-Type: <i>Byte</i></p> <p>Data-Range: 0 - 127</p>															
<p>NoInc</p>	<p>A value that specifies if the Location property will be incremented each time the oEEProm object's Value property is read or written.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the NoInc Property.</p> <table border="1" data-bbox="555 1496 1437 1906"> <thead> <tr> <th><u>NoInc</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The oEEProm Object increments the Location property each time the Value property is read or written</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The oEEProm Object does not increment the Location property each time the Value property is read or written</td> </tr> </tbody> </table>	<u>NoInc</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The oEEProm Object increments the Location property each time the Value property is read or written	1	cvTrue	The oEEProm Object does not increment the Location property each time the Value property is read or written						
<u>NoInc</u>	<u>Constant</u>	<u>Description</u>														
0	cvFalse	The oEEProm Object increments the Location property each time the Value property is read or written														
1	cvTrue	The oEEProm Object does not increment the Location property each time the Value property is read or written														
<p>String</p>	<p>The Value property represented as a string.</p> <p>Note: The String property is available in OOPic Basic</p>															

	Version 2 Data-Type: <i>String</i>									
Value	A value indicating the contents of the EEPROM's memory location specified by the Location property. When read, the value stored in the EEPROM memory chip will be returned. When Written, the value is stored into the EEPROM memory chip. Data-Type: <i>Byte, Default</i> Data-Range: 0 - 255									
Width	The number of Bytes to read/write Data-Type: <i>Bit</i> Data-Range: 0 - 1 The following table list the values of the Width Property.									
	<table border="1"> <thead> <tr> <th><u>Width</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cv8Bit</td> <td>The oEEProm Object will read and write 8 bits at a time</td> </tr> <tr> <td>1</td> <td>cv16Bit</td> <td>The oEEProm Object will read and write 16 bits at a time</td> </tr> </tbody> </table>	<u>Width</u>	<u>Constant</u>	<u>Description</u>	0	cv8Bit	The oEEProm Object will read and write 8 bits at a time	1	cv16Bit	The oEEProm Object will read and write 16 bits at a time
<u>Width</u>	<u>Constant</u>	<u>Description</u>								
0	cv8Bit	The oEEProm Object will read and write 8 bits at a time								
1	cv16Bit	The oEEProm Object will read and write 16 bits at a time								

Methods: The following table lists the Methods of the **oEEProm** Object

<u>Methods</u>	<u>Description</u>
Data	Fills the EEPROM with the specified arguments and sets the Location property to the data's address.

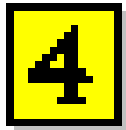
See Also: **oI2C** Object

Caution: If a program writes to the EEPROM memory chip when the [Location](#) is positioned before the program's end, then care must be taken to prevent it from overwriting the program code. If the [Location](#) has not been allocated for data, then any writes to the

oEEProm Object will overwrite the program code and the EEPROM memory will have to be re-programmed in order to retrieve the program code that was overwritten.

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oNibble Object

An Object that maintains a 4-bit variable.

Description: A Variable Object whose [Value](#) property consists of 4 bits.

Operation: An **oNibble** Object variable stores a 4 bit value in its [Value](#) property and maintains other values that represent the status of those 8 bits. A [NonZero](#) property indicates if the value of the 8 bits are not zero.

Remarks: An **oNibble** Object uses only one byte of storage space. Therefore, it is one of the fastest & most efficient numeric variables.

Size: 1 Byte

Properties: The following table lists the Properties of the **oNibble** Object

<u>Property</u>	<u>Description</u>
Address	Returns a pointer to the address of the oNibble Object instance. Data-Type: <i>Number-Pointer, ReadOnly</i> Data-Range: 0 - 127


NonZero	<p>Set to 1 when the Value property is not zero and cleared to 0 when the Value property is zero.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the NonZero Property.</p> <table border="1"> <thead> <tr> <th><u>NonZero</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The Value property is zero.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The Value property is not zero.</td> </tr> </tbody> </table>	<u>NonZero</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The Value property is zero.	1	cvTrue	The Value property is not zero.
<u>NonZero</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	The Value property is zero.								
1	cvTrue	The Value property is not zero.								
String	<p>The Value property represented as a string.</p> <p>Note: The String property is available in OOPic Basic Version 2</p> <p>Data-Type: <i>String</i></p>									
Value	<p>The value of the 4 bits.</p> <p>Data-Type: <i>Nibble, Default</i></p> <p>Data-Range: 0 - 15</p>									

Methods:

The following table lists the Methods of the **oNibble** Object

<u>Methods</u>	<u>Description</u>
Clear	Clears the Value property to 0.
Dec	Decrements the Value property by 1.
Inc	Increments the Value property by 1.
Invert	Inverts the bits in the Value property
LShift	Shifts the bits in the Value property left.
RShift	Shifts the bits in the Value property Right.
Set	Sets the Value property to 15.

•

	<h2><u>oRAM Object</u></h2> <p>An Object that provides access to the system's Random-Access-Memory.</p>
---	---

Description: A Variable Object whose [Value](#) property is stored in the internal RAM memory.

