

# Event Driven Control Operator (EDCO) User Manual

Version 1.0



## **Dynatrol Systems Inc.**

#601, 4656 Westwinds Drive N.E.

Calgary, Alberta

Canada, T3J 3Z5

Phone: (403) 235-5611

Fax: (403) 235-5610

## **Copyright Notice**

Copyright © 2002 Dynatrol Systems Inc.  
All Rights Reserved.

This technical document is the copyrighted work of Dynatrol Systems Inc. and the property of Dynatrol Systems Inc. No part of this work may be copied or reproduced without the express written permission of Dynatrol Systems Inc.

Dynatrol Systems Inc. makes no warranty as to the accuracy or use of this document. Documentation may include technical or other inaccuracies or typographical errors. Any use of the technical documentation or the information contained therein is at the risk of the user. Dynatrol Systems Inc. reserves the right to make changes without prior notice.

## **Trademarks**

DS-32 is a trademark of Dynatrol Systems Inc.

Other product names mentioned in this document may be trademarks or registered trademarks of their respective companies.

<b>1</b>	<b>OVERVIEW .....</b>	<b>1-1</b>
1.1	Graphical User Friendly Interface.....	1-1
1.2	Efficient Execution for Time Critical Process Control Applications.....	1-1
1.3	Intelligent Feedback Oscillation Detection.....	1-2
<b>2</b>	<b>FUNCTIONAL SPECIFICATION .....</b>	<b>2-2</b>
2.1	EDCO Processor Interface General View.....	2-3
2.2	Control Primitives.....	2-3
2.3	Linking Operations .....	2-5
2.4	Glossary .....	2-6
<b>3</b>	<b>EDCO GUI.....</b>	<b>3-7</b>
3.1	Types of EDCO Control Icons.....	3-7
3.2	EDCO GUI Compiler .....	3-14
<b>4</b>	<b>EDCO CONFIGURATION .....</b>	<b>4-15</b>
4.1	Task Table.....	4-15
4.2	RIA Table .....	4-16
4.3	VRU Table .....	4-17
4.4	Digital Input Mapping Table .....	4-17
4.5	Accumulator Mapping Table .....	4-18
4.6	BCD Mapping Table.....	4-18
4.7	Latch/Pulse Mapping Table .....	4-19
4.8	Raise/Lower Mapping Table.....	4-19
4.9	Trip/Close Mapping Table.....	4-19
4.10	Pattern Mapping Table.....	4-20
4.11	Double Point Select before Operate Mapping Table .....	4-20
4.12	Analog Input Mapping Table.....	4-21
4.13	Analog Output Mapping Table .....	4-21
4.14	Internal Register Mapping Table .....	4-21
4.15	Control Primitive Table .....	4-22
4.16	Configuration Table .....	4-23
<b>5</b>	<b>EDCO PROCESSOR.....</b>	<b>5-23</b>



# **1 Overview**

EDCO is a group of software components that reside in the PC and Dynatrol Spectrum 32 RTU. EDCO is specially designed for time critical process control applications. It facilitates logic interlock, arithmetic calculations and sequential controls.

The following are the main features of EDCO:

## **1.1 Graphical User Friendly Interface**

The EDCO Graphical Interface will be implemented in the future.

EDCO Graphical Interface provides pictorial view of the control algorithm. With drag and drop control icons, complex control algorithm can be easily implemented and modified. Signals and Controls relationships can be easily visualized. With real time display, states and values of signals and controls are displayed in real time. This feature enables the user to easily trouble shoot the control algorithm.

Unlike ladder diagrams which signals and controls cannot be readily correlated. If signals are out of page bound, it is difficult to visualize the relationship of the signals involved. EDCO Graphical Interface presents the control icons with signal paths and control paths connected between control icons, therefore user is able to visually correlated the signals and controls involved.

## **1.2 Efficient Execution for Time Critical Process Control Applications**

Most of the control software, such as PLC ladder diagram style of software, the CPU processes all the rungs of the ladder diagram at every loop time. Even the inputs have not been changed, the CPU processes all the rungs regardless. If the rungs are not arranged sequentially according to the logic, that is the inputs of the rung come from the outputs of the rungs following the present rung, the control outputs may take many loop times to be affected. For feedback controls, the arrangement of rungs sequentially according to the logic is not possible. For this type of controls, the control outputs will take many loop times to be affected.

EDCO implements the processing very differently. When there is a change of input, EDCO will process all the operators that are affected by this change of input. That is EDCO only wakes up when there is a change to one of the inputs and only the operators that are affected by this change of input will be processed. With this feature, EDCO is very efficient and able to perform time critical process controls.

## **1.3 Intelligent Feedback Oscillation Detection**

In process control applications, feedback control is very common. That is the signals from the results of some operators are fed back to the inputs of the operators. Feedback can cause digital logic to be flip-flopping; that is oscillating between ON/OFF states. For analog signals, feedback can cause output oscillation or saturation to certain values. Using electronic components to implement the controls such as AND, OR gates and Amplifiers as analog adder or multiplier, the undesirable results (oscillation/saturation) usually localized to the affected portion of the control. However, in microprocessor based controls, the undesirable results could render the control inoperable because excessive use of CPU time. Undesirable oscillation or saturation is of little use in practical control applications. EDCO has built in algorithm to detect oscillation or saturation. When oscillation or saturation is detected, EDCO logs the warning messages indicating which operators have been processed many times in the single change of input. With the warning messages, undesirable oscillation or saturation can be easily eliminated.

The warning message is in the form of the following:

WARNING: Oscillation detected in Primitive # for more than N times.

# is the primitive index. N is the maximum allowable execution of the same primitive on a single value change event. This warning simply informs the user that the indicated primitive has been executed that many times in a single value change event. Since oscillation is of little practical application in real situation, the user should check the logic involved. Some control algorithms require many executions on the same primitive before it stabilized. For these cases, the user should increase the maximum allowable execution counts.

## **2 Functional Specification**

EDCO consists of 3 main components. They are as follows:

- EDCO Graphical User Interface (GUI): This is a piece of software resides in the PC. With this software, user is able to implement, modify, monitor and trouble shoot the control algorithm. EDCO GUI also translates the control diagram to downloadable file for the RTU.
- EDCO Configuration: This is part of DynaConfig configuration tool, which is also resided in the PC. With this software tool, user is able to configure the parameters that control the operation characteristics of EDCO. Also point mapping and allocation can be assigned and reserved. EDCO Configuration also translates the configuration into downloadable file for the RTU.
- EDCO Processor: This is a piece of software resides in the Dynatrol Spectrum 32

RTU. This software performs the logic interlock, arithmetic calculations and sequential controls according to the EDCO control operator configuration.

