

**TABLE OF CONTENTS**

**A..... 1**

AND - A & B & C 1

AND - A & B & NOT C 1

AND - A & NOT B 1

AND - NOT A & NOT B 1

Append LONG to String: 1

Append Numeric Character as ASCII to String 2

Append WORD to String 2

Are Any Bits Set in Table?: 2

Are No Bits Set in Table?: 2

Average 2

**B..... 3**

Bit AND - A & B & C 3

Bit Change: 3

Bit Copy: 3

Bit Copy with Mask 3

Bit Mirror: 3

Bit Semaphore Reset: 4

Break 4

**C..... 5**

Case < 5

Case = 5

Case > 5

Case Within Limits: 5

Change Event Analog <=: 6

Change Event Analog >=: 6

Change Event Counter <=: 6

Change Event Counter >=: 6

Change Event Frequency <=: 7

Change Event Frequency >=: 7

Change Event MOMO:: 7

Change Event Off Pulse <=: 7

Change Event Off Pulse >=: 8

Change Event On Pulse <=: 8

Change Event On Pulse >=: 8

Change Event Totalizer <=: 8

Change Event Totalizer >=: 8

Change Reaction Analog Out: 9

Change Reaction Analog Ramp: 9

Change Reaction MOMO: 9

Change Reaction PID Setpoint: 9

Change Reaction Pulse Off: 10

Change Reaction Pulse On: 10

Characters Waiting (Port)? 10

Clear Receive Buffer (Port) 10

Convert 16-Bit to 32-Bit (signed): 10

Convert 32-Bit to 16-Bit (signed): 10

Convert 8-Bit Gray Code to 360 Degrees: 11

Convert Modem ASCII to Binary: 11

Convert String Characters to Hex String: 11

**D..... 12**

Decrement Table Variable:	12
Delay False Exit	12
Delay True Exit	12

**E..... 13**

Else..	13
EndIf	13
EndSwitch	13
Exclusive Access?	13

**F..... 14**

Flip Flop	14
Full Table Bit Clear:	14
Full Table Bit Set:	14
Full Table Bit Shift:	14
Full Table Bit Test:	14
Full Table Bit Toggle:	15
Full Table Load (Float):	15
Full Table Load (Integer):	15
Full Table Load (String):	15

**G..... 16**

Generate and Append HEX CRC16R to String:	16
Generate Checksum on String (16bit):	16
Generate Forward CCITT on Character:	16
Generate Reverse CRC-16 on Table:	16
Get Analog Full Scale	16
Get Analog Zero Scale	16
Get ARCNET Error	17
Get ARCNET Sequence Number	17
Get Calling Chart Serial Number	17
Get Chart Serial Number	17
Get Event/Reaction ID#	17
Get Exclusive Access	17
Get I/O Point Channel and Port #	18
Get Long from String	18
Get Name of Block Causing Current Error:	18
Get Number of Characters Waiting (Port)	18
Get Receive Buffer (Port):	18
Get Task # (0-31)	18

**H..... 19**

Hash Table Init:	19
Hash Table Lookup:	19
Hypotenuse	19

**I..... 20**

If AND	20
If Bit	20
If Changed	20
If Chart Running	20
If Chart Stopped	20
If Chart Suspended	21
If Equal	21
If Equal AND	21
If Equal And NOT	21

If Equal OR	21
If Equal Strings	22
If Event Occurring	22
If Exclusive Access	22
If False	22
If Generating Interrupt	22
If Greater	23
If Greater AND	23
If Greater OR	23
If Greater or Equal	23
If I/O Communication Enabled	23
If Less	24
If Less or Equal	24
If OR	24
If Table Elements Equal	24
If Timer Expired	24
If Timer Expired AND	25
If Timer Expired And NOT	25
If Timer Expired OR	25
If True	25
If Within Limits	25
Increment Table Variable:	26
Increment Variable 2x	26
Increment Variable 4x	26
Integer Table Lookup:	26
Invert Previous Conditions	26
I/O Communication Enabled?	26
Is Bit Clear in Table?:	27
Is Bit Set in Table?:	27
Is Event NOT Occurring?	27
IVAL If ON	27
IVAL Turn On A Latch	27
IVAL Write Digital from Binary Table Value	27
IVAL Write Digital from Binary Value	28

**L ..... 29**

Latch Not Set?	29
Latch Set OR?	29
Link Float Variable to Table:	29
Link Integer Variable to Table:	29
Link String Variable to Table:	29
Link SubTable to Table:	29
Loop...Until	30

**M ..... 31**

Move Analog Counts to Table	31
Move Digital I/O Unit to Table	31
Move from FIFO Table:	31
Move From Float Array:	31
Move From Integer Array:	31
Move From String Array:	32
Move from String FIFO Table:	32
Move String Table to String Table	32
Move Table to Table:	32
Move Table to Table (16-Bit):	32
Move to Float Array:	33
Move to Float FIFO Table:	33
Move to Integer Array:	33
Move to Integer FIFO Table:	33
Move To String Array:	33

Move to String FIFO Table:	34
Move with No Conversion (32bit Copy)	34
Move Word to Table	34

<b>N.....</b>	<b>35</b>
Not Variable	35

<b>O .....</b>	<b>36</b>
OR - A or B or C	36
OR - A or B or NOT C	36
OR - A or NOT B	36
OR - NOT A or NOT B	36
OR??	36
OR Block Begin – OR Previous With...	37
OR Block End	37

<b>P .....</b>	<b>38</b>
Put Port into ASCII I/O Master Mode:	38
Put Port into Binary I/O Master Mode:	38
Put Port into Binary I/O Slave Mode:	38
Put Port into Normal (Non-I/O) Mode:	38
Put Substring:	38

<b>R.....</b>	<b>39</b>
Read Byte from Controller Memory	39
Read Hi-Density Brick Temperature	39
Read Long from Controller Memory	39
Read PID Parameter:	39
Read Word from Controller Memory	40
Receive Character Via Serial (Port)	40
Receive Long from (Port) 1 <sup>st</sup> =MSB:	40
Receive Long from (Port) 4 <sup>th</sup> =MSB:	40
Receive Long from Port 1 <sup>st</sup> =MSB:	40
Receive Long from Port 4 <sup>th</sup> =MSB:	40
Receive Modem ASCII as Binary String via Serial Port:	41
Receive OPTOMUX Msg/Addr:	41
Receive OPTOMUX Msg:	41
Receive String Via Serial (Port)	41
Receive Table Via Serial (Port)	41
Release Active Port	42
Release Exclusive Access	42
Request Port	42

<b>S.....</b>	<b>43</b>
Set ARCNET Return Error	43
Set ARCNET Sequence Number	43
Set Column Count in Array:	43
Set Digital I/O Unit from Binary Table Value	43
Set String Table into String Array Column	43
Set Table Element False	43
Set Table Element True	44
Start PID Averaging:	44
Start Terminal Task:	44
Stop PID Averaging:	44
Stop Terminal Task:	44
String Table Lookup:	44

String Table Lookup SubString:	45
Strip Leading Alpha Characters:	45
Strip Leading Characters:	45
Summation	45
Summation Begin...	45
Summation End	46
Swap Values	46
Switch ... EndSwitch	46

**T..... 47**

Table Bit On?:	47
Table Bit Semaphore Reset:	47
Table Element Bit Change:	47
Table Element Bit OR	47
Table Element False?	47
Table Element True?	48
Table Elements Equal (Float)?	48
Table Elements Equal (Integer)?	48
Test Timer Expired	48
Transmit Character Via Serial (Port)	48
Transmit Date (Port)	48
Transmit Formatted Number (Port))	48
Transmit Long to (Port) 1 <sup>st</sup> =MSB	49
Transmit Long to (Port) 4 <sup>th</sup> =MSB	49
Transmit Long to Port 1 <sup>st</sup> =MSB	49
Transmit Long to Port 4 <sup>th</sup> =MSB	49
Transmit NewLine Via Serial (Port)	49
Transmit NewLine Via Serial (Port) W/Timeout	49
Transmit Number (Port)	50
Transmit Number as Field (Port)	50
Transmit OPTOMUX Ack:	50
Transmit OPTOMUX Nak:	50
Transmit String Via Serial (Port)	50
Transmit String XON/XOFF (Port):	50
Transmit String XON/XOFF:	51
Transmit Table Via Serial (Port)	51
Transmit Time (Port)	51
Turn Sub Stepping Off	51
Turn Sub Stepping On	51

**U..... 52**

Until <	52
Until <=	52
Until <>	52
Until =	52
Until >	53
Until >=	53
Until AND	53
Until Chart Running	53
Until Chart Stopped	54
Until Chart Suspended	54
Until False	54
Until Off Latch	54
Until On Latch	54
Until OR	55
UNTIL Timer Expired	55
UNTIL Timer Expired OR	55
Until True	55

<b>W</b> .....	<b>56</b>
Write Byte to Controller Memory	56
Write Long to Controller Memory	56
Write String to PC Serial Port (ISA Only):	56
Write Word to Controller Memory	56

**A**

**AND - A & B & C**

Advanced Toolkit  
Logic Group

---

Arguments:	A
	B
	C
	Result

---

This command can be used in place of multiple commands and interim scratch variables. The logic within the command (in a pseudocode formula with the logical operations listed in blue) is "Result = A AND B AND C".

**AND - A & B & NOT C**

Advanced Toolkit  
Logic Group

---

Arguments:	A
	B
	C
	Result

---

This command can be used in place of multiple commands and interim scratch variables. The logic within the command (in a pseudocode formula with the logical operations listed in blue) is "Result = A AND B AND (NOT C)".

**AND - A & NOT B**

Advanced Toolkit  
Logic Group

---

Arguments:	A
	B
	Result

---

This command can be used in place of multiple commands and interim scratch variables. The logic within the command (in a pseudocode formula with the logical operations listed in blue) is "Result = (NOT A) AND (B)".

**AND - NOT A & NOT B**

Advanced Toolkit  
Logic Group

---

Arguments:	A
	B
	Result

---

This command can be used in place of multiple commands and interim scratch variables. The logic within the command (in a pseudocode formula with the logical operations listed in blue) is "Result = (NOT A) AND (NOT B)".

**Append LONG to String:**

Advanced Toolkit  
String Group

---

Arguments:	Value
	String

---

This command appends 4 bytes to a string representing the 32-bit value of the number exactly as it exists in memory, MSB first.. Usually it is used with the command "Get Long from String" as described on page 18

**Append Numeric Character as ASCII to String**

Advanced Toolkit  
String Group

Arguments: From  
To

This command is used to convert a single digit number to its corresponding ASCII character and place it into a string. For instance if the string contains "ABC" and the variable contains a value 2, the resultant string would be "ABC2".