Operation: The **oRAM** Object reads/writes a byte from/to the system RAM memory. The memory location that gets read from or written to is specified by the [Location](#) property.

Remarks: The **oRAM** Object allows an application access to the internal registers of the Microchip PIC16C74 chip.

Size: 3 Bytes

Properties: The following table lists the Properties of the **oRAM** Object

<u>Property</u>	<u>Description</u>
Address	Returns a pointer to the address of the oRAM Object instance. Data-Type: <i>Number-Pointer, ReadOnly</i> Data-Range: 0 - 127
Location	A value indicating the current read/write location within the system's RAM memory. Data-Type: <i>Byte</i> Data-Range: 0 - 255
String	The Value property represented as a string. Note: The String property is available in OOPic Basic Version 2 Data-Type: <i>String</i>

Value	An 8-bit value indicating the value of a location in RAM. Data-Type: <i>Byte, Default</i> Data-Range: 0 - 255
--------------	---

Methods: The following table lists the Methods of the **oRAM** Object

<u>Methods</u>	<u>Description</u>
Clear	Clears the Value property to 0.
Dec	Decrements the Value property by 1.
Inc	Increments the Value property by 1.
Invert	Inverts the bits in the Value property
LShift	Shifts the bits in the Value property left.
RShift	Shifts the bits in the Value property Right.
Set	Sets the Value property to 255.

Caution:

This Object is intended for advanced users and its use is not recommended to those without a thorough knowledge of the internal workings of the OOPic.

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•



oWord Object

An Object that maintains a 16-bit (2-byte) variable.

Description: A Variable Object whose [Value](#) property consists of 16 bits.

Operation: An **oWord** Object variable stores a 16 bit value in its [Value](#) property and maintains other values that represent the status of those 16 bits.

A [MSB](#) property indicates the Most-Significant-Bit of the value and a [NonZero](#) property indicates if the value of the 16 bits is not zero.

Size: 3 Bytes

Properties: The following table lists the Properties of the **oWord** Object

<u>Property</u>	<u>Description</u>									
Address	Returns a pointer to the address of the oWord Object instance. Data-Type: <i>Number-Pointer, ReadOnly</i> Data-Range: 0 - 127									
MSB	The Negative property returns the most significant bit (MSB) of the oWord Object's Value property. Data-Type: <i>Bit, Flag</i> Data-Range: 0 - 1 The following table list the values of the MSB Property. <table border="1" style="margin-left: 20px;"> <thead> <tr> <th><u>MSB</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvPositive</td> <td>The Most-Significant-Bit of the Value property is 0.</td> </tr> <tr> <td>1</td> <td>cvNegative</td> <td>The Most-Significant-Bit of the Value property is 1.</td> </tr> </tbody> </table>	<u>MSB</u>	<u>Constant</u>	<u>Description</u>	0	cvPositive	The Most-Significant-Bit of the Value property is 0.	1	cvNegative	The Most-Significant-Bit of the Value property is 1.
<u>MSB</u>	<u>Constant</u>	<u>Description</u>								
0	cvPositive	The Most-Significant-Bit of the Value property is 0.								
1	cvNegative	The Most-Significant-Bit of the Value property is 1.								
NonZero	Set to 1 when the Value property is not zero and cleared to 0 when the Value property is zero. Data-Type: <i>Bit, Flag</i> Data-Range: 0 - 1 The following table list the values of the NonZero Property. <table border="1" style="margin-left: 20px;"> <thead> <tr> <th><u>NonZero</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The Value property is zero.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The Value property is not zero.</td> </tr> </tbody> </table>	<u>NonZero</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The Value property is zero.	1	cvTrue	The Value property is not zero.
<u>NonZero</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	The Value property is zero.								
1	cvTrue	The Value property is not zero.								
String	The Value property represented as a string.									

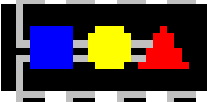
	Note: The String property is available in OOPic Basic Version 2 Data-Type: <i>String</i>
Value	The value of the 16 bits. Data-Type: <i>Word, Default</i> Data-Range: 0 - 65535

Methods: The following table lists the Methods of the **oWord** Object

<u>Methods</u>	<u>Description</u>
Clear	Clears the Value property to 0.
Dec	Decrements the Value property by 1.
Inc	Increments the Value property by 1.
Invert	Inverts the bits in the Value property
LShift	Shifts the bits in the Value property left.
RShift	Shifts the bits in the Value property Right.
Set	Sets the Value property to 65535.

Copyright(c) 1999,2000 by Savage Innovations All rights reserved

•

	<h2 style="margin: 0;"><u>OOPic Object</u></h2> <p style="margin: 0;">An Object that provides control of and maintains information about the OOPic.</p>
---	---

Description: A System Object that controls several functions of the OOPic.

Operation: The **OOPic** Object maintains and controls the internal operations of the OOPic Chip. Several of the internal operations can be controlled

and watched from the properties provided by the **OOPic** Object. Refer to the property's descriptions for more detailed information on each function.

Remarks: The **OOPic** Object is intrinsic. There is no class definition for it and new instances of it are not allowed.

