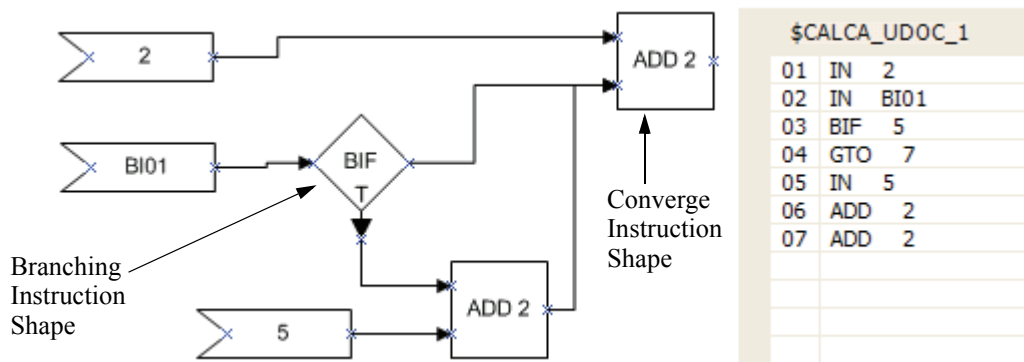


Logic Block Editor and Troubleshooting Tool



The Foxboro® Control Software Logic Block Editor and Troubleshooting Tool is a powerful tool for configuring STEPS within the I/A Series logic block types. It provides both graphical and text based program editing tools.

FEATURES

The Foxboro Control Software (FCS) Logic Block Editor enables the user to:

- ▶ Generate STEP logic using the traditional Reverse Polish Notation syntax
- ▶ Insert and delete lines of code to an existing program
- ▶ Annotate the STEP logic using comments notation
- ▶ Import and export STEP logic from an external source such as a text file
- ▶ Generate STEP logic segments using an equation editor
- ▶ Generate STEP logic using the instruction shape diagram-style graphical editor
- ▶ Insert text or graphical STEP logic segments from Library Objects
- ▶ Print the STEP logic in graphical or text form
- ▶ Provide live update displays of input, output and memory variables of deployed logic blocks (CALC, CALCA, LOGIC and MATH) on the graphical canvas.

BENEFITS

The Logic Block Editor provides substantial productivity enhancements for the:

- ▶ Project Manager who can now deliver a project complete with all of the documentation needed to support the on-going operation in a form that is familiar and understandable to the user.
- ▶ Design Engineer who now has a choice of tools from which to design including an RPN style macro language, an equation editor and a graphical design editor.
- ▶ Commissioning Engineer who can test the logic against either simulated or actual process conditions in the form in which it is created.
- ▶ Troubleshooting Engineer who can drill into complex process control & logic designs to determine which of many conditions has not yet been satisfied to enable process operations to continue.

OVERVIEW

The Logic Block Editor allows users to build and configure the programmable portion of the I/A Series logic blocks, which includes the CALC, CALCA, LOGIC and MATH blocks. Refer to the *Integrated Control Block Descriptions* (B0193AX) to learn about these Block types.

The Logic Block Editor provides a text-based and a graphical-based editor for editing code in logic blocks. Users may switch between them on-the-fly.

Text Editor

The Text Editor enables the user to add, insert or delete instructions specified in Reverse Polish Notation (RPN). The editor assigns the various STEPS to the appropriate logic block parameters once the editing session is complete.

The Text Editor is shown in Figure 1.