EDCO GUI will be implemented in the future. At the mean time, user can use other commercial CAD package to draw up the control diagram and use DynaConfig to input the control algorithm.

## **2.1 EDCO Processor Interface General View**

EDCO Processor interfaces with other applications via DBM. Internally, EDCO maintains a table of registers, which hold the intermediate data values. Before EDCO can process the control operators, signals from the external have to be brought into the EDCO internal registers. This is accomplished by linking the external system points to the internal registers. Conversely, when EDCO needs to output to the external points, internal registers are linked to external system control points. EDCO can also be controlled by external applications. This is accomplished by EDCO owned control points. Conversely, EDCO can also provide indications to external applications. This is accomplished by EDCO owned indication points.

To summarize, there are two ways the external application can influence EDCO. One is via system indication points such as DI, ACC, BCD, and AI. Another way is via EDCO owned control points such as L/P, R/L, T/C, DSP, PAT and AO. There are also two ways EDCO can control or inform external applications. One is via system control points such as L/P, R/L, T/C, DSP, PAT and AO. Another way is via EDCO owned indication points such as DI, ACC, BCD, and AI. For all cases, EDCO requires linking information such that EDCO is able to correlate system points and EDCO internal registers. This is accomplished via mapping tables in the EDCO configuration.

## **2.2 Control Primitives**

Control Primitives are control functions supported by EDCO. The basic structure of the Control Primitive consists of 2 inputs (IN1, IN2), an operator function code and 1 output (OUT). All the inputs and output are pointers to the EDCO internal registers.

The following describes the Control Primitives:

Note: Computer high level language “C” syntax is used for the definition.

- **NOP:** No operation
- **AND:** AND Gate:  $OUT = IN1 \& IN2$  (Arithmetic AND)
- **OR:** OR Gate:  $OUT = IN1 | IN2$  (Arithmetic OR)
- **XOR:** Exclusive OR Gate:  $OUT = IN1 \wedge IN2$  (Arithmetic Exclusive OR)
- **NOT:** Inverter (One’s Compliment):  $OUT = \sim IN1$ , IN2 is not used
- **S/R:** Set/Reset:  $OUT = 1$  if IN1 Bit 1: = 0 to 1 transition,  $OUT = 0$  if IN2 Bit 1: = 0 to 1 transition

- **PLS:** Pulse:  $OUT = A$  pulse ( Bit 1: 0 to 1, 1 to 0 ) with IN2 equal ms in duration if IN1 Bit 1: = 0 to 1 transition, note the output pulse is retriggerable. If IN2 is zero, OUT is always zero.
- **TDE:** Time Delay Energizer:  $OUT = 1$  if IN1 Bit 1: = 0 to 1 transition and stay = 1 for longer than IN2 ms.  $OUT = 0$  if IN1 Bit 1: = 0. If IN2 is zero, OUT follows the IN1.
- **TDD:** Time Delay De-energizer:  $OUT = 0$  if IN1 Bit 1: = 1 to 0 transition and stay = 0 for longer than IN2 ms.  $OUT = 1$  if IN1 Bit 1 = 1. If IN2 is zero, OUT follows the IN1.
- **ADD:** Addition:  $OUT = IN1 + IN2$
- **SUB:** Subtraction:  $OUT = IN1 - IN2$
- **MUL:** Multiplication:  $OUT = IN1 * IN2$
- **DIV:** Division:  $OUT = IN1 / IN2$
- **MOD:** Modulus:  $OUT = IN1 \% IN2$
- **GRT:** Greatest:  $OUT = IN1$  if  $IN1 > IN2$  else  $OUT = IN2$
- **SML:** Smallest:  $OUT = IN1$  if  $IN1 < IN2$  else  $OUT = IN2$
- **AVG:** Average:  $OUT = (IN1 + IN2) / 2$
- **DIF:** Absolute difference:  $OUT = IN1 - IN2$  if  $IN1 > IN2$  else  $OUT = IN2 - IN1$
- **SQR:** Square Root:  $OUT =$  Square root of IN1, IN1 is treated as unsigned 32 bit, IN2 is not used
- **TAB:** Tap Analog position to Binary conversion: Valid range of Tap Analog positions are 1 to 16,  $OUT = 2^{**} ( IN1 - 1 )$ ,  $OUT = 0$  if IN1 is out of range. IN2 is not used.
- **TBA:** Tap Binary position to Analog conversion: Valid values of Tap Binary positions are 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384. 32768,  $OUT = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16$  respectively.  $OUT = 0$  if IN1 is out of range. IN2 is not used.
- **RMP:** Ramp:  $OUT =$  Ramp to IN1 in IN2 ms on every change of IN1. Frequency to update the OUT is determined by the configuration. If IN2 is zero, OUT follows the IN1 with no delay.
- **CNT:** Counter: No action if  $IN2 = 0$ ,  $OUT = OUT + 1$  for every transition of IN1 if IN2 Bit 1 = 1,  $OUT = 0$  if IN2 Bit 1 = 0 to 1 transition.
- **SET:** Set Output:  $OUT = IN1$  if IN2 Bit 1 = 0 to 1 transition.
- **SEL:** Selector:  $OUT = IN1$  if IN2 Bit 1 = 1. 2 SELs form a selector.
- **CMP:** Comparator:  $OUT = 1$  if  $IN1 > IN2$ . Note: the Deadband is 1 in this comparator.
- **EQU:** Equal:  $OUT = 1$  if  $IN1 = IN2$ .
- **ATB:** Analog To BCD conversion: Valid range of Analog is 0 to 9999.  $OUT =$  BCD.  $OUT = 9999$  in BCD if IN1 is over range,  $OUT = 0$  if IN1 is under range. IN2 is not used.
- **BTA:** BCD To Analog conversion: Valid range of BCD is 0 to 9999 in BCD. OUT

= Analog value of IN1. OUT = -1 in IN1 is invalid. OUT = 9999 if IN1 is over range, OUT = 0 if IN1 is under range. IN2 is not used.

- **BSR:** Binary Shift Right:  $OUT = IN1 \gg IN2$ . Maximum IN2 is 32.
- **BSL:** Binary Shift Left:  $OUT = IN1 \ll IN2$ . Maximum IN2 is 32.
- **OT1:** Zero To One: At initialization, or any change on either inputs IN1 or IN2, regardless to the state of Output OUT, OUT = 0 to 1 transition. This is used to generate timing at initialization or on command.
- **WOF:** Weekly On Off: IN1 and IN2 are weekly milliseconds (milliseconds after Monday Midnight). IN1 is the On time. When the IN1 weekly millisecond is reached, OUT = 1. IN2 is the Off time. When the IN2 weekly millisecond is reached, OUT = 0. If the value of the input is greater than or equal to 604,800,000 (maximum weekly milliseconds =  $1000*60*60*24*7$ ), the On or Off action is disable accordingly. At startup, OUT = 1 if current weekly millisecond is greater than or equal IN1 and less than IN2, else OUT = 0. If any one of the inputs is greater than or equal to 604,800,000, no action is performed for OUT. In the EDCO GUI, user is prompted with weekday, hour, minute, seconds and millisecond fields for the input values.