**Append WORD to String**

Advanced Toolkit  
String Group

Arguments: Value  
String

This command appends 2 bytes to a string representing the lower 16-bits of the number as it exists in memory, MSB first. Max value appended is 65535 (FF FF). A value of 65536 is appended as two bytes (00 00).

**Are Any Bits Set in Table?:**

Standard Toolkit  
Logical Conditions Group

Arguments: Of Table

This command checks entire tables to determine whether any bits anywhere (in that table) are set. Returns true if so, false if not. Testing is stopped at the 1st non-zero element.

**Are No Bits Set in Table?:**

Standard Toolkit  
Logical Conditions Group

Arguments: Of Table

This command checks entire tables to determine whether all the bits anywhere (in that table) are cleared. Returns true if so, false if not. Testing is stopped at the 1st non-zero element.

**Average**

Advanced Toolkit  
Mathematical Group

Arguments: Average  
With  
Result

This command takes two values and averages them, returning the result.

**B**

<b>Bit AND - A &amp; B &amp; C</b>		Advanced Toolkit Logical Group
Arguments:	Bit AND A With B And C Put Result In	
This command operates the same as the command "AND A With B & C" only at a bit by bit level.		

<b>Bit Change:</b>		Standard Toolkit Logical Group
Arguments:	Control Bit Change	
This command is used to set or clear a bit in a variable based on the logical value of a control variable.		
'Control'	is the Boolean Set/Clear value	
'Bit'	is the bit to change	
'Change'	is the variable containing the bit to change	

<b>Bit Copy:</b>		Standard Toolkit Logical Group
Arguments:	Source Bit Destination Bit	
This command moves the state of a single bit in one variable to a bit in another variable without changing the rest of the bits in that result variable.		

<b>Bit Copy with Mask</b>		Standard Toolkit Logical Group
Arguments:	Source Mask Destination Put Result In	
This command takes a source variable and copies only the masked bits into the destination value. As an example, If the Source is 10101010, the Mask is 11110000, the Destination is 11001100, then the Result would be 10101100.		

<b>Bit Mirror:</b>		Standard Toolkit Logical Group
Arguments:	— Put Result In	
This command takes the 32 bits in a variable and reverses their order putting the result in a second variable. As an example, the bit mirror (8 bits) of a variable with a binary value of "0000101" is "10100000".		

**Bit Semaphore Reset:**

Standard Toolkit  
Logical Group

Arguments: <None>

This command clears the Bit Change lock semaphore and can be placed in the first block in the POWERUP chart and called once. *This command is no longer required as the semaphores now automatically unlock after a time-out period. It is included for those who wish to use it for good program initialization.*

When the IDAC West bit level commands are used in a program, a locking semaphore is set to make sure that the bit operand is not altered by another command simultaneously. While the semaphore is set, any other bit modification command will pause until it has been cleared. This semaphore is set and cleared within each command, but if a power loss occurs at just the right time (or just the wrong time, depending on your viewpoint) that semaphore can remain set.

**Break**

Standard Toolkit  
Logical Group

Arguments: <None>

This command is used to indicate the end of a Case option statement. It is only to be used within a Switch/Endswitch pair.

Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**C**

<b>Case &lt;</b>		Standard Toolkit Logical Group
	Arguments: _____	
	This command compares the value of the preceding "Switch" command's argument. If the value is less than a specific value, the code following this command and until the corresponding "Break" command is executed, the program then jumps to just past the 'EndSwitch' statement.	
	Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.	
<b>Case =</b>		Standard Toolkit Logical Group
	Arguments: _____	
	This command compares the value of the preceding "Switch" command's argument. If the value is equal to a specific value, the code following this command and until the corresponding "Break" command is executed, the program then jumps to just past the 'EndSwitch' statement.	
	Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.	
<b>Case &gt;</b>		Standard Toolkit Logical Group
	Arguments: _____	
	This command compares the value of the preceding "Switch" command's argument. If the value is less than a specific value, the code following this command and until the corresponding "Break" command is executed, the program then jumps to just past the 'EndSwitch' statement.	
	Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.	
<b>Case Within Limits:</b>		Advanced Toolkit Logical Group
	Arguments: >= Low Limit <= High Limit	
	This command compares the value of the preceding "Switch" command's argument. If the value is within or equal to specific high and low limits, the code following this command and until the corresponding "Break" command is executed, the program then jumps to just past the 'EndSwitch' statement.	
	Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.	

**Change Event Analog <=:**

Standard Toolkit  
Event/Reaction Group

Arguments: New Limit Event/Reaction Put Status In
This command will change the limit trigger for an analog event of the type: "Analog Input/Output <= Value" Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.

**Change Event Analog >=:**

Standard Toolkit  
Event/Reaction Group

Arguments: New Limit Event/Reaction Put Status In
This command will change the limit trigger for an analog event of the type: "Analog Input/Output >= Value" Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.

**Change Event Counter <=:**

Standard Toolkit  
Event/Reaction Group

Arguments: New Limit Event/Reaction Put Status In
This command will change the limit trigger for a digital event of the type: "Counter/Quadrature <= Value" Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.

**Change Event Counter >=:**

Standard Toolkit  
Event/Reaction Group

Arguments: New Limit Event/Reaction Put Status In
This command will change the limit trigger for a digital event of the type: "Counter/Quadrature >= Value" Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.

**Change Event Frequency <=:**

Standard Toolkit  
Event/Reaction Group

Arguments: New Limit Event/Reaction Put Status In
This command will change the limit trigger for a digital event of the type: "Frequency <= Value" Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.

**Change Event Frequency >=:**

Standard Toolkit  
Event/Reaction Group

Arguments: New Limit Event/Reaction Put Status In
This command will change the limit trigger for a digital event of the type: "Frequency >= Value" Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.

**Change Event MOMO::**

Standard Toolkit  
Event/Reaction Group

Arguments: New Must On New Must Off Event/Reaction Put Status In
This command will change the MOMO trigger for a digital event of the type: "MOMO Match" Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.

**Change Event Off Pulse <=:**

Standard Toolkit  
Event/Reaction Group

Arguments: New Limit Event/Reaction Put Status In
This command will change the limit trigger for a digital event of the type: "Off Pulse <= Value" Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.

**Change Event Off Pulse >=:**

Standard Toolkit  
Event/Reaction Group

<p>Arguments: New Limit Event/Reaction Put Status In</p>
<p>This command will change the limit trigger for a digital event of the type: "Off Pulse &gt;= Value" Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.</p>

**Change Event On Pulse <=:**

Standard Toolkit  
Event/Reaction Group

<p>Arguments: New Limit Event/Reaction Put Status In</p>
<p>This command will change the limit trigger for a digital event of the type: "On Pulse &lt;= Value" Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.</p>

**Change Event On Pulse >=:**

Standard Toolkit  
Event/Reaction Group

<p>Arguments: New Limit Event/Reaction Put Status In</p>
<p>This command will change the limit trigger for a digital event of the type: "On Pulse &gt;= Value" Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.</p>

**Change Event Totalizer <=:**

Standard Toolkit  
Event/Reaction Group

<p>Arguments: New Limit Event/Reaction Put Status In</p>
<p>This command will change the limit trigger for a digital event of the type: "Off/On Totalizer &lt;= Value" Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.</p>

**Change Event Totalizer >=:**

Standard Toolkit  
Event/Reaction Group

<p>Arguments: New Limit Event/Reaction Put Status In</p>
<p>This command will change the limit trigger for a digital event of the type: "Off/On Totalizer &gt;= Value" Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.</p>

**Change Reaction Analog Out:**

Standard Toolkit  
Event/Reaction Group

Arguments: New Analog Data  
Event/Reaction  
Put Status In

This command will change the setpoint for an analog E/R Reaction of the type "Set Analog Output"  
Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.

**Change Reaction Analog Ramp:**

Standard Toolkit  
Event/Reaction Group

Arguments: New Endpoint  
New Slope  
Event/Reaction  
Put Status In

This command will change the endpoint and slope for an analog E/R Reaction of the type "Ramp Analog Output to Endpoint"  
Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.

**Change Reaction MOMO:**

Standard Toolkit  
Event/Reaction Group

Arguments: New Must On  
New Must Off  
Event/Reaction  
Put Status In

This command will change the Output Mask for a digital E/R Reaction of the type "Set MOMO Outputs"  
Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.

**Change Reaction PID Setpoint:**

Standard Toolkit  
Event/Reaction Group

Arguments: New Setpoint  
Event/Reaction  
Put Status In

This command will change the setpoint for an analog E/R Reaction of the type "Set PID Setpoint"  
Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.

**Change Reaction Pulse Off:**

Standard Toolkit  
Event/Reaction Group

Arguments: New Pulse Data Event/Reaction Put Status In
This command will change the Pulse Time for a digital E/R Reaction of the type "Pulse Output Off" Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.

**Change Reaction Pulse On:**

Standard Toolkit  
Event/Reaction Group

Arguments: New Pulse Data Event/Reaction Put Status In
This command will change the Pulse Time for a digital E/R Reaction of the type "Pulse Output On" Once this command is issued, the program must issue an "Enable Scanning for Event" command before the new parameters are enabled. If interrupts are to occur on this event, an "Enable Interrupt on Event" command is also required.

**Characters Waiting (Port)?**

Standard Toolkit  
Communication - Serial Group

Arguments: <None>
This condition command returns true if there are any characters waiting to be read from the currently locked communications port. See page <b>Error! Bookmark not defined.</b> for a brief explanation of " <b>Error! Reference source not found.</b> ".

**Clear Receive Buffer (Port)**

Standard Toolkit  
Communication - Serial Group

Arguments: <None>
This command is used to clear all characters within the receive buffer of the currently locked port. See page <b>Error! Bookmark not defined.</b> for a brief explanation of " <b>Error! Reference source not found.</b> ".

**Convert 16-Bit to 32-Bit (signed):**

Advanced Toolkit  
Mathematical Group

Arguments: 16 Bit Integer
This command takes a value that has been stored as a signed 16-bit number (as received from an external device) and converts it to a signed 32-bit number (standard within the mistic controller).

**Convert 32-Bit to 16-Bit (signed):**

Advanced Toolkit  
Mathematical Group

Arguments: 32 Bit Integer
This command takes a value that has been stored as a signed 32-bit number (standard within the mistic controller) and converts it to a signed 16-bit number. Typically this will then be transmitted to another device that requires information in this format.