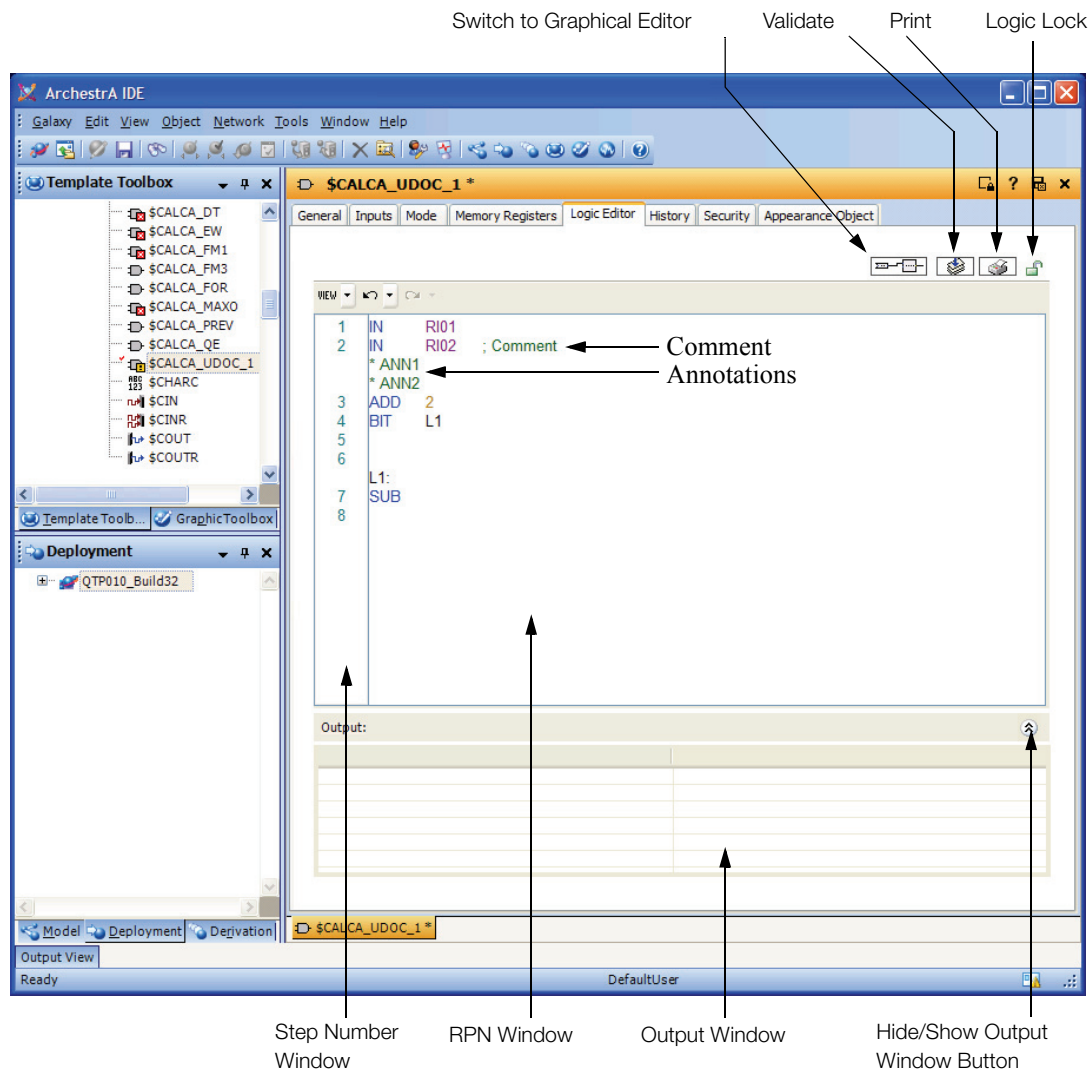


Figure 1. Text Editor

Reference labels can be added to a statement line to facilitate the design of conditional branching.

Comments can be added at the end of a statement or as a separate statement line and are deployed to the target controller; are visible in the HMI source code display and consume a STEP parameter in the CALC Block.

Comments are marked by a semi-colon.

04 ; This is a standard comment on one line.

05 SUB ; This is a comment on the same line as an executable statement.

Annotations can also be interspersed with the statements. Annotation lines are not deployed to the target controller and therefore don't consume a STEP. Annotation lines are marked by an asterisk.

02 IN RI02

* This is an annotation between two statements

03 ADD 2

The Text Editor supports a Formula Editor (Figure 2) to enable the user to develop statements as regular expressions with nested parentheses and insert the RPN-resolved results into the statement list.