It should be noted that all Boolean operations are on all 32 bits. It is up to the user to ensure that the results are as desired. Care should be taken according to the Input and Output Linking operation definitions as described in the next section.

## **2.3 Linking Operations**

There are two types of linking, namely Input linking and Output linking. All linking information is contained in the mapping tables of the configuration. The data type and ownership of the points determine the Input or Output linking. The following summarize the linking types and operation performed on each data type.

### **Input linking (System Input Points: Owned by other applications other than EDCO):**

- DI: If DI = 0, Internal Register = 0, if DI = 1, Internal Register = 1
- ACC: Internal Register = ACC Value (Support Delta ACC)
- BCD: Internal Register = BCD Value
- AI: Internal Register = AI Value

### **Input linking (EDCO Owned Output Points):**

- L/P: If L/P = latch OFF, Internal Register = 0, if L/P = latch ON, Internal Register = 1, if L/P = Pulse, Internal Register Pulses according to the L/P parameters.
- R/L: If R/L = Raise, Internal Register = 0 to 1 transition, if R/L = Lower, Internal Register = 1 to 0 transition,
- T/C: If T/C = Trip, Internal Register = 1 to 0 transition, if T/C = Close, Internal Register = 0 to 1 transition,

- DSP: Internal Register Pulses according to the DSP parameters.
- PAT: Internal Register = PAT value
- AO: Internal Register = AO value

**Output linking (System Output Points: Owned by other applications other than EDCO):**

- L/P: If Internal Register = 0, L/P = latch OFF, if Internal Register != 0, L/P = ON or pulses according to the configuration of the L/P mapping table, duration is from the configuration, and number pulses = Internal Register value.
- R/L: If Internal Register = 0, no action, if Internal Register = -ve value, R/L = Lower with duration equal to the value of the internal register, if Internal Register = +ve value, R/L = Raise with duration equal to the value of the internal register.
- T/C: If Internal Register Bit 1 = 1 to 0 transition, T/C = Trip, if Internal Register Bit 1 = 0 to 1 transition, T/C = Close, Duration is according to the configuration in the T/C mapping table.
- DSP: If Internal Register != 0, DSP = Pulse with duration equal to Internal Register value, , only the lower 16 bit of the Internal Register is used as the duration timer.
- PAT: PAT value = Internal Register Value
- AO: AO value = Internal Register Value

Note: For L/P configured as Pulse duration or Pulse train, R/L, T/C and DPS, the controls only execute if there are changes to the corresponding linked internal register. This is consistent with the event driven concept. If there is no change, output does not need be performed.

**Output linking (EDCO Owned Indication Points):**

- DI: If Internal Register = 0, DI = 0, if Internal Register Bit 1 = 1, DI = 1
- ACC: ACC value = Internal Register Value (Send Absolute Accumulator to DBM)
- BCD: BCD value = Internal Register Value
- AI: AI value = Internal Register Value

It should be noted: The internal register is a 32 bit integer. When it is linked to a 16 bit value, the lower 16 bit is used. When it is linked to a 1 bit value, only the first bit is used.

For ACC, input to EDCO is a delta value, output to DBM is an absolute value.

## **2.4 Glossary**

- **EDCO Internal Registers:** EDCO Internal Registers are 32 bit signed integers. They are used to store intermediate values for bridging the control primitives and system points. When it is used for SQR function, it is treated as a 32 bit unsigned

integer.

- **Control Icons:** Control Icons are control symbols found in the EDCO GUI. These Icons can be built from Control Primitives. Some Control Icons have the same equivalent of Control Primitives.
- **Control Operator:** General terms for Control Icons and Control Primitives.
- **Control Primitives:** These are the control functions supported by EDCO Processor.
- **Mapped Points:** System points that are used by the application. These points are mapped in the mapping tables of the EDCO configuration.
- **Operator Icons:** See Control Icons.
- **Operator Primitives:** See Control Primitives.
- **Owned Points:** Points that belong to the EDCO processor task.
- **System Points:** Points that belong to all the RIAs of the RTU. Depending on the context, owned points are also system points. For example, in the configuration mapping tables, it is legal to map the owned points in these tables. Under this context, the owned points are also system points.

### **3 EDCO GUI**

EDCO GUI resides in the PC. It provides the tools for user to implement, edit and trouble shoot the control algorithm. It presents the algorithm in a graphical diagram with control icons.

With drag and drop control icons, complex control algorithm can be easily implemented and modified. EDCO control diagram consists of control icons with signal paths and control paths connected between control icons. Signals and Controls relationships can be easily visualized. With real time display, states and values of signals and controls are displayed in real time. This feature enables the user to easily trouble shoot the control algorithm.