Size: 5 Bytes

Properties: The following table lists the Properties of the **OOPic** Object

<u>Property</u>	<u>Description</u>									
ExtVRef	<p>A value that specifies the source of the voltage reference for the OOPics analog to digital module.</p> <p>Data-Type: Bit</p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the ExtVRef Property.</p> <table border="1"> <thead> <tr> <th><u>ExtVRef</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvOff</td> <td>oA2D Objects use a 5 volt range.</td> </tr> <tr> <td>1</td> <td>cvOn</td> <td>oA2D Objects use a voltage range specified by the voltage present on I/O line 4.</td> </tr> </tbody> </table>	<u>ExtVRef</u>	<u>Constant</u>	<u>Description</u>	0	cvOff	oA2D Objects use a 5 volt range.	1	cvOn	oA2D Objects use a voltage range specified by the voltage present on I/O line 4.
<u>ExtVRef</u>	<u>Constant</u>	<u>Description</u>								
0	cvOff	oA2D Objects use a 5 volt range.								
1	cvOn	oA2D Objects use a voltage range specified by the voltage present on I/O line 4.								
Hz1	<p>A 1-bit value that cycles once every second. The actual cycle time is .99957Hz</p> <p>Data-Type: Bit, Flag</p> <p>Data-Range: 0 - 1</p>									
Hz60	<p>A 1-bit value that cycles every 60hz.</p> <p>Data-Type: Bit, Flag</p> <p>Data-Range: 0 - 1</p>									
Node	<p>A value used when two or more OOPics "talk" to each other via the I2C network. A Node value of more than 0 is the OOPic's I2C network address.</p>									

	<p>Data-Type: <i>Byte</i></p> <p>Data-Range: 0 - 127</p> <p>The following table list the values of the Node Property.</p> <table border="1"> <thead> <tr> <th><u>Node</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvOff</td> <td>The OOPic's I2C network is inactive.</td> </tr> <tr> <td>1 - 127</td> <td></td> <td>The OOPic's I2C address is set.</td> </tr> </tbody> </table>	<u>Node</u>	<u>Constant</u>	<u>Description</u>	0	cvOff	The OOPic's I2C network is inactive.	1 - 127		The OOPic's I2C address is set.
<u>Node</u>	<u>Constant</u>	<u>Description</u>								
0	cvOff	The OOPic's I2C network is inactive.								
1 - 127		The OOPic's I2C address is set.								
Operate	<p>A value that specifies the power mode of the OOPic Chip.</p> <p>Data-Type: <i>Bit, Flag, Default</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Operate Property.</p> <table border="1"> <thead> <tr> <th><u>Operate</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvOff</td> <td>The OOPic is powered down.</td> </tr> <tr> <td>1</td> <td>cvOn</td> <td>The OOPic is on.</td> </tr> </tbody> </table>	<u>Operate</u>	<u>Constant</u>	<u>Description</u>	0	cvOff	The OOPic is powered down.	1	cvOn	The OOPic is on.
<u>Operate</u>	<u>Constant</u>	<u>Description</u>								
0	cvOff	The OOPic is powered down.								
1	cvOn	The OOPic is on.								
Pause	<p>A value that specifies if the program flow is suspended.</p> <p>Data-Type: <i>Bit, Flag</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the Pause Property.</p> <table border="1"> <thead> <tr> <th><u>Pause</u></th> <th><u>Constant</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>cvFalse</td> <td>The program executed statements normally.</td> </tr> <tr> <td>1</td> <td>cvTrue</td> <td>The program's execution is suspended.</td> </tr> </tbody> </table>	<u>Pause</u>	<u>Constant</u>	<u>Description</u>	0	cvFalse	The program executed statements normally.	1	cvTrue	The program's execution is suspended.
<u>Pause</u>	<u>Constant</u>	<u>Description</u>								
0	cvFalse	The program executed statements normally.								
1	cvTrue	The program's execution is suspended.								
PullUp	<p>A value that specifies the state of the internal pull-up resistors on I/O lines 8 - 15.</p> <p>Data-Type: <i>Bit</i></p> <p>Data-Range: 0 - 1</p> <p>The following table list the values of the PullUp Property.</p>									

<u>PullUp</u>	<u>Constant</u>	<u>Description</u>
0	cvOff	The pull-up resistors are not connected.
1	cvOn	The pull-up resistors are connected.

Reset A value that resets the OOPic when set.

Data-Type: *Bit, Flag*

Data-Range: 0 - 1

The following table list the values of the **Reset** Property.

<u>Reset</u>	<u>Constant</u>	<u>Description</u>
0	cvFalse	The OOPic operates normally.
1	cvTrue	The OOPic will reset setting the Startstat property to 3.

StartStat A value that indicates the cause of the last OOPic reset.

Data-Type: *Nibble, ReadOnly*

Data-Range: 0 - 3

The following table list the values of the **StartStat** Property.

<u>StartStat</u>	<u>Constant</u>	<u>Description</u>
0		Last reset was caused by power-on.
1		Last reset was caused by the reset line.
2		Last reset was caused by power-brownout.
3		Last reset was caused by Watch-Dog-Timer or the Reset property being set.