**Convert 8-Bit Gray Code to 360 Degrees:**

Advanced Toolkit  
Mathematical Group

Arguments: Gray Code Put Result In
This command takes a gray code value (0-FF hex) and returns the position of the encoder in degrees (0-359). Since 0 and 360 are the same point, the value returned will never read 360. Because the resolution of 8 bit gray code is not as great as the result, the value is rounded to the closest number.

**Convert Modem ASCII to Binary:**

Advanced Toolkit  
String Group

Arguments: _____
This command takes a string of characters in the Opto 22 Modem ASCII format (HEX) and converts it to a string of binary characters. Typically "Modem ASCII" is not readable by humans, but can be sent by even the simplest modem. This allows a controller to be used as an interpreter of the information coming from a Mystic Host Master when the ASCII protocol is used.

**Convert String Characters to Hex String:**

Advanced Toolkit  
String Group

Arguments: Source Delimiter Put Hex String In
This command converts every character in a string to 2 Hexadecimal characters plus an optional delimiter. The delimiter can be any single character string, or an empty string (""). This command is useful in debugging binary communications. Use to view binary strings in HEX. For example, the string "ABC HIJ", with a delimiter of a space character (ASCII 32), would be converted to "41 42 43 20 49 4A 4B".

**D**

***Decrement Table Variable:***

Standard Toolkit  
Mathematical Group

Arguments: Index Table	
This command is used in exactly the same manner as the equivalent variable commands, except it acts upon a table element, instead of a variable.	

***Delay False Exit***

Standard Toolkit  
Miscellaneous Condition Group

Arguments: MSec	
This is condition block command that is typically used to release the time slice for repetitive condition loops. If the condition block evaluates false, a delay will be initiated (in milliseconds) before exiting through the false connection to the next block.	
When a condition block loops back to itself (while waiting for a variable to be true), these commands can be used in place of a separate operation block that does nothing but delay for 1ms to release the time slice.	
Note: This command MUST be the last block in the statement.	

***Delay True Exit***

Standard Toolkit  
Miscellaneous Condition Group

Arguments: Msec	
This is condition block command that is typically used to release the time slice for repetitive condition loops. If the condition block evaluates true, a delay will be initiated (in milliseconds) before exiting through the True connection to the next block.	
When a condition block loops back to itself (while waiting for a variable to be true), these commands can be used in place of a separate operation block that does nothing but delay for 1ms to release the time slice.	
Note: This command MUST be the last block in the statement	

**E**

<b><i>Else..</i></b>		Standard Toolkit Logical Group
	<p>Arguments: &lt;None&gt;</p> <p>This command is an intermediate command that must be placed between an "If" command and an "Endif" command. Any code following the Else command (until the corresponding "Endif" is reached) will be executed if the corresponding "If" command has evaluated false.</p> <p>Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.</p>	
<b><i>EndIf</i></b>		Standard Toolkit Logical Group
	<p>Arguments: &lt;None&gt;</p> <p>This command ends the "If..[Else]..Endif" sequence. Each use of this command must be paired with an equivalent "If" command.</p> <p>Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.</p>	
<b><i>EndSwitch</i></b>		Standard Toolkit Logical Group
	<p>Arguments: &lt;None&gt;</p> <p>This command is used to indicate the end of a Switch Selection statement.</p> <p>Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.</p>	
<b><i>Exclusive Access?</i></b>		Standard Toolkit Logical Condition Group
	<p>Arguments: Semaphore</p> <p>This command requires the use of a semaphore variable. It attempts to get access to that variable exclusively and if it gets it, returns a true. If the access is not allowed (another task has it) a false decision is returned. The semaphore variable can be used to indicate that no other task should be accessing an area of data, port, etc.</p> <p>Exclusive Access can be released by using the "Release Exclusive Access" command or by setting the semaphore variable value to zero (false). If this chart is granted access, other charts checking this 'Semaphore' variable will find that it's not False and should wait until it becomes False.</p>	

**F**

**Flip Flop**

Advanced Toolkit  
Logical Group

<b>Arguments:</b>	Set Reset Output
<p>This command takes two arguments (SET and RESET) that determine the state of the result. It works in the same manner as a standard Latching Relay works. In pseudo-code the command operates as follows:</p> <pre> if SET = TRUE then OUTPUT = TRUE if RESET = TRUE then OUTPUT = FALSE if (SET = FALSE) and (RESET = FALSE) then OUTPUT = OUTPUT                     </pre>	

**Full Table Bit Clear:**

Standard Toolkit  
Logical Group

<b>Arguments:</b>	Clear Bit Table
<p>This command operates in the same manner as its OptoControl counterpart (Bit Clear) except that it operates on a bit numbered within an entire table. Bits in the table are numbered from 0 to <math>n</math>. Where <math>n = \text{Qty of Elements} \times 32</math>.</p>	

**Full Table Bit Set:**

Standard Toolkit  
Logical Group

<b>Arguments:</b>	Set Bit Table
<p>This command operates in the same manner as its OptoControl counterpart (Bit Set) except that it operates on a bit numbered within an entire table. Bits in the table are numbered from 0 to <math>n</math>. Where <math>n = \text{Qty of Elements} \times 32</math>.</p>	

**Full Table Bit Shift:**

Standard Toolkit  
Logical Group

<b>Arguments:</b>	Shift Amount Table Put Carry In
<p>This command operates in the same manner as its OptoControl counterpart (Bit Shift) except that it shifts all of the bits in the entire table left or right. The shift amount can be from <math>-32</math> to <math>+32</math>. The bits that are shifted past the end of the table (in either direction) are returned in the "Put Carry In" argument variable.</p>	

**Full Table Bit Test:**

Advanced Toolkit  
Logical Group

<b>Arguments:</b>	Test Bit Table Move To
<p>This command operates in the same manner as its OptoControl counterpart (Bit Test) except that it operates on a bit numbered within an entire table. Bits in the table are numbered from 0 to <math>n</math>. Where <math>n = \text{Qty of Elements} \times 32</math>.</p>	

**Full Table Bit Toggle:**

Standard Toolkit  
Logical Group

Arguments: Toggle Bit  
Table

This command operates in the same manner as its OptoControl counterpart (Bit Toggle) except that it operates on a bit numbered within an entire table. Bits in the table are numbered from 0 to  $n$ . Where  $n = \text{Qty of Elements} \times 32$ .

**Full Table Load (Float):**

Standard Toolkit  
Miscellaneous Group

Arguments: Value To Load  
Table

This command will completely fill a float table with a single, known value.

**Full Table Load (Integer):**

Standard Toolkit  
Miscellaneous Group

Arguments: Value To Load  
Table

This command will completely fill an integer table with a single, known value.

**Full Table Load (String):**

Standard Toolkit  
String Group

Arguments: Table  
String To Load

This command will completely fill a string table to a single, known value.

**G**

<b>Generate and Append HEX CRC16R to String:</b>		Advanced Toolkit String Group
Arguments:	String	
This command will calculate a Reverse CRC16 value on a string, convert that value to a four-character HEX string and then append that HEX string to the original string.		
<b>Generate Checksum on String (16bit):</b>		Advanced Toolkit String Group
Arguments:	String Put Result In	
This command will generate a standard 16bit checksum on a string and return the result.		
<b>Generate Forward CCITT on Character:</b>		Advanced Toolkit String Group
Arguments:	Character CRC Total	
This command is used to calculate a forward CCITT value on a series of characters in a sequence. The primary purpose of this command is to handle strings that are greater than the 127 character per string limit specified within OptoControl. The CRC total value can be preset to any required value as needed at the beginning of a CRC calculation (Typically 0 or -1).		
<b>Generate Reverse CRC-16 on Table:</b>		Advanced Toolkit Miscellaneous Group
Arguments:	Starting Index Table Put Result In	
This command takes a numeric table and a starting index and calculates a Reverse CRC-16 value based on the subsequent 32 elements in the table. Typical use is to ensure data integrity during a table transfer. Normally used with the 'Transmit Table via Serial', 'Receive Table via Serial', 'Transmit Long via Serial', and 'Receive Long via Serial' commands.		
<b>Get Analog Full Scale</b>		Standard Toolkit Analog Point Group
Arguments:	Analog Channel Put Result In	
This command returns the value specified for an analog point to read when it is at maximum setting. It is used to convert engineering units to a raw data (0-4095 for example).		
<b>Get Analog Zero Scale</b>		Standard Toolkit Analog Point Group
Arguments:	Analog Channel Put Result In	
This command returns the value specified for an analog point to read when it is at minimum setting. It is used to convert engineering units to a raw data (0-4095 for example).		

<b>Get ARCNET Error</b>	Advanced Toolkit Communication - Network Group
Arguments: Put Error In	
Used when receiving ARCnet messages via port 4. Use just before retrieving the message in port 4. This is the method for receiving the Arcnet error code. This command is only used for custom ARCnet transactions.	

<b>Get ARCNET Sequence Number</b>	Advanced Toolkit Communication – Network Group
Arguments: Put Result In	
Used when receiving ARCnet messages via port 4. If >=0, it is the message sequence number. If -1, no sequence number was sent.	

<b>Get Calling Chart Serial Number</b>	Standard Toolkit Chart Group
Arguments: Put Result In	
This operation is used to identify which chart actually called the current one. Serial numbers are assigned dynamically when the program is started.	

<b>Get Chart Serial Number</b>	Standard Toolkit Chart Group
Arguments: Chart Put Result In	
This operation returns the unique serial number of each chart. This is typically used with the command "Get Calling Chart Serial Number" to allow determination of the calling task. Serial numbers are assigned dynamically when the program is started.	

<b>Get Event/Reaction ID#</b>	Standard Toolkit Event/Reaction Group
Arguments: E/R Put Result In	
This command returns the number of the event/reaction as assigned to a brick, as displayed in the Event/Reaction configuration window.	

<b>Get Exclusive Access</b>	Standard Toolkit Logical Group
Arguments: Semaphore Put Result In	
This command attempts to lock a semaphore variable. The semaphore variable can be used to indicate that no other task should be accessing an area of data. If the access is granted the result variable will contain a value of -1 (True). Exclusive Access can be released by using the "Release Exclusive Access" command or by setting the semaphore variable value to zero (false).	