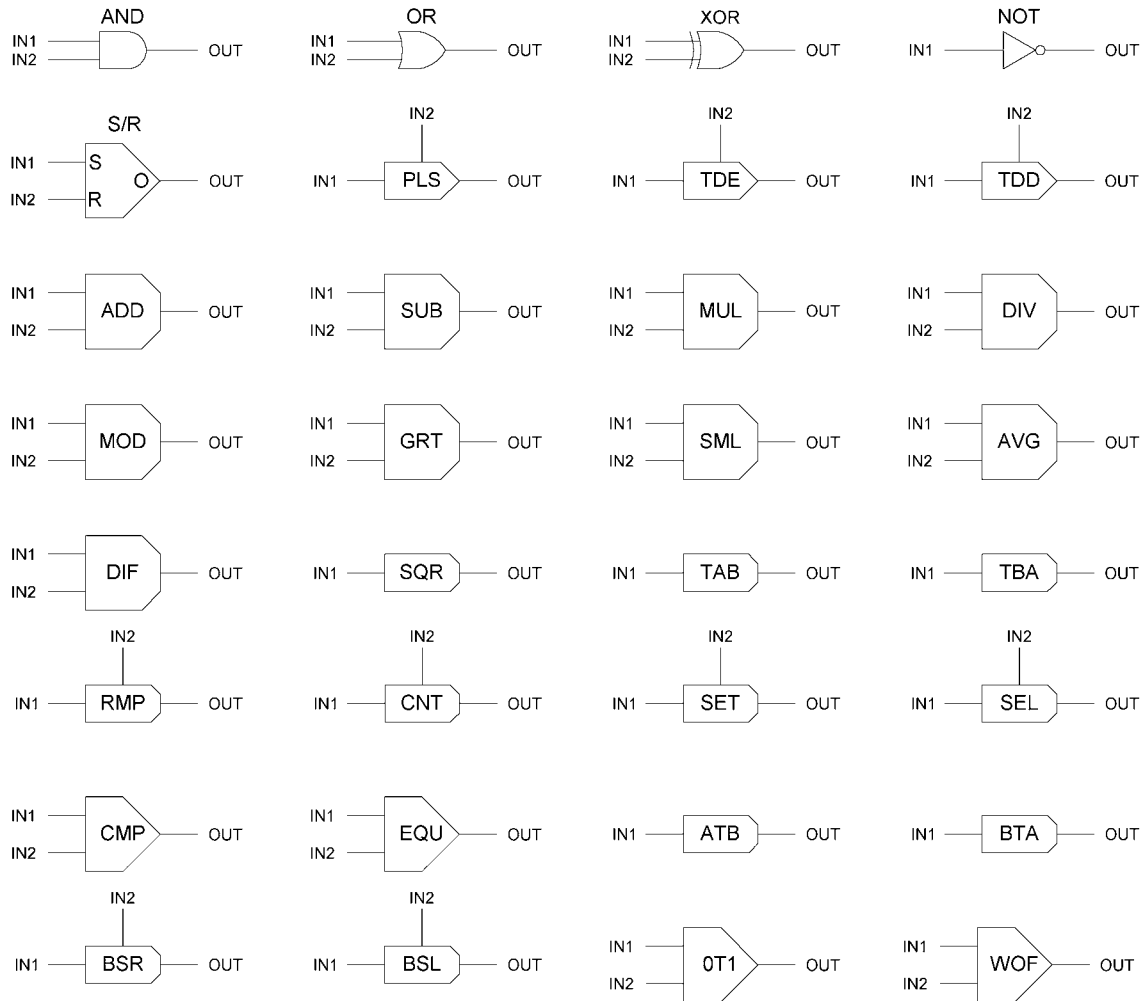
EDCO GUI provides tools for user to define Control Icons that consist of Control Primitives. That is to say the EDCO control primitives are subset of EDCO control icons. EDCO GUI also extends the number of inputs for the AND gate and OR gate to more than 2 inputs.

EDCO GUI also allows the user to use mnemonic assignment for EDCO internal registers.

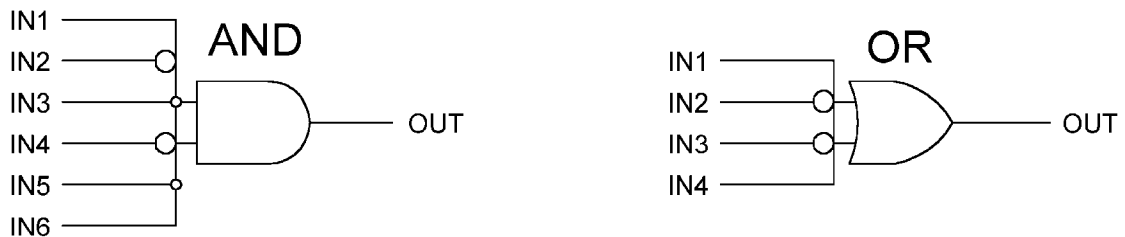
#### **3.1 Types of EDCO Control Icons**

To summarize, there are 3 types of EDCO control icons:

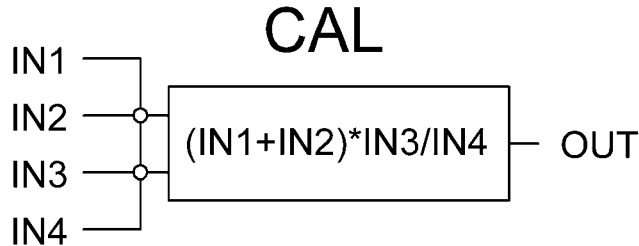
- The first type is control icons representing the EDCO control primitives as presented in the following diagram. The following summarizes the control primitives supported by the EDCO Processor:



- The second type is implied control icons. There are 3 control icons that allow the user to extend the number of inputs up to 8 inputs; namely for the AND gate, OR gate and Calculation Icon. The inputs of AND gate or OR gate can be more than 2 inputs. The inputs of these gates can be inverted as well. The following illustrate the AND gate and OR gate icons:



The Calculation Icon provides the user to input the calculation by simply type in the formula, such as follows:



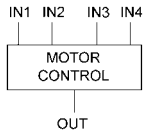
GUI compiler translates the above control icons into EDCO control primitives that represent the corresponding logics and calculations.

- The third type of control icons are designed by user that consist of EDCO control primitives. The following are 3 examples of EDCO control icons that made up of EDCO control primitives:

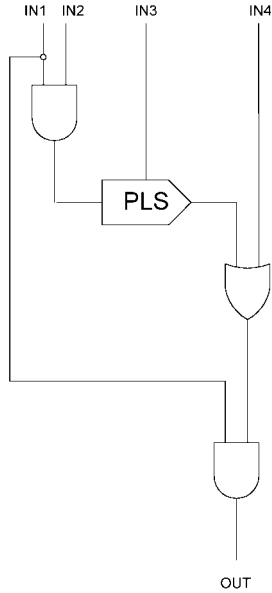
The first example is Motor Control. Input 1 is the MOTOR RUN Command. Input 2 is the MOTOR SAFTY SWITCH; it has to be ON before motor control can be issued. Input 3 is the Timer for the Motor Control to stay ON. If the Motor Status does not come ON within this time, the Motor Control drops OFF. Input 4 is the MOTOR STATUS. OUT is the MOTOR CONTORL.

It is important to note that all digital controls are OFF if the remote/local switch is in the local mode. Therefore depending on the nature of the control, user should design the controls to account for the control drops OFF when remote/local switch is in local mode. In the above example, the MOTOR CONTROL drops OFF if the MOTOR STATUS is OFF.

**MOTOR CONTROL ICON**

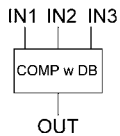


**MOTOR CONTROL PRIMITIVES**

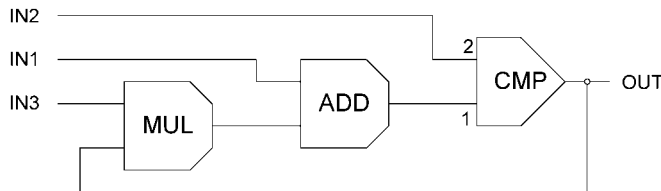


The second example is to provide deadband for the Comparator function. Inputs 1 and 2 are two variables for comparison. Input 3 is the deadband.