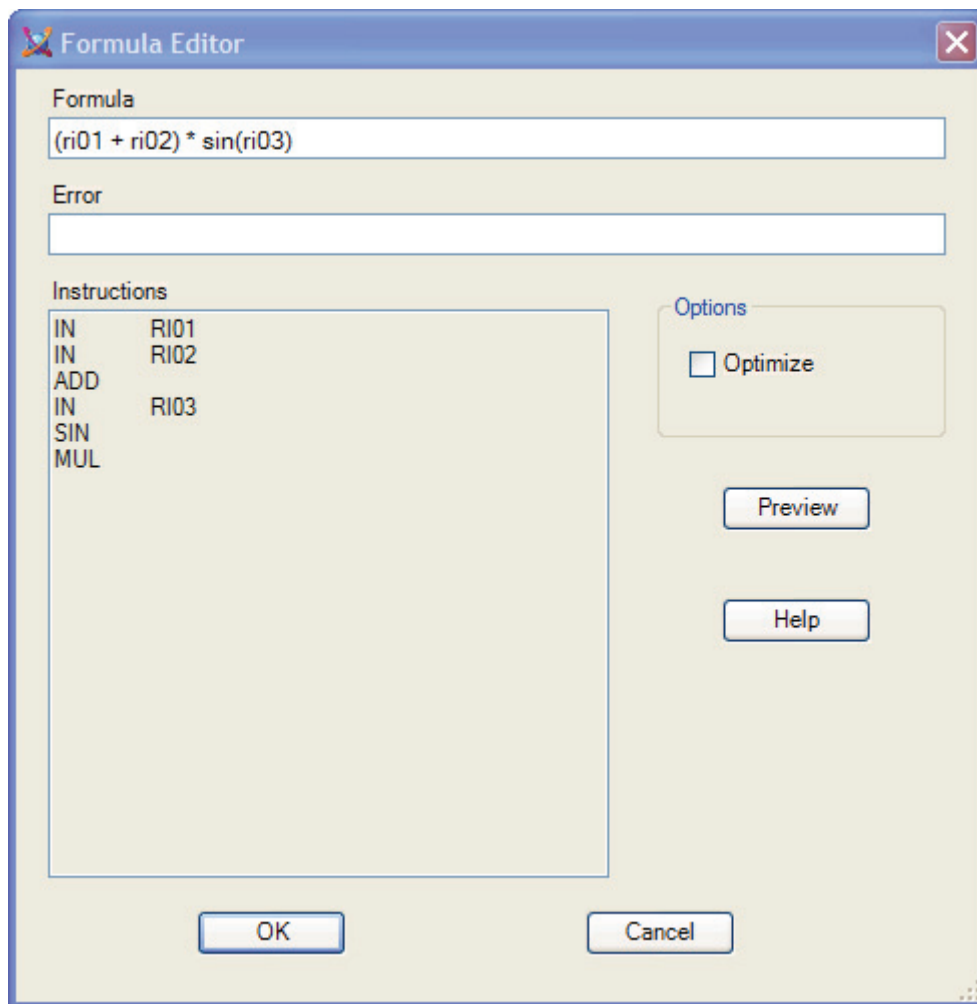


Figure 2. Formula Editor Dialog

The Formula Editor dialog allows users to enter and maintain formulas in regular expression format using:

- ▶ arithmetic operations: + , - , * , / ;
- ▶ logic operations: | , & , ^ , ~ , ! ;
- ▶ Nested parentheses (,)
- ▶ Block parameters.
- ▶ Constants
- ▶ Functions:
 - ADD, SUB, MUL, DIV, SQR, SQRT
 - DEC, IDIV, IMOD, INC, MEDN, TRC
 - ABS, AVE, MAX, MIN
 - ALN, ALOG, EXP, LN, LOG
 - ACOS, ASIN, ATAN, COS, SIN, TAN
 - AND, NAND, NOR, NOT, NXOR, OR, XOR

The Text Editor can also be used to create code snippets that are preserved as Library Objects and available to be inserted into a logic block's statement list.

Logic can be imported from an external text file and inserted either directly into a logic block's statement list or into a Library Object.

The Text Editor can print representations of the logic block's code in RPN format with its associate input parameters.

Graphical Editor

The Graphical Editor is a Visio-based drawing tool that supports a wide variety of drawing tools via menu picks as shown in Figure 3.

