# DXMR50-LB Programmable Logic Block Configuration Software

Original Instructions
p/n: 252379 Rev. B
04-Mar-26
© Banner Engineering Corp. All rights reserved. www.bannerengineering.com

ПРОМЭЛЕКТРОНИКА/ sales@prom-elec.com

# Contents


## Chapter 1 Overview ..... 3
DXMR50-LB-2M Models ..... 3
DXMR50-LB-2PA Models ..... 4
DXMR50-LB-MPA Models ..... 4

## Chapter 2 Configuration Instructions ..... 6
Register Flow ..... 6
DXMR50 Configuration Software Overview ..... 7
Save and Upload the Configuration File ..... 7
Starting Configuration ..... 8
Data Stores ..... 8
Action Rules ..... 9
Threshold Rules ..... 10
Register Copy Rules ..... 11
Math and Logic Rules ..... 11
Control Logic Rules ..... 11
Trending and Filtering Rules ..... 14
Tracker Rules ..... 15
Decoder Rules ..... 16
Delay and Scaler Rules ..... 16
IO Configuration ..... 16
Create a Read Rule for the DXMR50-LB-MPA Series ..... 16
Create a Read Rule for the DXMR50-LB-2M Series ..... 17
Create a Write Rule for the DXMR50-LB-MPA Series ..... 17
Create a Write Rule for the DXMR50-LB-2M Series ..... 18
Modbus Settings ..... 18
Advanced Tab ..... 19
Configuration Description ..... 19
Read/Write Register Values ..... 20
Reprogram ..... 20
Examples ..... 21
Configure the DXMR50-LB Logic Block ..... 21
Save and Upload the Configuration File ..... 7
Configure a DXMR50-LB-2PA-2P for AND Logic ..... 22

## Chapter 3 Product Support ..... 25
DXMR50-LB Configuration Software Release Notes ..... 25
Banner Engineering Corp. Software Copyright Notice ..... 25


© Banner Engineering Corp. All rights reserved. www.bannerengineering.com

Chapter Contents

DXMR50-LB-2M Models ..... 3
DXMR50-LB-2PA Models ..... 4
DXMR50-LB-MPA Models ..... 4

# Chapter 1 Overview

Banner's DXMR50-LB Programmable Logic Block consolidates data from multiple sources to provide local data processing and logic functionality to solve applications independent of a higher-level control system.

The DXMR50-LB has two input ports, a programming port, and an output port. Data is collected into the internal logic controller to facilitate edge processing.

The programming port is a male M12 connection that also provides common power and ground to all M12 Modbus ports. Use the programming port and the BWA-UCT-900 adapter cable to connect to a PC to use the DXMR50 Configuration Software to configure the device.

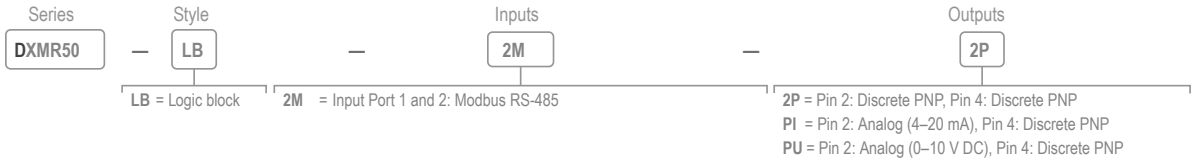
*DXMR50-LB input and output ports*




Program the DXMR50-LB's logic controller using action rules and the DXMR50 Configuration Software.

- Thresholds (IF/THEN/ELSE) with timers, minimum on/off time
- Math/logic rules (arithmetic and bitwise operators)
- Control logic (logical operators and SR/T/D/JK flip-flops)
- Copy and decoding rules
- Trending (multiple averaging filters)
- Tracking (counts, on/off times)
- Delays (On Delay, Off Delay, On One-Shot, Off One-Shot, Pulse Stretch On, Pulse Stretch Off, Totalizer On, Totalizer Off)
- Scalers

# DXMR50-LB-2M Models




*DXMR50-LB-2M Ports*

| Port | 2P Model | PI Model | PU Model |
|---|---|---|---|
| Port 0/Power | Pin 1 and 3: Power<br>RTU server for configuration and register access | | |