**COMPARATOR WITH DEADBAND**

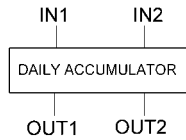
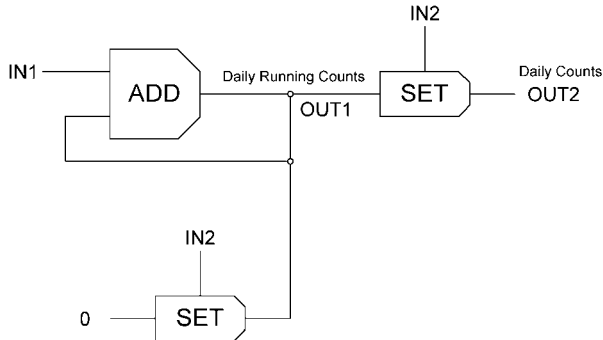


**COMPARATOR WITH DEADBAND PRIMITIVES**



GUI compiler translates the above control icons into EDCO control primitives that represent the corresponding logics and calculations.

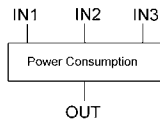
The third example is daily accumulator. Input 1 is delta change of accumulator in interest. Input 2 is the midnight event. OUT 1 is the daily running counts. Out 2 is the last midnight accumulated counts.

**DAILY ACCUMULATOR ICON****DAILY ACCUMULATOR PRIMITIVES**

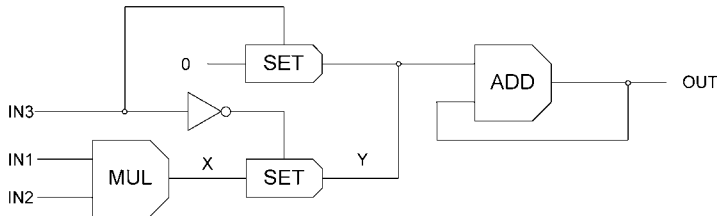
The ADD primitive is a simple integrator. Every time Accumulator IN1 (delta Accumulator) changed, OUT1 accumulates the counts. That is OUT1 is daily running counts. At midnight, IN2 pulses and causes SET copy OUT1 to OUT2. That is OUT2 is the last midnight daily-accumulated counts. At the same time OUT1 is reset to zero count.

It should be noted, the sequence of the SET primitives is important in this case. The second SET primitive has to follow the first SET primitive. If the first SET primitive follows the second primitive, the OUT2 is set to zero all the time. The OUT1 and OUT2 can be linked to Analog Inputs, BCDs or Accumulators. If it is linked to Accumulators, EDCO processor sends absolute accumulator values to DBM. Care should also be taken when feed back is used as in this case. The output of the ADD is feed back to the input. Since the implementation is event driven, if IN1 comes from many arithmetic primitives, every value change of the inputs to the arithmetic primitives will activate the ADD integrator. This may not be desirable. The next example is accumulation of power consumption.

**POWER CONSUMPTION ICON**



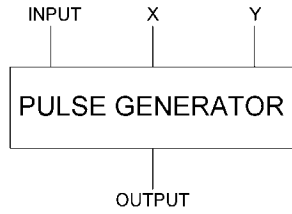
**POWER CONSUMPTION PRIMITIVES**



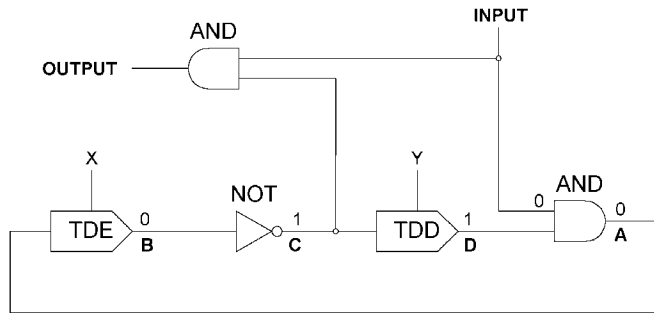
The fourth example is Power Consumption Accumulation. IN1 and IN2 are Voltage and Current respectively. X is the instantaneous power. IN3 is a periodic pulse. The period of this pulse is depending of the frequency of accumulation in interest. The first SET primitive resets Y to zero. This action triggers the ADD primitive to accumulate. Since Y is zero, the value of OUT is unaffected. The second SET sets Y to the value of the power X. This action also triggers the ADD primitive to accumulate the power to OUT. The first SET is needed in case the value of POWER X did not change. Without the first SET primitive, if there were no value change in Y, ADD would not accumulate. Also noted that the trigger of second SET is inverted. That is the rising edge of IN3 reset Y to zero and falling edge of IN3 sets Y equal to X.

The Fifth example is a pulse generator. When the Input is 0, Output is 0. When the Input is 1, Output pulses continuously with X milliseconds ON time and Y millisecond OFF time.