**Get I/O Point Channel and Port #**

Advanced Toolkit  
Communication – I/O Group

Arguments: I/O Point  
Put Channel In  
Put Port In

This command returns the I/O unit Channel and Port number for the specified I/O Point. This is typically used to dynamically modify a program’s behavior based on where a particular point is located.

**Get Long from String**

Advanced Toolkit  
String Group

Arguments: String  
Index  
Put Result In

This command reads four bytes from a location within a string and converts them Numeric Variable. This is typically used as a companion to the command “Append Long to String”.

**Get Name of Block Causing Current Error:**

Standard Toolkit  
Controller Group

Arguments: Put In

This command places the location (block name) of the block that caused the error on the top of the error stack. This information is the same as that shown in the Debug - Processor Error Information Window.

**Get Number of Characters Waiting (Port)**

Standard Toolkit  
Communication – Serial Group

Arguments: Put Result In

This command returns the actual count of characters that are waiting to be received in the currently locked port.

**Get Receive Buffer (Port):**

Advanced Toolkit  
Communication – Serial Group

Arguments: # of Characters  
Put In String

This command moves a number of characters within the currently locked communications port to a string. The command copies characters regardless of ASCII value.

The number of characters read can be less than requested for the following reasons:

1. The length of the target string is too short.
2. There aren't that many characters in the receive buffer.

Any characters not read from the buffer will remain in the buffer.

**Get Task # (0-31)**

Standard Toolkit  
Chart Group

Arguments: Result In

This operation provides the task ID number within the scheduling kernel.

**H**

**Hash Table Init:**

Standard Toolkit  
String Group

Arguments: String Table  
Hash Table  
Put Result In

Initializes the Integer Hash Table with the "Hash numbers". Assumes that the string lookup table has all entries in it. If more items are added to the table, this command should be issued again to reset the lookup table.

The integer Hash table size **must** be set to a prime number at least twice the size of the string lookup table.

Returns a 0 if all is well, -2 if the integer table is too short, -3 if the Integer Table size is not a Prime number.

**Hash Table Lookup:**

Standard Toolkit  
String Group

Arguments: String Table  
Hash Table  
String To Find  
Put Result In

Finds the first string table index where the string matches the contents of the string table exactly. Matches are case sensitive. Returns a -1 if no match.

**Hypotenuse**

Advanced Toolkit  
Mathematical Group

Arguments: A  
B  
Put Result In

This command calculates the hypotenuse of a right triangle given the length of the other two sides.

This command is equivalent to  $R = \sqrt{a^2 + b^2}$

<b>!</b>		
<b>If AND</b>		Advanced Toolkit Logical Group

Arguments: If And	
This command executes a logical AND of two values and determines if the result is true (non-zero). If so, the true portion of the "If..[else]..Endif" code is executed.	
Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.	

<b>If Bit</b>		Standard Toolkit Logical Group
---------------	--	-----------------------------------

Arguments: Data Source Bit To Test	
This command checks to see if a bit is set. If so, the true portion of the "If..[else]..Endif" code is executed.	
Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.	

<b>If Changed</b>		Standard Toolkit Logical Group
-------------------	--	-----------------------------------

Arguments: Compare Reference Variable	
This command compares a value with a stored reference. If the value has changed since the last compare, the true portion of the if/endif code is executed, and the new value is copied to the reference variable.	
Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.	

<b>If Chart Running</b>		Standard Toolkit Chart Group
-------------------------	--	---------------------------------

Arguments: Chart	
This command checks to see if a chart is running. If so, the true portion of the "If..[else]..Endif" code is executed.	
Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.	

<b>If Chart Stopped</b>		Standard Toolkit Chart Group
-------------------------	--	---------------------------------

Arguments: Chart	
This command checks to see if a chart is stopped. If so, the true portion of the "If..[else]..Endif" code is executed.	
Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.	

***If Chart Suspended***

Standard Toolkit  
Chart Group

Arguments: Chart

This command checks to see if a chart is suspended. If so, the true portion of the "If..[else]..Endif" code is executed.

Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

***If Equal***

Standard Toolkit  
Logical Group

Arguments: If  
Is Equal To

This command checks to see if two values are equal. If so, the true portion of the "If..[else]..Endif" code is executed.

Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

***If Equal AND***

Advanced Toolkit  
Logical Group

Arguments: If  
Is Equal To  
AND

This command checks to see if two values are equal and if another object contains a non-zero (true) value. If so, the true portion of the "If..[else]..Endif" code is executed.

Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

***If Equal And NOT***

Advanced Toolkit  
Logical Group

Arguments: If  
Is Equal To  
And NOT

This command checks to see if two values are equal and if another object contains a zero (false) value. If so, the true portion of the "If..[else]..Endif" code is executed.

Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

***If Equal OR***

Advanced Toolkit  
Logical Group

Arguments: If  
Is Equal To  
OR

This command checks to see if two values are equal OR if another object contains a non-zero (true) value. If either is so, the true portion of the "If..[else]..Endif" code is executed.

Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**If Equal Strings**

Standard Toolkit  
String Group

Arguments: _____ With
This command checks to see if two strings are equal (both case and length sensitive). If so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**If Event Occurring**

Standard Toolkit  
Logical Group

Arguments: Event/Reaction
This command checks to see if an event/reaction is currently occurring. If so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**If Exclusive Access**

Standard Toolkit  
Logical Group

Arguments: Semaphore
This command requires the use of a semaphore variable. It attempts to get access to that variable exclusively and if it gets it, the true portion of the "If..[Else]..Endif" code is executed.. If the access is not allowed (another task has it) the false portion of the code (if present) is executed. Exclusive Access can be released by using the "Release Exclusive Access" command or by setting the semaphore variable value to zero (false). Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**If False**

Standard Toolkit  
Logical Group

Arguments: _____
This command checks to see if an object contains a zero (false) value. If so, the true portion of the "If..[Else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**If Generating Interrupt**

Standard Toolkit  
Event/Reaction Group

Arguments: _____
This command checks to see if an I/O unit is currently generating an interrupt. If so, the true portion of the "If..[Else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

***If Greater***

Standard Toolkit  
Logical Group

---

Arguments: If Is Greater Than
This command checks to see if the value of the object in the first argument is greater than the value of the object in the second argument. If so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

***If Greater AND***

Advanced Toolkit  
Logical Group

---

Arguments: If Is Greater Than AND
This command checks to see if the value of the object in the first argument is greater than the value of the object in the second argument and the third object contains a true (non-zero) value. If so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

***If Greater OR***

Advanced Toolkit  
Logical Group

---

Arguments: If Is Greater Than OR
This command checks to see if the value of the object in the first argument is greater than the value of the object in the second argument or the third object contains a true (non-zero) value. If either is so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

***If Greater or Equal***

Standard Toolkit  
Logical Group

---

Arguments: If Is >= to
This command checks to see if the value of the object in the first argument is greater or equal to the value of the object in the second argument. If so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

***If I/O Communication Enabled***

Standard Toolkit  
I/O Unit Group

---

Arguments: _____
This command checks to see if the I/O Point, Unit (Brick), PID, E/R, etc., is enabled. If so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**If Less**

Standard Toolkit  
Logical Group

Arguments: If Is Less Than
This command checks to see if the value of the object in the first argument is less than the value of the object in the second argument. If so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**If Less or Equal**

Standard Toolkit  
Logical Group

Arguments: If Is <= to
This command checks to see if the value of the object in the first argument is less than or equal to the value of the object in the second argument. If so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**If OR**

Advanced Toolkit  
Logical Group

Arguments: If OR
This command checks to see if the value of either of two objects is true (non-zero). If so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**If Table Elements Equal**

Advanced Toolkit  
Logical Group

Arguments: Index Table Index Table
This command checks to see if the values of two table elements are the same. If so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**If Timer Expired**

Standard Toolkit  
Logical Group

Arguments: Timer
This command checks to see if a timer has expired. If so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

***If Timer Expired AND***

Advanced Toolkit  
Logical Group

Arguments: Timer AND
This command checks to see if a timer has expired and the value of another object is non-zero (true). If so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

***If Timer Expired And NOT***

Advanced Toolkit  
Logical Group

Arguments: Timer And NOT
This command checks to see if a timer has expired and the value of another object is zero (false). If so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

***If Timer Expired OR***

Advanced Toolkit  
Logical Group

Arguments: Timer OR
This command checks to see if a timer has expired or the value of another object is non-zero (true). If either is so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

***If True***

Standard Toolkit  
Logical Group

Arguments: _____
This command checks to see if the value of an object is non-zero (true). If so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

***If Within Limits***

Advanced Toolkit  
Logical Group

Arguments: Test Value >= Low Limit <= High Limit
This command checks to see if an object has a value less than or equal to a high limit and greater than or equal to a low limit. If so, the true portion of the "If..[else]..Endif" code is executed. Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

<b>Increment Table Variable:</b>	Standard Toolkit Mathematical Group
Arguments: Index Table	
This command is used in exactly the same manner as the equivalent variable commands, except it acts upon a table element, instead of a variable.	

<b>Increment Variable 2x</b>	Advanced Toolkit Mathematical Group
Arguments: _____	
This operation works the same as the "Increment Variable" command, except that 2 is added to the value of the variable instead of just 1. This command executes faster than the corresponding "Add" command.	

<b>Increment Variable 4x</b>	Advanced Toolkit Mathematical Group
Arguments: _____	
This operation works the same as the "Increment Variable" command, except that 4 is added to the value of the variable instead of just 1. This command executes faster than the corresponding "Add" command.	

<b>Integer Table Lookup:</b>	Standard Toolkit Miscellaneous Group
Arguments: Value to Find Table Put Result In	
This command is used to look through an integer table for a particular value. If it is found, the lowest index number that contains the value is placed in the Result variable. If there is no match in the entire table, a value of -1 is placed in the Result variable.	

<b>Invert Previous Conditions</b>	Standard Toolkit Miscellaneous Group
Arguments:	
This condition takes the result of any conditions (in the same condition block) previously solved and inverts the result. Subsequent commands in the same block will be ANDed with the opposite of the logic thus far determined. This allows combinations of AND logic not before possible with the existing commands.	
<p style="margin-left: 40px;">Example :     Timer Expired?           <i>Timer_A</i>                              Invert Previous Conditions                              Timer Expired?           <i>Timer_B</i></p> <p style="margin-left: 40px;">This example would prove true only when Timer_A was NOT expired Timer_B was.</p>	

<b>I/O Communication Enabled?</b>	Standard Toolkit I/O Unit Condition Group
Arguments: I/O	
This condition returns a true result if the specified I/O Point, Unit (Brick), PID, E/R, etc., currently has communication enabled.	