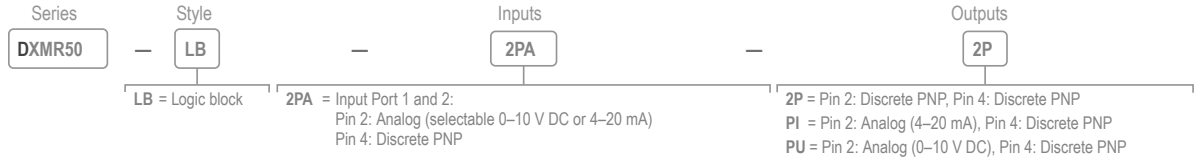
Continued on page 4


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

Continued from page 3

| Port | 2P Model | PI Model | PU Model |
|---|---|---|---|
| Input Port 1<br>Input Port 2 | Pin 1 and 3: Power<br>Pin 2: Modbus RS-485 (+)<br>Pin 4: Modbus RS-485 (-) | | |
| Output Port | Pin 2: Discrete PNP<br>Pin 4: Discrete PNP | Pin 2: Analog (4-20 mA)<br>Pin 4: Discrete PNP | Pin 2: Analog (0-10 V)<br>Pin 4: Discrete PNP |

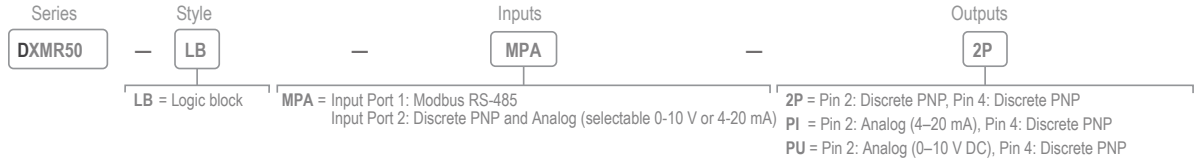
# DXMR50-LB-2PA Models




*DXMR50-LB-2PA Ports*

| Port | 2P Model | PI Model | PU Model |
|---|---|---|---|
| Port 0/Power | Pin 1 and 3: Power<br>RTU server for configuration and register access | | |
| Input Port 1<br>Input Port 2 | Pin 1 and 3: Power<br>Pin 2: Analog (selectable 0-10 V or 4-20 mA)<br>Pin 4: Discrete PNP | | |
| Output Port | Pin 2: Discrete PNP<br>Pin 4: Discrete PNP | Pin 2: Analog (4-20 mA)<br>Pin 4: Discrete PNP | Pin 2: Analog (0-10 V)<br>Pin 4: Discrete PNP |

# DXMR50-LB-MPA Models




*DXMR50-LB-MPA Ports*

| Port | 2P Model | PI Model | PU Model |
|---|---|---|---|
| Port 0/Power | Pin 1 and 3: Power<br>RTU server for configuration and register access | | |
| Input Port 1 | Pin 1 and 3: Power<br>Pin 2: Modbus RS-485 (+)<br>Pin 4: Modbus RS-485 (-) | | |
| Input Port 2 | Pin 1 and 3: Power<br>Pin 2: Analog (selectable 0-10 V DC or 4-20 mA)<br>Pin 4: Discrete PNP | | |

Continued on page 5


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

Continued from page 4

| Port | 2P Model | PI Model | PU Model |
|---|---|---|---|
| Output Port | Pin 2: Discrete PNP<br>Pin 4: Discrete PNP | Pin 2: Analog (4-20 mA)<br>Pin 4: Discrete PNP | Pin 2: Analog (0-10 V)<br>Pin 4: Discrete PNP |


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

Chapter Contents

Register Flow ..... 6
DXMR50 Configuration Software Overview ..... 7
Save and Upload the Configuration File ..... 7
Starting Configuration ..... 8
Data Stores ..... 8
Action Rules ..... 9
IO Configuration ..... 16
Advanced Tab ..... 19
Examples ..... 21

# Chapter 2 Configuration Instructions

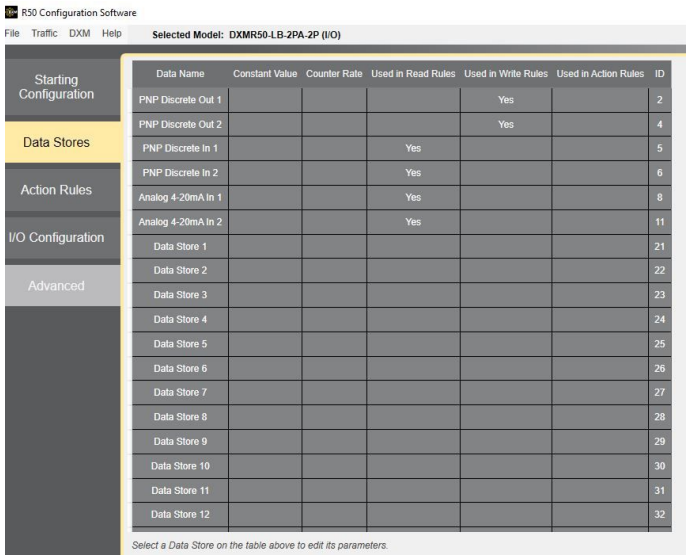
## Register Flow

The DXMR50-LB register data flows through the local registers, which are data storage elements within the processor. Use the DXMR50-LB Logic Block Configuration Software to program the controller to move register data from the data store registers to the I/O ports.