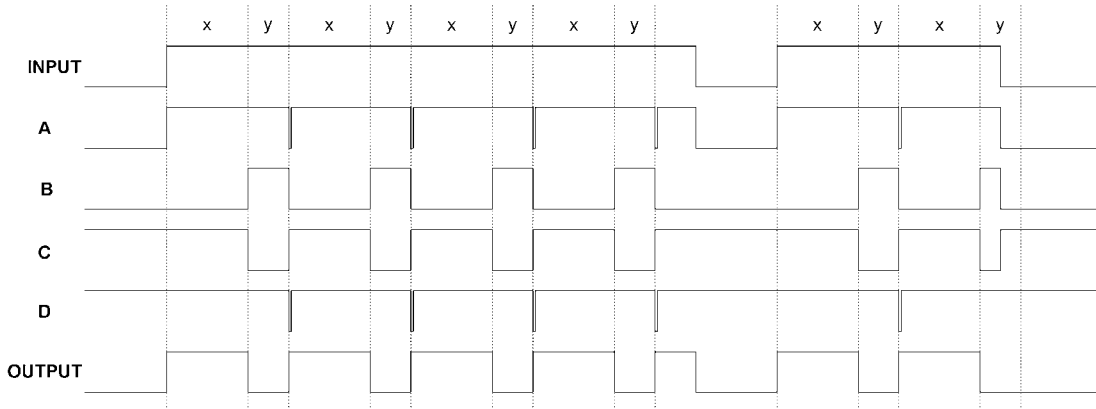
PULSE GENERATOR ICON



PULSE GENERATOR PRIMITIVES

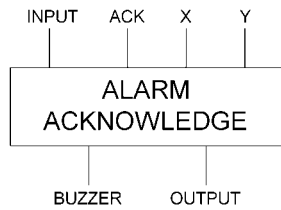


TIMING DIAGRAM

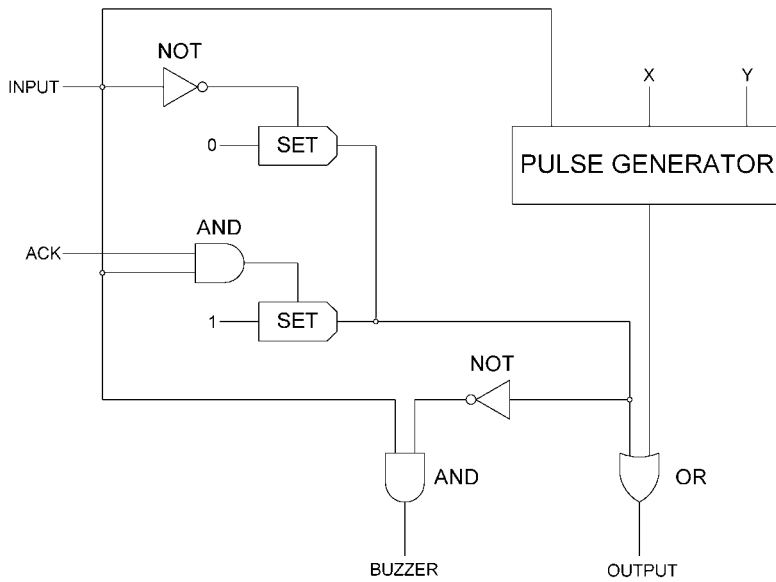


The sixth example is alarm acknowledge. When Input is 1 (alarm condition), Output is pulsing and buzzer is on (indicating unacknowledged alarm condition). Acknowledge is a momentary on button. If acknowledge button is issued when the alarm condition is ON, output latches to ON and buzzer is OFF. If Input is 0 (normal condition), Output and buzzer are OFF.

ALARM ACKNOWLEDGE ICON



ALARM ACKNOWLEDGE PRIMITIVES



EDCO GUI allows nested Control icons as indicated in the above example. That is the Control icon Pulse Generator is used in the Alarm Acknowledge icon.

### **3.2 EDCO GUI Compiler**

The user uses the EDCO GUI to implement the Control Algorithm in the form of EDCO control diagram. The EDCO Control Diagram consists of EDCO control icons. The EDCO GUI compiler takes the information from the control diagram and translates the icons into EDCO control primitives. The file consists of control primitives can be downloaded to the RTU. The EDCO processor resides in the RTU processes the controls and calculations according to the EDCO primitives. EDCO GUI also performs integrity checks during compilation. The control primitive file is same as the control primitive configuration table as described in the next section.

## **4 EDCO Configuration**

EDCO Configuration is part of DynaConfig Configuration. There are 16 tables for the EDCO Configuration.

### **4.1 Task Table**

The Task Table is P02\_TASK. The parameters in this table control the operation characteristics of EDCO. Only personnel with proper password is allowed to edit this table. There is only one record in this table.

The following describes each parameter:

1. TaskPriority: Data Type is a signed integer. Valid range is 10 to 127. This is the running task priority of the EDCO processor task.
2. RamSegment: Data Type is unsigned long. Valid range is 1 to 20. This is the RAM segment that the EDCO processor used during run time.
3. Retrigger: Data Type is unsigned integer. Valid range is 0 to 65535. 0 = retrigger is not allowed. Non zero value = number of retriggers allowed. 65535=always retrigger,
4. DI\_EvtMsk: Data Type is unsigned integer. This mask is used by the DBM to determine whether a DI event should be sent to this application or not. It is set to report Value change only.
5. ACC\_EvtMsk: Data Type is unsigned integer. This mask is used by the DBM to determine whether an Accumulator event should be sent to this application or not. It is set to report Value change only.
6. BCD\_EvtMsk: Data Type is unsigned integer. This mask is used by the DBM to determine whether a BCD event should be sent to this application or not. It is set to report Value change only.
7. LP\_EvtMsk: Data Type is unsigned integer. This mask is used by the DBM to determine whether a LP event should be sent to this application or not. It is set to no report needed.
8. RL\_EvtMsk: Data Type is unsigned integer. This mask is used by the DBM to determine whether a RL event should be sent to this application or not. It is set to no report needed.
9. PAT\_EvtMsk: Data Type is unsigned integer. This mask is used by the DBM to determine whether a PAT event should be sent to this application or not. It is set to no report needed.
10. DPS\_EvtMsk: Data Type is unsigned integer. This mask is used by the DBM to determine whether a DPS event should be sent to this application or not. It is set to no report needed.
11. TC\_EvtMsk: Data Type is unsigned integer. This mask is used by the DBM to determine whether a TC event should be sent to this application or not. It is set to no report needed.

12. AI\_EvtMsk: Data Type is unsigned integer. This mask is used by the DBM to determine whether an AI event should be sent to this application or not. It is set to report Value change only.
13. AO\_EvtMsk: Data Type is unsigned integer. This mask is used by the DBM to determine whether an AO event should be sent to this application or not. It is set to no report needed.
14. DEV\_EvtMsk: Data Type is unsigned integer. This mask is used by the DBM to determine whether a DEV event should be sent to this application or not. It is set to no report needed.
15. SYS\_DigEvtMsk: Data Type is unsigned integer. This mask is used by the DBM to determine whether a System Digital Status event should be sent to this application or not. It is set to no report needed.
16. SYS\_AnaEvtMsk: Data Type is unsigned integer. This mask is used by the DBM to determine whether a System Analog Status event should be sent to this application or not. It is set to no report needed.

## **4.2 RIA Table**

The RIA Table is P02\_RIA. The parameters in this table determine the corresponding number of owned points of the EDCO processor task. It should be noted, all the input points are indication points (DI, ACC, BDC, AI) for external applications. All the control points (LP, RL, TC, DSP, PAT, AO) can be used by external application to influence or control the EDCO processor. There is only one record in this table.

The following describes each parameter:

1. NumDI: Data Type is unsigned integer. This parameter is the number of owned DIs for this application.
2. NumACC: Data Type is unsigned integer. This parameter is the number of owned Accumulators for this application.
3. NumBCD: Data Type is unsigned integer. This parameter is the number of owned BCDs for this application.
4. NumLP: Data Type is unsigned integer. This parameter is the number of owned LPs for this application.
5. NumRL: Data Type is unsigned integer. This parameter is the number of owned RLs for this application.
6. NumPAT: Data Type is unsigned integer. This parameter is the number of owned PATs for this application.
7. NumDPS: Data Type is unsigned integer. This parameter is the number of owned DPSs for this application.
8. NumTC: Data Type is unsigned integer. This parameter is the number of owned TCs for this application.
9. NumAI: Data Type is unsigned integer. This parameter is the number of owned AIs for this application.
10. NumAO: Data Type is unsigned integer. This parameter is the number of owned AOs for this application.