<b>Is Bit Clear in Table?:</b>		Standard Toolkit
		Logical Condition Group
Arguments:	Test Bit Of Table	
This command checks an entire table to determine if a specific bit is clear. Bits in the table are numbered from 0 to <i>n</i> . Where $n = \text{Qty of Elements} \times 32$ .		

<b>Is Bit Set in Table?:</b>		Standard Toolkit
		Logical Condition Group
Arguments:	Test Bit Of Table	
This command checks an entire table to determine if a specific bit is set. Bits in the table are numbered from 0 to <i>n</i> . Where $n = \text{Qty of Elements} \times 32$ .		

<b>Is Event NOT Occurring?</b>		Advanced Toolkit
		Event/Reaction Condition Group
Arguments:	Event/Reaction	
This command checks to see if an event is not currently occurring. It is equivalent to the "Is Event Occurring" command with the opposite response.		

<b>IVAL If ON</b>		Advanced Toolkit
		Simulation Group
Arguments:	If	
This command checks to see if the IVAL of a digital point is on (true). If so, the true portion of the "If..[else]..Endif" code is executed.		
Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.		

<b>IVAL Turn On A Latch</b>		Advanced Toolkit
		Simulation Group
Arguments:	_____	
This command is used to activate an input point latch as if an off-to-on transition had occurred in the simulation mode (IVAL).		

<b>IVAL Write Digital from Binary Table Value</b>		Advanced Toolkit
		Simulation Group
Arguments:	I/O Unit Index Table	
This command is used to preset a group of input and output points' Internal Values when they are not enabled. The command takes a value from an element in a table and uses the lower 16 bits to turn on or off the IVAL's of the points on the I/O Unit.		

***IVAL Write Digital from Binary Value***

Advanced Toolkit  
Simulation Group

---

Arguments: Value  
I/O Unit

---

This command is used to preset a group of input and output points' Internal Values when they are not enabled. The command takes a value and uses the lower 16 bits to turn on or off the IVAL's of the points on the I/O Unit.

**L**

<b>Latch Not Set?</b>		Advanced Toolkit
		Digital Point Condition Group
Arguments:	Latch	
This condition returns true if a latch on an input point has NOT been set yet.		
<b>Latch Set OR?</b>		Advanced Toolkit
		Digital Point Condition Group
Arguments:	Value Latch	
This condition returns true if a latch is set or a variable contains a true (non-zero) value.		
<b>Link Float Variable to Table:</b>		Standard Toolkit
		Miscellaneous Group
Arguments:	Float Var Link To Index In Table	
This command links a float variable to an element in a float table. For an explanation of linking see page <b>Error! Bookmark not defined.</b>		
<b>Link Integer Variable to Table:</b>		Standard Toolkit
		Miscellaneous Group
Arguments:	Integer Var Link To Index In Table	
This command links an integer variable to an element in an integer table. For an explanation of linking see page <b>Error! Bookmark not defined.</b>		
<b>Link String Variable to Table:</b>		Advanced Toolkit
		Miscellaneous Group
Arguments:	String Var Link To Index In Table	
This command links a string variable to an element in a string table.		
<b>Link SubTable to Table:</b>		Advanced Toolkit
		Miscellaneous Group
Arguments:	Sub Table Master Table Master Table Index	
This command links one table to a portion of a second table.		
The term "SUB-TABLE" is used to define a table that is linked to a portion of another table. As an example: Table A has a length of 20 elements, Table B (with a length of 10 elements) is linked to A at element 4. Thus, Table B – element 0 is the same as Table A - element 4. This allows multiple tables to be linked into the same master table at different indexes.		

***Loop...Until***

Advanced Toolkit

Logical Group

---

Arguments: <None>

---

This command marks the beginning of an in-block looping sequence. All code between this command and the corresponding "Until" command will be executed continuously until the "Until" command allows completion.

Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**M**

<b>Move Analog Counts to Table</b>	Advanced Toolkit I/O Unit Group
Arguments: Index Table Board	
This command reads an analog I/O unit and returns the raw Input and Output values (before scaling) and places them into a numeric table. Typical Range is 0-4095. However, due to module overrange capability, readings of 8192 or more can be seen. Underrange values can also exist, (to -32768)	

<b>Move Digital I/O Unit to Table</b>	Advanced Toolkit I/O Unit Group
Arguments: I/O Unit Index Table	
This command reads a digital I/O unit and places the binary value of the digital points into a numeric table, in the lower 16 bits of the specified element.	

<b>Move from FIFO Table:</b>	Standard Toolkit String Group
Arguments: Table To	
This command pulls the first element (0) of a numeric table and places it into a result variable, shifting all of the other values down (toward 0) by one index. The table size determines the number of shifted values that can be stored. This actually creates a LIFO (Last-In, First-Out) table if used with the "Move To" commands.	

<b>Move From Float Array:</b>	Advanced Toolkit Miscellaneous Group
Arguments: Row Column Float Table Put Result In	
This command gets information from a two-dimensional floating point array (created using the "Set Column Count in Array:" command) by specifying the row and column of the item to be retrieved.	

<b>Move From Integer Array:</b>	Advanced Toolkit Miscellaneous Group
Arguments: Row Column Integer Table Put Result In	
This command gets information from a two-dimensional integer array (created using the "Set Column Count in Array:" command) by specifying the row and column of the item to be retrieved.	

**Move From String Array:**

Advanced Toolkit  
String Group

Arguments: Row (0-n) Column (0-n) Array Alias To String
This command retrieves information from a two dimensional string array. The array is created using the "Set String Table into String Array Column" command.

**Move from String FIFO Table:**

Standard Toolkit  
String Group

Arguments: Table To
This command pulls the first element (0) of a string table and places it into a result variable, shifting all of the other values down (toward 0) by one index. The table size determines the number of shifted values that can be stored.

**Move String Table to String Table**

Standard Toolkit  
String Group

Arguments: Index Table To Index To Table
This command moves an element of one string table directly to an element in another without the use of an intermediate temporary scratch string variable.

**Move Table to Table:**

Standard Toolkit  
Miscellaneous Group

Arguments: Source Table Dest Table Dest Index How Many
This command moves any number of elements of one table, starting at element 0 to another table at a (potentially) different starting index.

**Move Table to Table (16-Bit):**

Advanced Toolkit  
Miscellaneous Group

Arguments: Source Table Dest Table Dest Index How Many
This command copies the lower 16 bits of a number of table elements to another table, leaving the upper 16 unchanged.

**Move to Float Array:**

Advanced Toolkit  
Miscellaneous Group

Arguments: Value Row Column Float Table
This command places information into a two-dimensional floating point array (created using the "Set Column Count in Array:" command) by specifying the row and column of the destination for the value.

**Move to Float FIFO Table:**

Standard Toolkit  
Miscellaneous Group

Arguments: Input Value Table Put Output In
This command shifts the values in a Float table up by one index, adding the new value to element 0 and placing the last value into the result variable. (This is called a First-In, First-Out (FIFO) table). The table size determines the number of shifted values that can be stored.

**Move to Integer Array:**

Advanced Toolkit  
Miscellaneous Group

Arguments: Value Row Column Integer Table
This command places information into a two-dimensional integer array (created using the "Set Column Count in Array:" command) by specifying the row and column of the destination for the value.

**Move to Integer FIFO Table:**

Standard Toolkit  
Miscellaneous Group

Arguments: Input Value Table Put Output In
This command shifts the values in an Integer table up by one index, adding the new value to element 0 and placing the last value into the result variable. (This is called a First-In, First-Out (FIFO) table). The table size determines the number of shifted values that can be stored.

**Move To String Array:**

Advanced Toolkit  
String Group

Arguments: String Row (0-n) Column (0-n) Array Alias
This command stores information into a two dimensional string array. The array is created using the "Set String Table into String Array Column" command.

**Move to String FIFO Table:**

Standard Toolkit  
String Group

Arguments: Input String Table Put Output In
This command shifts the values in a string table up by one index, adding the new value to element 0 and placing the last value into the result variable. (This is called a First-In, First-Out (FIFO) table). The table size determines the number of shifted values that can be stored.

**Move with No Conversion (32bit Copy)**

Advanced Toolkit  
Miscellaneous Group

Arguments: From To
This command copies numeric values without any conversion. Floating point values and integer values are normally converted to the destination's format before moving. This command actually copies these values, bit-by-bit, from one to the other. Since floating point values are stored in a completely different format than integer values, this copy can make numbers unreadable and unusable if used when not desired.

**Move Word to Table**

Advanced Toolkit  
Miscellaneous Group

Arguments: Index Table LSB MSB
This command takes two values (as LSB and MSB bytes) and converts them to a single, 16-bit unsigned, value and stores them in a numeric table.

**N**

***Not Variable***

Advanced Toolkit  
Logical Group

---

Arguments: \_\_\_\_\_

---

This command is equivalent to the pseudocode formula "A = NOT A".

**O**

<b>OR - A or B or C</b>	Advanced Toolkit Logical Group
Arguments: A B C Put Result In	
This command can be used in place of multiple commands and interim scratch variables. The logic within the command (in a pseudocode formula with the logical operations listed in blue) is "Result = A <b>OR B OR C</b> "	

<b>OR - A or B or NOT C</b>	Advanced Toolkit Logical Group
Arguments: A B C Put Result In	
This command can be used in place of multiple commands and interim scratch variables. The logic within the command (in a pseudocode formula with the logical operations listed in blue) is "Result = A <b>OR B OR (NOT C)</b> ".	

<b>OR - A or NOT B</b>	Advanced Toolkit Logical Group
Arguments: A B Put Result In	
This command can be used in place of multiple commands and interim scratch variables. The logic within the command (in a pseudocode formula with the logical operations listed in blue) is "Result = A <b>OR (NOT B)</b> ".	

<b>OR - NOT A or NOT B</b>	Advanced Toolkit Logical Group
Arguments: A B Put Result In	
This command can be used in place of multiple commands and interim scratch variables. The logic within the command (in a pseudocode formula with the logical operations listed in blue) is "Result = <b>(NOT A) OR (NOT B)</b> ".	

<b>OR??</b>	Advanced Toolkit Logical Condition Group
Arguments: Is OR OR OR	
This Condition allows up to four variables to be OR'd together. The condition is true if any of the variables are true. If one of the variables is not needed, place a literal zero (0) in that argument.	