The DXMR50-LB moves and manipulates data using an on-board collection of uint32 registers referred to as data stores. Each DXMR50-LB model has 20 general-purpose data stores that can be used in tandem with action rules to transform data and achieve a wide range of logic operations.

Device-specific input and output data stores are also defined for each DXMR50-LB model; the logic established using action rules should accept data from the input data store(s) and ultimately place the results in the output data store(s).

*Input/output data stores*




Note: Users familiar with non-R50 DXM models may note that data stores behave much like the local registers configured on those devices. Data stores serve as an abstraction for these registers, enforcing standards regarding register quantity and use (input, output, or generic) for the DXMR50 device family.





© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

# DXMR50 Configuration Software Overview

Use the DXMR50-LB Logic Block Configuration Software to customize your DXMR50-LB Programmable Logic Block configuration and process data from the controller.

Download the latest version of all configuration software from www.bannerengineering.com. For detailed instructions on how to program any DXMR50-LB Logic Block, download the configuration software manual, p/n 252379.

This configuration file defines parameters for the DXMR50-LB and saves the configuration in an XML file on the PC. After the configuration file is saved, connect the DXMR50-LB to your PC using adapter cordset **BWA-UCT-900** to upload the XML configuration file to the DXMR50-LB for operation.

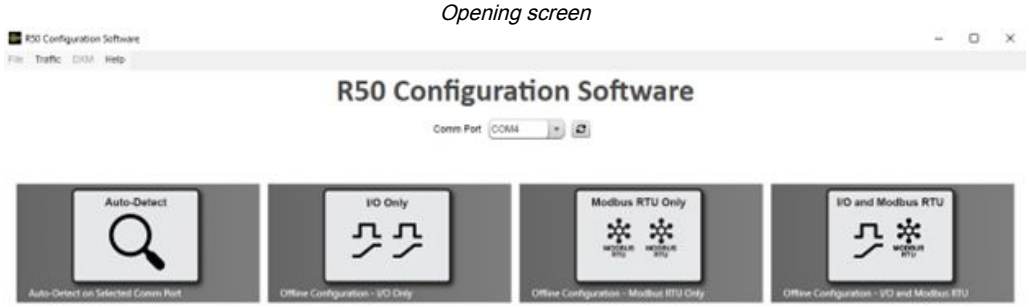
Use the configuration software either while connected to a DXM or as a standalone configuration software.

To detect the DXMR50-LB that you have connected to your PC using the BWA-UCT-900 adaptor, select **Auto Detect** to load the **Starting Configuration** screen.

The top-level menus are similar to other Windows programs: **File**, **Traffic**, **DXM**, and **Help**.

- Use the **File** menu to manage the loading and saving of the XML configuration file created by the configuration software.
- Use the **Traffic** menu to view data traffic on the serial bus.
- Use the **DXM** menu to send and retrieve XML configuration files to or from a connected DXM.
- Use the **Help** menu to load the configuration software instruction manual in your default PDF viewer.

*Opening screen*




To create an XML file in offline mode (without a DXMR50-LB connected to the PC), select the **Offline Configuration** options for DXMR50-2P series (I/O Only), the DXMR50-2M series (Modbus RTU Only), or the DXMR50-MPA series (I/O and Modbus RTU)

# Save and Upload the Configuration File

After making any changes to the configuration, you must save the configuration files to your computer, then upload it to the device.

Changes to the XML file are not automatically saved. Save your configuration file before exiting the software and before sending the XML file to the device to avoid losing data. If you select **DXM › Send XML Configuration to DXM** before saving the configuration file, the software will prompt you to choose between saving the file or continuing without saving the file.

1. Save the XML configuration file to your hard drive by going to the **File › Save As** menu.
2. Go to the **DXM › Send XML Configuration to DXM** menu.

*Status indicator bar*




   - If the Application Status indicator is red, close and restart the DXMR50-LB Logic Block Configuration Software, unplug and re-plug in the cable and reconnect the DXM to the software.
   - If the Application Status indicator is green, the file upload is complete.
   - If the Application Status indicator is gray and the green status bar is in motion, the file transfer is in progress.