11. NumDEV: Data Type is unsigned integer. This parameter is the number of owned DEVs for this application.

### **4.3 VRU Table**

The VRU Table is P02\_VRU. The parameters in this table determine the VRU ownership of the mapped points. For this application, there is only one VRU. That is there is only one record in this table.

The following describes each parameter:

1. NumDI: Data Type is unsigned integer. This parameter is the number of system DIs used for this application.
2. pDI: Pointer to the DI Mapping table.
3. NumACC: Data Type is unsigned integer. This parameter is the number of system ACCs used for this application.
4. pACC: Pointer to the ACC Mapping table.
5. NumBCD: Data Type is unsigned integer. This parameter is the number of system BCDs used for this application.
6. pBCD: Pointer to the BCD Mapping table.
7. NumLP: Data Type is unsigned integer. This parameter is the number of system LPs used for this application.
8. pLP: Pointer to the LP Mapping table.
9. NumRL: Data Type is unsigned integer. This parameter is the number of system RLs used for this application.
10. pRL: Pointer to the RL Mapping table.
11. NumPAT: Data Type is unsigned integer. This parameter is the number of system PATs used for this application.
12. pPAT: Pointer to the PAT Mapping table.
13. NumDPS: Data Type is unsigned integer. This parameter is the number of system DPSs used for this application.
14. pDPS: Pointer to the DPS Mapping table.
15. NumTC: Data Type is unsigned integer. This parameter is the number of system TCs used for this application.
16. pTC: Pointer to the TC Mapping table.
17. NumAI: Data Type is unsigned integer. This parameter is the number of system AIs used for this application.
18. pAI: Pointer to the AI Mapping table.
19. NumAO: Data Type is unsigned integer. This parameter is the number of system AOs used for this application.
20. pAO: Pointer to the AO Mapping table.

### **4.4 Digital Input Mapping Table**

The Digital Input Mapping Table is P02\_DI. The parameters in this table determine

which DI system points are used in this application. It also establishes the bridging between the DI system points and EDCO internal registers. It should be noted, the points that belong to other RIA applications are input points to the EDCO processor. The points that are owned by EDCO are indication points (EDCO output points) that can be used by other applications. The number of records is the number of system DI points used in this application.

The following describes each parameter:

1. RIAIndex: Data Type is unsigned integer. This is the RIA application Index. This index determines which RIA application the point belongs to.
2. DIPoint: Data Type is unsigned integer. This is the DI point index of the RIA.
3. Linking: Data Type is signed integer. This is the index to the EDCO internal register.

## **4.5 Accumulator Mapping Table**

The Accumulator Mapping Table is P02\_ACC. The parameters in this table determine which ACC system points are used in this application. It also establishes the bridging between the ACC system points and EDCO internal registers. It should be noted, the points that belong to other RIA applications are input points to the EDCO processor. The points that are owned by EDCO are indication points (EDCO output points) that can be used by other applications. The number of records is the number of system ACC points used in this application.

The following describes each parameter:

1. RIAIndex: Data Type is unsigned integer. This is the RIA application Index. This index determines which RIA application the point belongs to.
2. ACCPoint: Data Type is unsigned integer. This is the ACC point index of the RIA.
3. Linking: Data Type is signed integer. This is the index to the EDCO internal register.

## **4.6 BCD Mapping Table**

The BCD Mapping Table is P02\_BCD. The parameters in this table determine which BCD system points are used in this application. It also establishes the bridging between the BCD system points and EDCO internal registers. It should be noted, the points that belong to other RIA applications are input points to the EDCO processor. The points that are owned by EDCO are indication points (EDCO output points) that can be used by other applications. The number of records is the number of system BCD points used in this application.

The following describes each parameter:

1. RIAIndex: Data Type is unsigned integer. This is the RIA application Index. This index determines which RIA application the point belongs to.
2. BCDPoint: Data Type is unsigned integer. This is the BCD point index of the RIA.

3. Linking: Data Type is signed integer. This is the index to the EDCO internal register.

## **4.7 Latch/Pulse Mapping Table**

The Latch/Pulse Mapping Table is P02\_LP. The parameters in this table determine which LP system points are used in this application. It also establishes the bridging between the LP system points and EDCO internal registers. It should be noted, the points that belong to other RIA applications are output points of the EDCO processor. The points that are owned by EDCO are control input points that can be used by other applications to influence or control the operation of EDCO. The number of records is the number of system LP points used in this application.

The following describes each parameter:

1. RIAIndex: Data Type is unsigned integer. This is the RIA application Index. This index determines which RIA application the point belongs to.
2. LPPoint: Data Type is unsigned integer. This is the LP point index of the RIA.
3. Linking: Data Type is signed integer. This is the index to the EDCO internal register.
4. CtrlType: Data Type is unsigned integer. This is the control type. This parameter is only applicable for system points not owned by EDCO. If value = 0, LP is configured as ON/OFF. If value is 1, it is configured as pulse duration type, Duration is determined by the value of the EDCO internal register. If value is greater than 1, the value is the ON and OFF duration in milliseconds. The number of pulses is determined by the value of the EDCO internal register.

## **4.8 Raise/Lower Mapping Table**

The Raise/Lower Mapping Table is P02\_RL. The parameters in this table determine which RL system points are used in this application. It also establishes the bridging between the RL system points and EDCO internal registers. It should be noted, the points that belong to other RIA applications are output points of the EDCO processor. The points that are owned by EDCO are control input points that can be used by other applications to influence or control the operation of EDCO. The number of records is the number of system RL points used in this application.

The following describes each parameter:

1. RIAIndex: Data Type is unsigned integer. This is the RIA application Index. This index determines which RIA application the point belongs to.
2. RLPoint: Data Type is unsigned integer. This is the RL point index of the RIA.
3. Linking: Data Type is signed integer. This is the index to the EDCO internal register.

## **4.9 Trip/Close Mapping Table**

The Trip/Close Mapping Table is P02\_TC. The parameters in this table determine which TC system points are used in this application. It also establishes the bridging

between the TC system points and EDCO internal registers. It should be noted, the points that belong to other RIA applications are output points of the EDCO processor. The points that are owned by EDCO are control input points that can be used by other applications to influence or control the operation of EDCO. The number of records is the number of system TC points used in this application.

The following describes each parameter:

1. RIAIndex: Data Type is unsigned integer. This is the RIA application Index. This index determines which RIA application the point belongs to.
2. TCPoint: Data Type is unsigned integer. This is the TC point index of the RIA.
3. Linking: Data Type is signed integer. This is the index to the EDCO internal register.
4. PulseDuration: Data Type is unsigned long. This is the ON pulse duration. This parameter is only applicable for system points not owned by EDCO.