**OR Block Begin – OR Previous With...**

Advanced Toolkit  
Logical Condition Group

Arguments: <None>										
<p>This auxiliary condition statement allows conditions within the same <i>Condition</i> block to work in an OR fashion. Any logical true/false exit condition created by the statements prior to this command are OR'd with the true/false condition created by statements within the block.</p> <p>All conditions within a block are "ANDed" with all of the others, and (therefore) must all be true to have a "True" exit. This statement (with its corresponding "Block End" statement) act as an OR statement with parentheses.</p> <p>Example: If DigIn0 is currently Off, DigIn1 is currently On, AnaIn currently is at 60, the following conditions will exit as "True".</p> <table style="margin-left: 40px;"> <tr> <td>Is → DigIn0 → ON?</td> <td>False</td> </tr> <tr> <td>OR Block Begin – Or Previous With</td> <td>OR (</td> </tr> <tr> <td>Is → DigIn1 → ON?</td> <td>TRUE</td> </tr> <tr> <td>Is → AnaIn → Greater Than? → 50</td> <td>TRUE</td> </tr> <tr> <td>OR Block End)</td> <td></td> </tr> </table> <p>In the preceding example, the determination of true/false exit can be written (as taken from the right hand column) as "Result = False OR ( True AND True)".</p> <p><b>Warning:</b> These commands can only be used as matched pairs in a "Block Begin/Block End" sequence. They must only be used within the same condition block as the corresponding "OR Block End" statement. Failure to follow these rules, or an unfinished pair may cause a fault during download or may cause program malfunction.</p>	Is → DigIn0 → ON?	False	OR Block Begin – Or Previous With	OR (	Is → DigIn1 → ON?	TRUE	Is → AnaIn → Greater Than? → 50	TRUE	OR Block End)	
Is → DigIn0 → ON?	False									
OR Block Begin – Or Previous With	OR (									
Is → DigIn1 → ON?	TRUE									
Is → AnaIn → Greater Than? → 50	TRUE									
OR Block End)										

**OR Block End**

Advanced Toolkit  
Logical Condition Group

Arguments: <None>
<p>This auxiliary condition statement completes the "OR Block Begin" statement described above. It can only be used with that command. For a description of the purpose of this command, see above.</p> <p><b>Warning:</b> These commands can only be used as matched pairs in a "Block Begin/Block End" sequence. They must only be used within the same condition block as corresponding "OR Block End" statement. Failure to follow these rules, or an unfinished pair may cause a fault during download or may cause program malfunction.</p>

**P**

<b><i>Put Port into ASCII I/O Master Mode:</i></b>	Advanced Toolkit Communication – I/O Group
Arguments: Port# (0-3)	
This command sets an I/O port into the 8-bit ASCII CRC mode that can be used by the I/O units if jumpers are set appropriately. This mode is useful for transmission over a modem. OptoControl allows the determination of this mode at configuration time, this command can be used during execution time.	

<b><i>Put Port into Binary I/O Master Mode:</i></b>	Advanced Toolkit Communication – I/O Group
Arguments: Port# (0-3)	
This command sets an I/O port into the 9 Bit Binary CRC mode that can be used by the I/O units if standard jumper settings are used. This mode is useful for transmission over a modem. OptoControl allows the determination of this mode at configuration time, this command can be used during execution time.	

<b><i>Put Port into Binary I/O Slave Mode:</i></b>	Advanced Toolkit Communication – I/O Group
Arguments: Port# (0-3)	
This command sets an I/O port into 9-bit binary slave mode for communication as a Remote Bricks to a mistic controller. There are no additional commands relating to emulating a brick and it is assumed that the programmer will be writing serial communications routines based on the Brick protocol manual.	

<b><i>Put Port into Normal (Non-I/O) Mode:</i></b>	Advanced Toolkit Communication – I/O Group
Arguments: Port# (0-3)	
This command removes the I/O communication driver from a serial port and allows it to be used for normal communication. Unless I/O is assigned to a port during configuration, this is the default setting for all ports when the program is run.	

<b><i>Put Substring:</i></b>	Standard Toolkit String Group
Arguments: Source Put At In string Fill Character	
This command works, as you would expect it to, in the inverse manner as the "Get Substring" command supplied with OptoControl.	
You specify where in the destination string that you want the sub-string to begin and the sub-string is placed there. If the string variable is not long enough to contain the sub-string, as much as will fit is moved (similar to a "Move String" command). If the destination location is at a location beyond the end of the existing string, "Fill Characters" are used to fill between the end of the pre-existing string and the sub-string's beginning.	

**R**

<b>Read Byte from Controller Memory</b>		Advanced Toolkit Controller Group															
Arguments: Address Put Result In																	
This command reads one byte (8 bits) from a location in the controller's memory. <b>Warning:</b> Direct memory manipulation commands, such as this command, should be used with caution. Monitoring or modifying certain memory locations can cause unexpected operation and program failure.																	
<b>Read Hi-Density Brick Temperature</b>		Advanced Toolkit I/O Unit Group															
Arguments: From Move To																	
This command returns the current temperature of the Hi-Density brick specified as determined by the brick's on board sensor.																	
<b>Read Long from Controller Memory</b>		Advanced Toolkit Controller Group															
Arguments: Address Put Result In																	
This command reads four bytes (32 bits) from a location in the controller's memory. <b>Warning:</b> Direct memory manipulation commands, such as this command, should be used with caution. Monitoring or modifying certain memory locations can cause unexpected operation and program failure.																	
<b>Read PID Parameter:</b>		Advanced Toolkit PID Group															
Arguments: PID Loop Param# (0-15) Put Value In Put Status In																	
This command will return the desired PID parameter. A return status of zero(0) means a successful read occurred. There are 14 parameters available.																	
<table border="0"> <tr> <td>0:(Int) Control Word</td> <td>5:(Flt) Setpoint</td> <td>10:(Flt) Setpoint limit max</td> </tr> <tr> <td>1:(Int) Scan rate counts</td> <td>6:(Flt) Output</td> <td>11:(Flt) Setpoint limit min</td> </tr> <tr> <td>2:(Int) Output in counts</td> <td>7:(Flt) Gain</td> <td>12:(Flt) Output limit max</td> </tr> <tr> <td>3:(Int) IN,Setp,Out Ch#</td> <td>8:(Flt) Integral</td> <td>13:(Flt) Output limit min</td> </tr> <tr> <td>4:(Flt) Input</td> <td>9:(Flt) Derivative</td> <td></td> </tr> </table>		0:(Int) Control Word	5:(Flt) Setpoint	10:(Flt) Setpoint limit max	1:(Int) Scan rate counts	6:(Flt) Output	11:(Flt) Setpoint limit min	2:(Int) Output in counts	7:(Flt) Gain	12:(Flt) Output limit max	3:(Int) IN,Setp,Out Ch#	8:(Flt) Integral	13:(Flt) Output limit min	4:(Flt) Input	9:(Flt) Derivative		
0:(Int) Control Word	5:(Flt) Setpoint	10:(Flt) Setpoint limit max															
1:(Int) Scan rate counts	6:(Flt) Output	11:(Flt) Setpoint limit min															
2:(Int) Output in counts	7:(Flt) Gain	12:(Flt) Output limit max															
3:(Int) IN,Setp,Out Ch#	8:(Flt) Integral	13:(Flt) Output limit min															
4:(Flt) Input	9:(Flt) Derivative																
<b>NOTE:</b> Be sure to select the proper Variable TYPE to put the value in or it will be meaningless!																	

**Read Word from Controller Memory**

Advanced Toolkit  
Controller Group

Arguments: Address  
Put Result In

This command reads two bytes (16 bits) from a location in the controllers memory.

**Warning:** Direct memory manipulation commands, such as this command, should be used with caution. Monitoring or modifying certain memory locations can cause unexpected operation and program failure.

**Receive Character Via Serial (Port)**

Standard Toolkit  
Communication - Serial Group

Arguments: Put Result In

This command will get one character from the currently locked port. See page **Error! Bookmark not defined.** for a brief explanation of "**Error! Reference source not found.**". This command will wait indefinitely until a character appears.

**Receive Long from (Port) 1<sup>st</sup>=MSB:**

Advanced Toolkit  
Communication – Serial Group

Arguments: Move To

This command will receive four bytes from the currently locked communications port and will convert them into a long integer value. It assumes that the bytes are arriving at the port with the most significant first and the least significant last. This command will wait indefinitely for all four characters to appear.

**Receive Long from (Port) 4<sup>th</sup>=MSB:**

Advanced Toolkit  
Communication - Serial Group

Arguments: Move To

This command will receive four bytes from the currently locked communications port and will convert them into a long integer value. It assumes that the bytes are arriving at the port with the least significant first and the most significant last. This command will wait indefinitely for all four characters to appear.

**Receive Long from Port 1<sup>st</sup>=MSB:**

Advanced Toolkit  
Communication - Serial Group

Arguments: Port# (0-3)  
Move To

This command will receive four bytes from a communications port and will convert them into a long integer value. It assumes that the bytes are arriving at the port with the most significant first and the least significant last.

**Receive Long from Port 4<sup>th</sup>=MSB:**

Advanced Toolkit  
Communication - Serial Group

Arguments: Port# (0-3)  
Move To

This command will receive four bytes from a communications port and will convert them into a long integer value. It assumes that the bytes are arriving at the port with the most significant first and the least significant last.

**Receive Modem ASCII as Binary String via Serial Port:**

Advanced Toolkit  
Communication - Serial Group

Arguments: Put In String  
Use Port  
Put Result In

This command receives (from a serial port) a string of characters that are in the Opto 22 Modem ASCII format (HEX) and converts it to a string of binary characters. Typically "Modem ASCII" is not readable by humans, but can be sent by even the simplest modem. This allows a controller to be used as an interpreter of the information coming from a Mystic Host Master when the ASCII protocol is used.

**Receive OPTOMUX Msg/Addr:**

Advanced Toolkit  
Communication - I/O Group

Arguments: Address  
Port  
Message  
Result

This command receives a string that is formatted as an OptoMux message through the selected port. It then returns the status, intended address, and message body. If the checksum fails, a non-zero result will occur. When used with the "Transmit Optomux Ack/Nak" commands, this allows the controller to act as an OptoMux slave device.

**Receive OPTOMUX Msg:**

Advanced Toolkit  
Communication - I/O Group

Arguments: Address  
Port  
Message  
Result

This command receives a string that is formatted as an OptoMux message over the selected port, whose address also matches the requested address. If a message is returned but the checksum fails, a non-zero result will occur. This command will not return until a message is received for the appropriate address. When used with the "Transmit Optomux Ack/Nak" commands, this allows the controller to act as an OptoMux slave device.