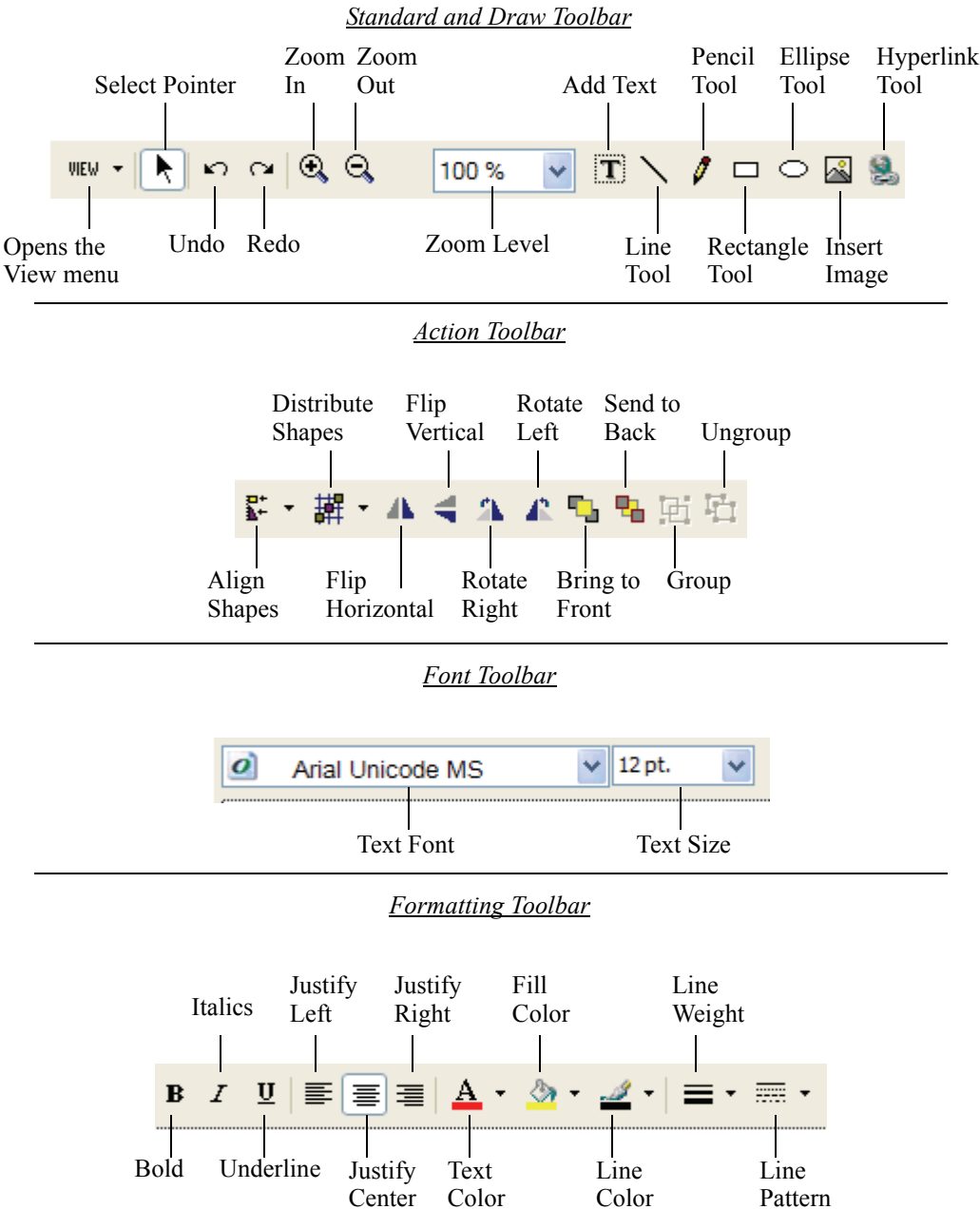


Figure 3. Graphical Editor Drawing Tools

The Graphical Editor enables the user to add, insert or delete STEP instructions represented as graphical instruction shapes.

These instruction shapes are dragged from the Instruction Tree Window to the Graphical Editor Canvas and interconnected via line drawing between their respective input and output parameters.