## **4.10 Pattern Mapping Table**

The Pattern Mapping Table is P02\_PAT. The parameters in this table determine which PAT system points are used in this application. It also establishes the bridging between the PAT system points and EDCO internal registers. It should be noted, the points that belong to other RIA applications are output points of the EDCO processor. The points that are owned by EDCO are control input points that can be used by other applications to influence or control the operation of EDCO. The number of records is the number of system PAT points used in this application.

The following describes each parameter:

1. RIAIndex: Data Type is unsigned integer. This is the RIA application Index. This index determines which RIA application the point belongs to.
2. PATPoint: Data Type is unsigned integer. This is the PAT point index of the RIA.
3. Linking: Data Type is signed integer. This is the index to the EDCO internal register.

## **4.11 Double Point Select before Operate Mapping Table**

The Double Point Select before Operate Mapping Table is P02\_DPS. The parameters in this table determine which DPS system points are used in this application. It also establishes the bridging between the DPS system points and EDCO internal registers. It should be noted, the points that belong to other RIA applications are output points of the EDCO processor. The points that are owned by EDCO are control input points that can be used by other applications to influence or control the operation of EDCO. The number of records is the number of system DPS points used in this application.

The following describes each parameter:

1. RIAIndex: Data Type is unsigned integer. This is the RIA application Index. This index determines which RIA application the point belongs to.

2. DPSPoint: Data Type is unsigned integer. This is the DPS point index of the RIA.
3. Linking: Data Type is signed integer. This is the index to the EDCO internal register.

## **4.12 Analog Input Mapping Table**

The Analog Input Mapping Table is P02\_AI. The parameters in this table determine which AI system points are used in this application. It also establishes the bridging between the AI system points and EDCO internal registers. It should be noted, the points that belong to other RIA applications are input points to the EDCO processor. The points that are owned by EDCO are indication points (EDCO output points) that can be used by other applications. The number of records is the number of system AI points used in this application.

The following describes each parameter:

1. RIAIndex: Data Type is unsigned integer. This is the RIA application Index. This index determines which RIA application the point belongs to.
2. AIPoint: Data Type is unsigned integer. This is the AI point index of the RIA.
3. Linking: Data Type is signed integer. This is the index to the EDCO internal register.

## **4.13 Analog Output Mapping Table**

The Analog Output Mapping Table is P02\_AO. The parameters in this table determine which AO system points are used in this application. It also establishes the bridging between the AO system points and EDCO internal registers. It should be noted, the points that belong to other RIA applications are output points of the EDCO processor. The points that are owned by EDCO are control input points that can be used by other applications to influence or control the operation of EDCO. The number of records is the number of system AO points used in this application.

The following describes each parameter:

1. RIAIndex: Data Type is unsigned integer. This is the RIA application Index. This index determines which RIA application the point belongs to.
2. AOPoint: Data Type is unsigned integer. This is the AO point index of the RIA.
3. Linking: Data Type is signed integer. This is the index to the EDCO internal register.

## **4.14 Internal Register Initial Value Table**

The Internal Register Initial Value Table is P02\_REG. The number of records in this table is the number of EDCO internal registers. The parameters in this table determine the initial values of the registers.

The following describes each parameter:

1. Value: Data Type is signed long. This is the initial value of the register. The registers can be used to store constants as well.

## **4.15 Control Primitive Table**

The Control Primitive Table is P02\_PRI. The number of records in this table is the number of EDCO primitive instructions. This table contains the control algorithm in the form of series of EDCO control primitives.

The following describes each parameter:

1. Input1: Data Type is signed integer. This is first input of the control primitive. The value is the index to the EDCO internal register table.
2. Input2: Data Type is signed integer. This is second input of the control primitive. The value is the index to the EDCO internal register table.
3. PrimitiveTypeCode: Data Type is signed integer. This is the control primitive type code which represent the control primitive according to the following table:

<b>Primitive type Code</b>	<b>Primitive Type</b>
0	<b>NOP:</b> No Operation
1	<b>AND:</b> AND Gate
2	<b>OR:</b> OR Gate
3	<b>XOR:</b> Exclusive OR Gate
4	<b>NOT:</b> Inverter
5	<b>SR:</b> Set/Reset
6	<b>PLS:</b> Pulse
7	<b>TDE:</b> Timer Delay Energizer
8	<b>TDD:</b> Time Delay De-energizer
9	<b>ADD:</b> Addition
10	<b>SUB:</b> Subtraction
11	<b>MUL:</b> Multiplication
12	<b>DIV:</b> Division
13	<b>MOD:</b> Modulus
14	<b>GRT:</b> Greatest
15	<b>SML:</b> Smallest
16	<b>AVG:</b> Average
17	<b>DIF:</b> Absolute Difference
18	<b>SQR:</b> Square Root
19	<b>TAB:</b> Tap Analog position to Binary
20	<b>TBA:</b> Tap Binary position to Analog
21	<b>RMP:</b> Ramping
22	<b>CNT:</b> Counter

23	<b>SET:</b> Set value
24	<b>SEL:</b> Selector
25	<b>CMP:</b> Comparator
26	<b>EQU:</b> Equal
27	<b>ATB:</b> Analog To BCD conversion
28	<b>BTA:</b> BCD To Analog conversion
29	<b>BSR:</b> Binary Shift Right
30	<b>BSL:</b> Binary Shift Left
31	<b>0T1:</b> Zero to One transition
32	<b>WOF:</b> Weekly ON OFF

4. Output: Data Type is signed integer. This is output of the control primitive. The value is the index to the EDCO internal register table.

## **4.16 Configuration Table**

The CFG Table is P02\_CFG. The parameters in this table control the operation characteristics of EDCO. This table can be edited by user.

The following describes each parameter:

1. InitTimer: Data Type is unsigned longer. This parameter is the time to wait before start up the initialization of EDCO processor.
2. RampUpdateInterval: Data Type is unsigned longer. This parameter is the periodic interval for updating of output for the RMP control primitive in milliseconds.
3. MaxExecCnts: Data Type is unsigned char. This parameter is used to warn the user that the number of executions of the same primitive has been exceeded on one single change of value event. This feature is used to detect Feedback Oscillation.

## **5 EDCO Processor**

EDCO Processor is a piece of software resides in the Dynatrol Spectrum 32 RTU. This software performs the logic interlock, arithmetic calculations and sequential controls according to the EDCO control primitive configuration.

When there is a change of input, EDCO processes all the control primitives that are affected by this change of input. That is EDCO only wakes up when there is a change to one of the inputs and only the operators that are affected by this change of input will be processed. With this feature, EDCO is very efficient and able to perform time critical process controls.