**Receive String Via Serial (Port)**

Standard Toolkit  
Communication - Serial Group

Arguments: Put Result In

This command will receive a string from the previously locked serial port. The command works in the same manner as the "Receive String Via Serial Port" command within OptoControl. This command will wait indefinitely for a carriage return or the number of characters equal to the length of the target string to appear.

**Receive Table Via Serial (Port)**

Standard Toolkit  
Communication - Serial Group

Arguments: Index  
Table

This command will receive 32 elements into a numeric table from the previously locked serial port. The command works in the same manner as the "Receive Table Via Serial Port" command within OptoControl. This command will wait indefinitely for all 128 bytes to appear.

**Release Active Port**

Standard Toolkit  
Communication - Serial Group

Arguments: <None>

This command is used to release a communications port that has been previously locked to a chart using the "Request Port" command.

**Release Exclusive Access**

Standard Toolkit  
Logical Group

Arguments: Semaphore

This command clears a semaphore variable to allow another task exclusive access to that variable. The exclusive access is granted through any command that checks or requests it.

**Request Port**

Standard Toolkit  
Communication - Serial Group

Arguments: Port Number  
Put Status In

This command is used to lock a communications port for use by a single chart. The parameters are the port number and a result variable. If the port is in use by another chart, or is unavailable for any other reason, the result is returned as a "False" or zero value.

**S**

<b>Set ARCNET Return Error</b>		Advanced Toolkit
		Communication - Network Group
Arguments: Return Error		
Used when sending ARCnet messages via port 4. Use before the "Print NewLine to Port" command. This is the method for passing error codes via Arcnet. This command is only used for custom ARCnet transactions.		
<b>Set ARCNET Sequence Number</b>		Advanced Toolkit
		Communication - Network Group
Arguments: To		
Used when sending ARCnet messages via port 4. Use to send a sequence id number from 0-127. -1 = no sequence ID#.		
<b>Set Column Count in Array:</b>		Advanced Toolkit
		Miscellaneous Group
Arguments: # of Columns Table		
This command initializes a table to be handled as a two-dimensional array. Once this command is issued, the other array commands can be used for placing and retrieving information.		
<b>Set Digital I/O Unit from Binary Table Value</b>		Standard Toolkit
		I/O Unit Group
Arguments: I/O Unit Index Table		
This command will set up to 16 digital outputs in a single I/O unit based on the binary value of an element in a numeric table.		
<b>Set String Table into String Array Column</b>		Advanced Toolkit
		String Group
Arguments: String Table Column (0-n) Array Alias		
This command initializes an integer table (the alias) to have it become a link into a 2 dimensional string array. The programmer assigns normal string tables to each column of the array using this command, and then stores and retrieves information from the array using the "Move To/From String Array" commands.		
Tables can be any width, but should all be the same length. Size the array alias table equal to or greater than the number of columns.		
<b>Set Table Element False</b>		Standard Toolkit
		Logical Group
Arguments: Index Table		
This command is used in exactly the same manner as the equivalent variable commands, except it acts upon a table element, instead of a variable.		

**Set Table Element True**

Standard Toolkit

Logical Group

Arguments: Index  
Table

This command is used in exactly the same manner as the equivalent variable commands, except it acts upon a table element, instead of a variable.

**Start PID Averaging:**

Advanced Toolkit

PID Group

Arguments: PID Loop  
Put Result In

This command allows the PID loop to use the averaged (filtered) value of the input as its process variable. Analog averaging must have been activated previously for that channel.

**Start Terminal Task:**

Advanced Toolkit

Chart Group

Arguments: Put Result In

This command is used to start the terminal task supplied by Opto 22 for use on the G4LC32 Classic Controller (file: mystic.trm). This task converts the front panel display and keypad on that controller to a parameter access panel. This task was supplied with Cyrano and is still useable with OptoControl.

**Stop PID Averaging:**

Advanced Toolkit

PID Group

Arguments: PID Loop  
Put Result In

This command tells the PID to use the current value of the PID INPUT CHANNEL, rather than the filtered value.

**Stop Terminal Task:**

Advanced Toolkit

Chart Group

Arguments: Put Result In

This command is used to Stop the terminal task supplied by Opto 22 for use on the G4LC32 Classic Controller (file: mystic.trm). This task converts the front panel display and keypad on that controller to a parameter access panel. This task was supplied with Cyrano and is still useable with OptoControl.

**String Table Lookup:**

Standard Toolkit

String Group

Arguments: Search For  
In Table  
Put Result In

This command just looks through a string table for a string. If an exact match (case and length sensitive) is found, the index is placed in the result variable. If there is no match, a -1 is placed in the result variable.

**String Table Lookup SubString:**

Advanced Toolkit  
String Group

<p>Arguments: String Table SubString to Find Start Index Put Result In</p>
<p>This command searches through the strings within a table to see if any of them contain the sub-string within (as a portion of) them. The index of the first table entry that contains that substring is placed in the result variable. If there are no matches, a -1 is placed in the result. The search IS case sensitive.</p>

**Strip Leading Alpha Characters:**

Advanced Toolkit  
String Group

<p>Arguments: String</p>
<p>This command is used to remove non-numeric characters from the beginning of a string to allow the string to be converted to its numeric equivalent value. The OptoControl commands that do a numeric conversion require that the first character in the string be numeric, or else the result will be zero.</p>

**Strip Leading Characters:**

Standard Toolkit  
String Group

<p>Arguments: String Char to Strip</p>
<p>This command is used to remove any quantity of a specific character from the beginning of a string.</p>

**Summation**

Advanced Toolkit  
Mathematical Group

<p>Arguments: Plus</p>
<p>This command works within a "Summation Begin..." and a "Summation End" pair of commands. Each time this command is executed, the assigned value is added to the running summation.</p> <p><b>Note:</b> this command can only be used as a portion of a summation sequence and must only be used within the same operation block as the rest of the corresponding sequence. Failure to follow these rules, or an unfinished Begin/End pair may cause a fault during download or may cause program malfunction.</p>

**Summation Begin...**

Advanced Toolkit  
Mathematical Group

<p>Arguments: &lt;None&gt;</p>
<p>This command indicates the beginning of a summation sequence. Once this command is executed, a zero value is moved to the running summation. The "Summation" command is used to add values to the running total.</p> <p><b>Note:</b> this command can only be used as a portion of a summation sequence and must only be used within the same operation block as the rest of the corresponding sequence. Failure to follow these rules, or an unfinished Begin/End pair may cause a fault during download or may cause program malfunction.</p>

**Summation End**

Advanced Toolkit  
Mathematical Group

Arguments: Put Result In

This command is used to indicate the end of a summation sequence. Once this command is issued the running summation total is moved to the specified destination variable and the sequence is complete.

**Note:** this command can only be used as a portion of a summation sequence and must only be used within the same operation block as the rest of the corresponding sequence. Failure to follow these rules, or an unfinished Begin/End pair may cause a fault during download or may cause program malfunction.

**Swap Values**

Advanced Toolkit  
Mathematical Group

Arguments: Swap  
With

This command causes two variables of like types (integer-integer or float-float) to exchange values.

**Switch ... EndSwitch**

Advanced Toolkit  
Logical Group

Arguments: Switch Value

This command initiates a Switch Selection portion of code. The argument's value is used in the subsequent Case commands for comparison. Each switch command must have a corresponding "EndSwitch" command.

**Note:** this command can only be used as a portion of a matched pair in a Switch/Endswitch sequence and must only be used within the same operation block as the rest of the corresponding sequence. Failure to follow these rules, or an unfinished pair may cause a fault during download or may cause program malfunction.

**T**

<b>Table Bit On?:</b>	Advanced Toolkit Logical Condition Group
Arguments: Index Table Bit To Test	
This command checks a specific bit in a specific table element to determine if it is set or not.	

<b>Table Bit Semaphore Reset:</b>	Standard Toolkit Logical Group
Arguments: <None>	
This command clears the Table Bit Change lock semaphore and can be placed in the first block in the POWERUP chart and called once. <i>This command is no longer required as the semaphores now automatically unlock after a time-out period. It is included for those who wish to use it for good program initialization.</i>	
The function of this command is identical to the "Bit Semaphore Reset:" operation described on page (except a different semaphore is used). For a greater depth understanding of the purpose of this command, please read that section.	

<b>Table Element Bit Change:</b>	Advanced Toolkit Logical Group
Arguments: Control Bit To Change Index Table	
This command changes a specific bit in a table element based on the logical value of a control variable.	

<b>Table Element Bit OR</b>	Advanced Toolkit Logical Group
Arguments: Mask Index Table Put Result In	
This command is most easily described by the following pseudocode formula: Result = Table[Index] <b>BIT-OR</b> Mask_Variable	

<b>Table Element False?</b>	Standard Toolkit Logical Condition Group
Arguments: Index Table	
This command is used in exactly the same manner as the equivalent variable commands, except it acts upon a table element, instead of a variable.	

<b>Table Element True?</b>		Standard Toolkit
Arguments: Index Table		Logical Condition Group
This command is used in exactly the same manner as the equivalent variable commands, except it acts upon a table element, instead of a variable.		

<b>Table Elements Equal (Float)?</b>		Standard Toolkit
Arguments: Index Table Index Table		Logical Condition Group
This condition compares individual values of two float table elements.		

<b>Table Elements Equal (Integer)?</b>		Standard Toolkit
Arguments: Index Table Index Table		Logical Condition Group
This condition compares individual values of two integer table elements.		

<b>Test Timer Expired</b>		Advanced Toolkit
Arguments: Timer Put Result In		Time/Date Group
This operation checks to see if a timer has expired, if so it returns a true value.		

<b>Transmit Character Via Serial (Port)</b>		Standard Toolkit
Arguments: From		Communication - Serial Group
This command sends one character to the currently locked serial port. It works the same as the equivalent OptoControl command with the addition of the locked port characteristics.		

<b>Transmit Date (Port)</b>		Advanced Toolkit
Arguments: <None>		Communication - Serial Group
This command sends the current (formatted) date to the currently locked serial port. It works the same as the equivalent OptoControl command with the addition of the locked port characteristics.		