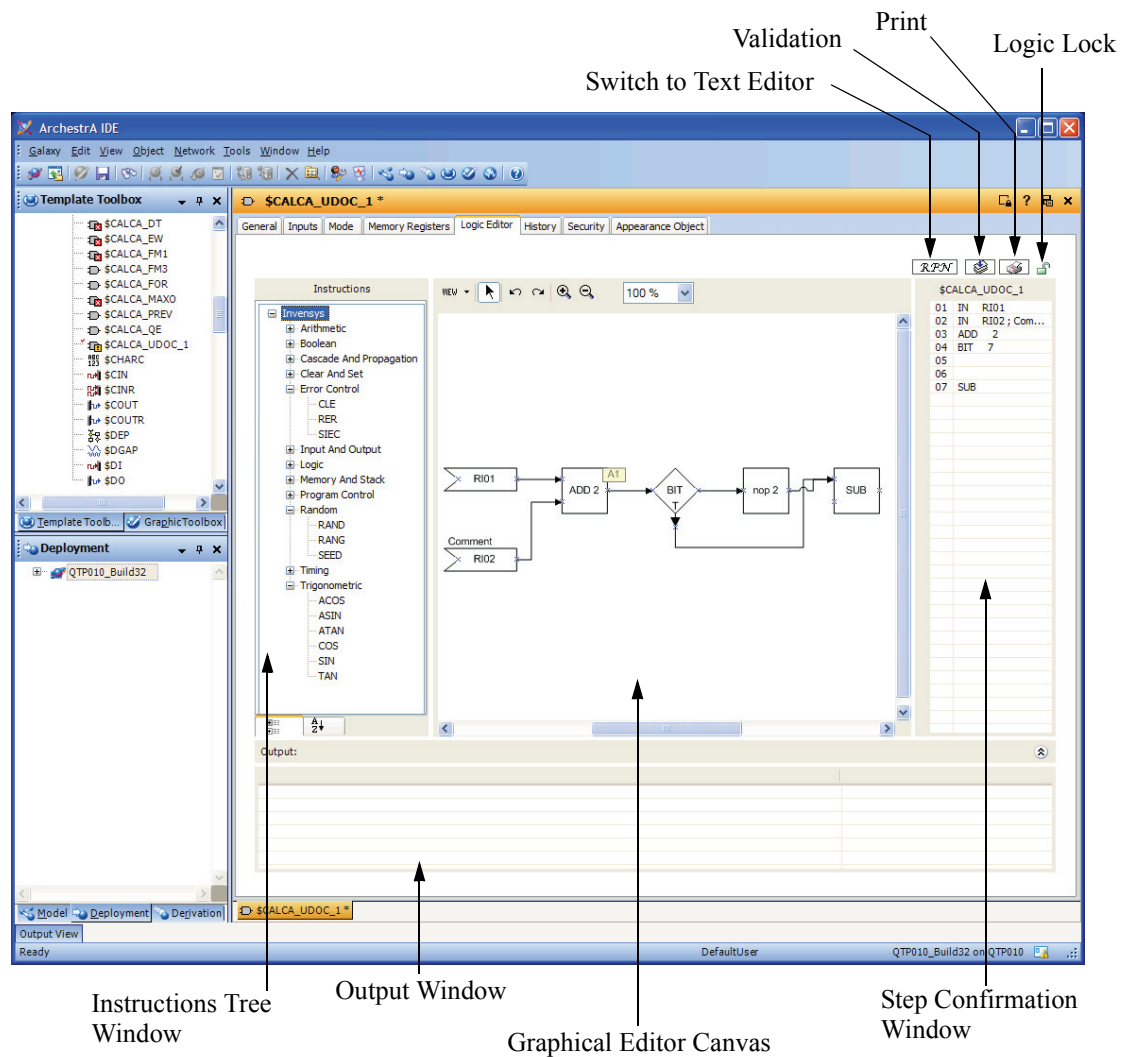


Figure 4. Graphical Editor Canvas

The editor assigns the instructions to the appropriate logic block's STEP parameters as the editing session is conducted. The actual RPN logic being created is shown in the STEP Confirmation Window.

A drawing consists of two standard types of representations. The first type are Input, Memory and Output registers while the second type are instructions. These are represented in a drawing as shown in Figure 5.

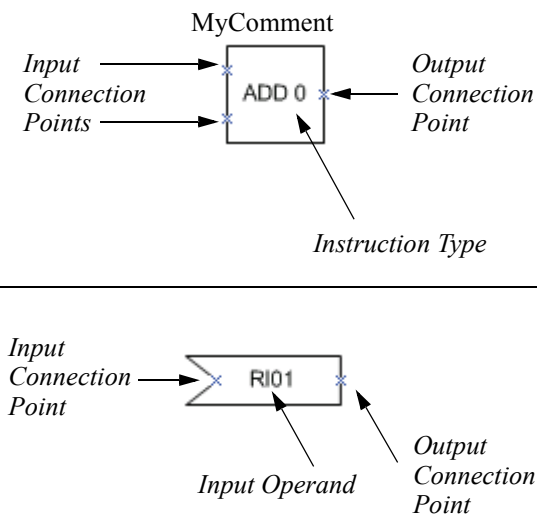


Figure 5. Input, Memory and Output Registers

The user can also create formulas as instruction shapes with no connected inputs and one connectable output, as shown in Figure 6.