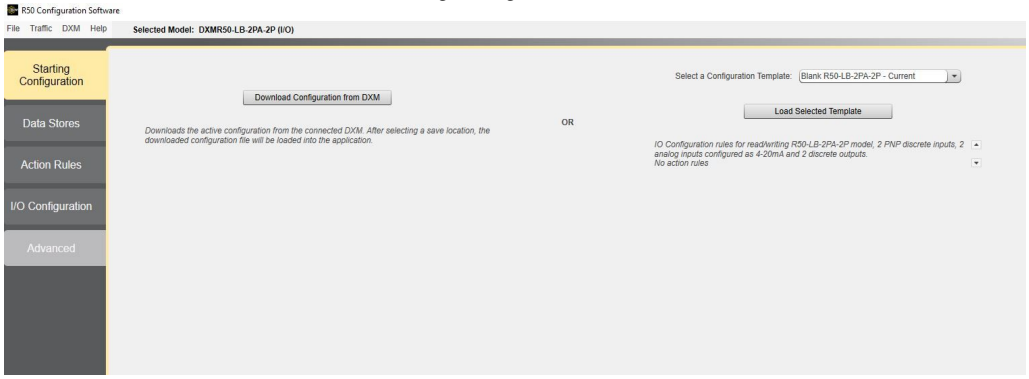
   After the file transfer is complete, the device reboots and begins running the new configuration.


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

# Starting Configuration

Use the **Starting Configuration** screen to download the active configuration from the connected DXM or to start a new configuration.

*Starting Configuration screen*

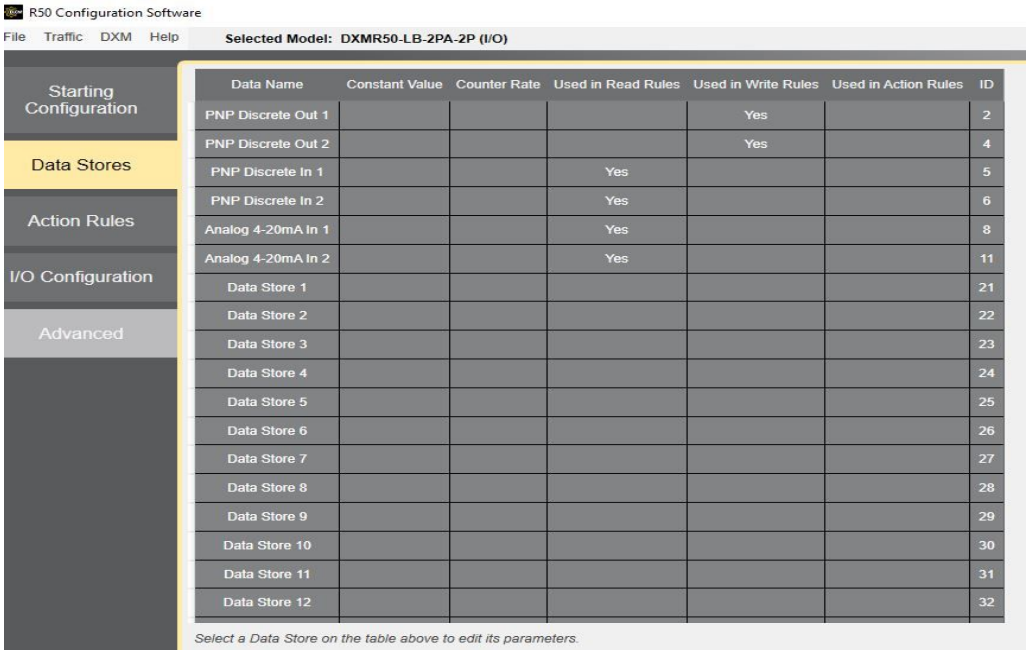



To start a new configuration, select a stock template from the drop-down list and select **Load Selected Template**. A description of the selected stock template is displayed below these controls. This option provides preloaded IO-Configuration rules to read/write the specific DXMR50 model you have selected and automatically assigns data store names.

# Data Stores

Use the **Data Stores** screen to view a list of defined registers or to make changes to registers and show where the register is used in the defined configuration.