<b>Transmit Formatted Number (Port)</b>		Advanced Toolkit
Arguments: From Length Decimals		Communication - Serial Group
This command sends a number formatted as floating point to the currently locked serial port. It works the same as the equivalent OptoControl command with the addition of the locked port characteristics.		

**Transmit Long to (Port) 1<sup>st</sup>=MSB**

Advanced Toolkit  
Communication - Serial Group

Arguments: Value to Send

This command will send four bytes to the currently locked communications port from a numeric variable, converting them to individual byte values. It will transmit them with the most significant first and the least significant last.

**Transmit Long to (Port) 4<sup>th</sup>=MSB**

Advanced Toolkit  
Communication - Serial Group

Arguments: Value to Send

This command will send four bytes to the currently locked communications port from a numeric variable, converting them to individual byte values. It will transmit them with the least significant first and the most significant last.

**Transmit Long to Port 1<sup>st</sup>=MSB**

Advanced Toolkit  
Communication - Serial Group

Arguments: Port# (0-3)  
Value to Send

This command will send four bytes to a communications port from a numeric variable, converting them to individual byte values. It will transmit them with the most significant first and the least significant last.

**Transmit Long to Port 4<sup>th</sup>=MSB**

Advanced Toolkit  
Communication - Serial Group

Arguments: Port# (0-3)  
Value to Send

This command will send four bytes to a communications port from a numeric variable, converting them to individual byte values. It will transmit them with the least significant first and the most significant last.

**Transmit NewLine Via Serial (Port)**

Standard Toolkit  
Communication - Serial Group

Arguments: <None>

This command sends a Carriage Return and Line Feed character to the currently locked serial port. It works the same as the equivalent OptoControl command with the addition of the locked port characteristics. This command will not return until the command has completed. For use on serial ports (0-3, and 5-6).

**Transmit NewLine Via Serial (Port) W/Timeout**

Standard Toolkit  
Communication - Serial Group

Arguments: Put Status In

This command sends the transmit buffer to the currently locked port. It is used primarily with the network ports to indicate to the controller that it should send the contents of the transmit buffer to the desired destination. It works the same as the equivalent OptoControl command with the addition of the locked port characteristics. A status return value of zero indicates success. For use on a network port only (4, 7-10)

<b>Transmit Number (Port)</b>	Advanced Toolkit Communication - Serial Group
Arguments: From	
This command sends a number in the default format to the currently locked serial port. It works the same as the equivalent OptoControl command with the addition of the locked port characteristics.	

<b>Transmit Number as Field (Port)</b>	Advanced Toolkit Communication - Serial Group
Arguments: From Length	
This command sends a number in the default format (taking up a certain length) to the currently locked serial port. It works the same as the equivalent OptoControl command with the addition of the locked port characteristics.	

<b>Transmit OPTOMUX Ack:</b>	Advanced Toolkit Communication – I/O Group
Arguments: Port Message Put Result In	
This command sends an acknowledgment/response message in the format expected by an OptoMux master. When used with the "Receive OptoMux Msg" commands, this allows the controller to act as an OptoMux slave device.	

<b>Transmit OPTOMUX Nak:</b>	Advanced Toolkit Communication – I/O Group
Arguments: Port Error Number Put Result In	
This command sends an acknowledgment message in the format expected by an OptoMux master. When used with the "Receive OptoMux Msg" commands, this allows the controller to act as an OptoMux slave device.	

<b>Transmit String Via Serial (Port)</b>	Standard Toolkit Communication - Serial Group
Arguments: From	
This command sends a string to the currently locked serial port. It works the same as the equivalent OptoControl command with the addition of the locked port characteristics.	

<b>Transmit String XON/XOFF (Port):</b>	Advanced Toolkit Communication - Serial Group
Arguments: String Timer Timer Preset Put Status In	
This command transmits a string to the currently locked serial port using the industry standard XON/XOFF protocol for throughput handshaking. The Timer and Timer preset are used to handle port lockups during an "XOFF" period.	

**Transmit String XON/XOFF:**

Advanced Toolkit  
Communication - Serial Group

Arguments: String  
Timer  
Use Port#  
Put Status In

This command transmits a string to a serial port using the industry standard XON/XOFF protocol for throughput handshaking. The Timer is used to handle port lockups during an "XOFF" period.

**Transmit Table Via Serial (Port)**

Standard Toolkit  
Communication - Serial Group

Arguments: Index  
Table

This command sends 32 elements of a numeric table to the currently locked serial port. It works the same as the equivalent OptoControl command with the addition of the locked port characteristics. See page **Error! Bookmark not defined.** for a brief explanation of "**Error! Reference source not found.**".

**Transmit Time (Port)**

Advanced Toolkit  
Communication - Serial Group

Arguments: <None>

This command sends the current (formatted) time to the currently locked serial port. It works the same as the equivalent OptoControl command with the addition of the locked port characteristics.

**Turn Sub Stepping Off**

Advanced Toolkit  
Chart Group

Arguments: Chart

This operation is used to assist in debugging subroutines. If the debugger is set-up to step through sub-routines (Auto Step or Single Step), this gives the program the ability to determine which routines are stepped into.

**Turn Sub Stepping On**

Advanced Toolkit  
Chart Group

Arguments: Chart

This operation is used to assist in debugging subroutines. If the debugger is set-up to step through sub-routines (Auto Step or Single Step), this gives the program the ability to determine which routines are stepped into. This command only works following a "Turn Sub-Stepping Off" command, while the debug mode is in single or auto-step.

**U**

<b>Until &lt;</b>		Standard Toolkit Logical Group
	Arguments: Until Is < Than	
	This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If the value of the first argument is less than the value of the second argument, the loop will end at this point and execution will continue, if not, the loop will be executed again.  Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.	
<b>Until &lt;=</b>		Standard Toolkit Logical Group
	Arguments: Until Is <= To	
	This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If the value of the first argument is less or equal to the value of the second argument, the loop will end at this point and execution will continue, if not, the loop will be executed again.  Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.	
<b>Until &lt;&gt;</b>		Standard Toolkit Logical Group
	Arguments: Until Is <> To	
	This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If the value of the first argument does not equal the value of the second argument, the loop will end at this point and execution will continue, if not, the loop will be executed again.  Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.	
<b>Until =</b>		Standard Toolkit Logical Group
	Arguments: Until Is = To	
	This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If the value of the first argument equals the value of the second argument, the loop will end at this point and execution will continue, if not, the loop will be executed again.  Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.	

**Until >**

Standard Toolkit  
Logical Group

Arguments: Until  
Is > Than

This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If the value of the first argument is greater than the value of the second argument, the loop will end at this point and execution will continue, if not, the loop will be executed again.

Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**Until >=**

Standard Toolkit  
Logical Group

Arguments: Until  
Is >= To

This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If the value of the first argument is greater or equal to the value of the second argument, the loop will end at this point and execution will continue, if not, the loop will be executed again.

Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**Until AND**

Advanced Toolkit  
Logical Group

Arguments: Until  
And

This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If the value of the first argument and the value of the second argument are both true (non-zero), the loop will end at this point and execution will continue, if not, the loop will be executed again.

Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**Until Chart Running**

Standard Toolkit  
Chart Group

Arguments: Chart

This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If the specific chart is running, the loop will end at this point and execution will continue, if not, the loop will be executed again.

Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**Until Chart Stopped**

Standard Toolkit  
Chart Group

Arguments: Chart

This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If the specific chart is stopped, the loop will end at this point and execution will continue, if not, the loop will be executed again.

Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**Until Chart Suspended**

Standard Toolkit  
Chart Group

Arguments: Chart

This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If the specific chart is suspended, the loop will end at this point and execution will continue, if not, the loop will be executed again.

Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**Until False**

Standard Toolkit  
Logical Group

Arguments: \_\_\_\_\_

This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If the value of the object is zero (false), the loop will end at this point and execution will continue, if not, the loop will be executed again.

Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**Until Off Latch**

Advanced Toolkit  
Logical Group

Arguments: Point

This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If the digital input off latch is set, the loop will end at this point and execution will continue, if not, the loop will be executed again.

Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**Until On Latch**

Advanced Toolkit  
String Group

Arguments: From  
To

This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If the digital input on latch is set, the loop will end at this point and execution will continue, if not, the loop will be executed again.

Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**Until OR**

Advanced Toolkit  
Logical Group

Arguments: Until  
OR

This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If either the value of the first argument or the value of the second argument is true (non-zero), the loop will end at this point and execution will continue, if not, the loop will be executed again.  
Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**UNTIL Timer Expired**

Standard Toolkit  
Logical Group

Arguments: Timer

This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If the timer has expired, the loop will end at this point and execution will continue, if not, the loop will be executed again.  
Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**UNTIL Timer Expired OR**

Advanced Toolkit  
Logical Group

Arguments: Timer  
OR

This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If either the timer has expired or the value of the second argument is true (non-zero), the loop will end at this point and execution will continue, if not, the loop will be executed again.  
Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**Until True**

Standard Toolkit  
Logical Group

Arguments: \_\_\_\_\_

This command is used to indicate the end of an in-block looping sequence initiated by the corresponding "Loop...Until" statement. If the value of the argument is true (non-zero), the loop will end at this point and execution will continue, if not, the loop will be executed again.  
Note: If this command is not matched with the required offsetting command(s) a failure will occur during download.

**W**

<b><i>Write Byte to Controller Memory</i></b>		Advanced Toolkit Controller Group
Arguments:	Value Address	
This command writes one byte (8 bits) to a location in the controllers memory.		
<b>Warning:</b> Direct memory manipulation commands, such as this command, should be used with caution. Monitoring or modifying certain memory locations can cause unexpected operation and program failure.		
<b><i>Write Long to Controller Memory</i></b>		Advanced Toolkit Controller Group
Arguments:	Value Address	
This command writes four bytes (32 bits) to a location in the controllers memory.		
<b>Warning:</b> Direct memory manipulation commands, such as this command, should be used with caution. Monitoring or modifying certain memory locations can cause unexpected operation and program failure.		
<b><i>Write String to PC Serial Port (ISA Only):</i></b>		Advanced Toolkit Communication - Serial Group
Arguments:	String Port Address	
This command causes a string to be transmitted out a serial port of the controlling PC. This command works only with the ISA controllers.		
<b><i>Write Word to Controller Memory</i></b>		Advanced Toolkit Controller Group
Arguments:	Value Address	
This command writes two bytes (16 bits) to a location in the controllers memory.		
<b>Warning:</b> Direct memory manipulation commands, such as this command, should be used with caution. Monitoring or modifying certain memory locations can cause unexpected operation and program failure.		