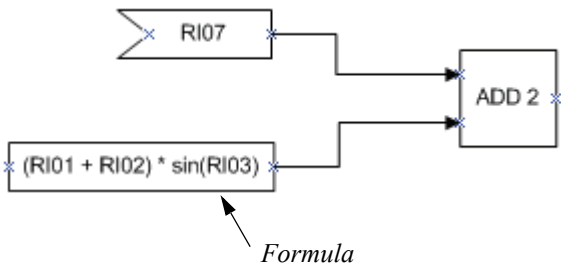


Figure 6. Formula as an Instruction Shape

Formulas are constructed in the same manner as in the Text Editor.

The Graphical Editor can import and export a logic diagram from or to a Visio (*.vsd) file.

With this editor, the user can create code snippets as graphical diagrams that are preserved as Library Objects and available to be added to the canvas of the graphical editor by drag-n-drop means.

Logic diagrams can be printed in WYSIWYG format. Optionally, the diagram's annotations and/or block inputs may be included as well.

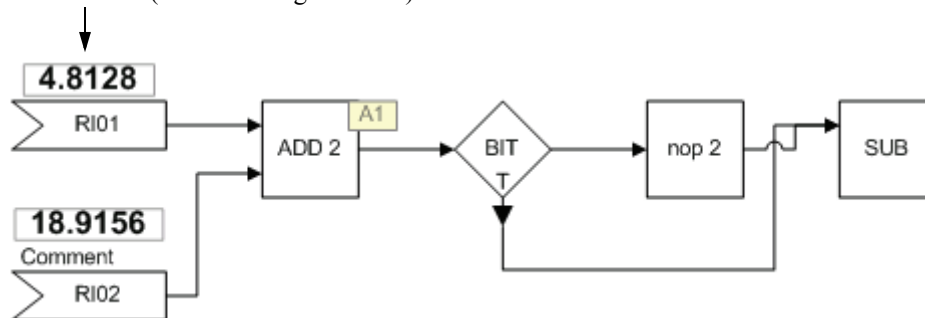
The editor supports printing complex logic diagrams with up to forty interconnected instruction shapes on 11" x 17" paper with clearly marked function types.

LIVE DATA MODE

Live Data mode in the Graphical Editor allows users to view live data values for each input, output and memory register when the logic block is deployed and running in a target controller. The value of each input, output and memory register instruction shape is displayed.

Real, Integer and Boolean values are shown in live data fields adjacent to their instruction shapes. Alternatively, Boolean [True/False] values can be illustrated by color coding the function block whose output represents a Boolean value. These representations are illustrated in Figure 7.

Live Data Field (for Real/Integer Values)



Boolean Instruction Shapes
With Colors Configured (Green = True, Red = False)

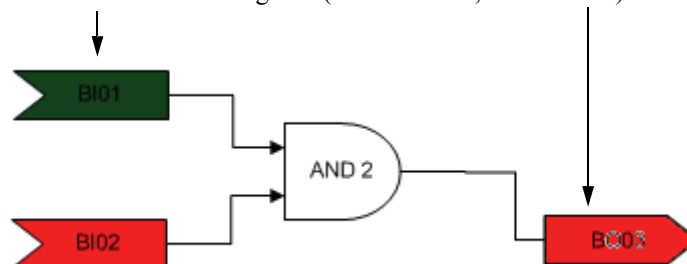


Figure 7. Live Data Fields

The layout of a graphical display can be simulated in Live Data Preview Mode prior to the logic block being deployed to a target controller. In this mode, a random number generator feeds values to the Input, Output and Memory representations as live update data to assist the user in determining the appropriate layout prior to deployment. The values generated in this preview mode do not correspond to the execution of the actual STEP logic.

HARDWARE AND SOFTWARE REQUIREMENTS

The Logic Block Editor and Troubleshooting Tool are components of the Foxboro Control Software (FCS) Configuration Tools.

For detailed instructions on how to use the Logic Block Editor and Troubleshooting Tool, refer to *Foxboro Control Software Logic Block Editor and Troubleshooting Tool* (B0750BL).

For more information on the FCS Configuration Tools, refer to the Product Specification Sheet *Foxboro Control Software Configuration Tools* (PSS 21S-10B3-B3).

For detailed information on the logic blocks (CALC, CALCA, LOGIC, MATH), refer to *Integrated Control Block Descriptions* (B0193AX).