*Data Stores screen*




| Data Name | Constant Value | Counter Rate | Used in Read Rules | Used in Write Rules | Used in Action Rules | ID |
|---|---|---|---|---|---|---|
| PNP Discrete Out 1 | | | | Yes | | 2 |
| PNP Discrete Out 2 | | | | Yes | | 4 |
| PNP Discrete In 1 | | | Yes | | | 5 |
| PNP Discrete In 2 | | | Yes | | | 6 |
| Analog 4-20mA In 1 | | | Yes | | | 8 |
| Analog 4-20mA In 2 | | | Yes | | | 11 |
| Data Store 1 | | | | | | 21 |
| Data Store 2 | | | | | | 22 |
| Data Store 3 | | | | | | 23 |
| Data Store 4 | | | | | | 24 |
| Data Store 5 | | | | | | 25 |
| Data Store 6 | | | | | | 26 |
| Data Store 7 | | | | | | 27 |
| Data Store 8 | | | | | | 28 |
| Data Store 9 | | | | | | 29 |
| Data Store 10 | | | | | | 30 |
| Data Store 11 | | | | | | 31 |
| Data Store 12 | | | | | | 32 |

The **Edit Data Store** tab presents all the parameters for a single register. Select the register to modify using your keyboard's arrow keys or by clicking within the corresponding row in the Data Stores table. Use this section to assign the register a name, value type, or scaling. The value type and scaling options are drop-down lists.


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

*Edit Data Store tab*




**Value**

None. The register has a value of 0 by default and can be modified by any source that has permission to set register values.

Constant. Forces the register to be set to the specified value. Used to compare values when using action rules. Constant values cannot be overwritten by any source that would normally have permissions to modify register values.

Counter. The value increments with the specified frequency (every 100 ms or every 1 second). Write to the data store registers to start the timer at a specific value.

**Scaling**

The scaling parameter changes the viewing of register data.

Scaling first applies a scale (add, subtract, multiple, divide, or modulo (remainder in division)), then adds an offset value. Select **Apply offset before scale value** to add the offset first and then apply the scale.

# Action Rules

Action rules perform simple logic functions and manipulation of data store register data.

Action rule processing is autonomous from other data store functions.

*Action Rules screen*




- **Control Logic** rules are binary rules, with the results being either 0 or 1
- **Decoder** rules copy a subset of the bits in one register to a specified bit index within another register
- **Delay** rules allow for time-based logic between source and destination registers
- **Math/Logic** rules deal with 32-bit register logic with results from 0 to 4,294,967,295
- **Register Copy** rules copy the contents of one register to another
- **Scaler** rules transform the domain of register values between source and destination registers
- **Threshold** rules create event-driven conditions
- **Tracker** rules monitor a register value and store the result of a function in another register
- **Trending** rules find average, minimum, and maximum values

Each action rule is processed in order, similar to the read/write rules. The groups of action rules are solved in the following order:

1. Calculate (Math), continually processed
2. Copy rules, processed only when a change of state is detected on the source register
3. Threshold rules, continually processed
4. Control Logic, continually processed
5. Trending, continually processed
6. Trackers, continually processed.


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

All Action Rules also have **Clone Selected Rule** or **Delete Selected Rule** commands to quickly copy/paste or delete the selected rule.

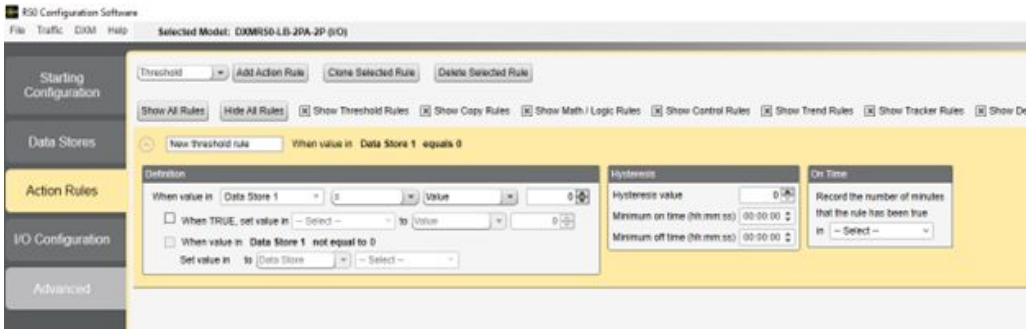
## Threshold Rules

A threshold rule triggers event messages to be sent to the cloud (event-driven push), pushes events to be stored to the local event log, and creates a standard push message to send all defined registers to the cloud.

A threshold rule creates a simple IF-THEN-ELSE comparison for a register to determine its value and set another register to indicate if the rule is true or false. Note that you must select a true option to set a false option. The definition section of the threshold rule sets the comparison and values. The definitions of the threshold rules can also be defined by the optional parameters, Hysteresis, On Time, and Logging options.

If selected, the false option (else condition) is applied. This creates an IF-THEN-ELSE structure. To remove the ELSE condition, deselect the false option or set the result register to itself.

*Threshold Rules*




1. Write the threshold rule as a sentence.
2. Select the data store register, operator, and value from the drop-down lists.
3. Define the value of the register when the statement is true and when it is false.

### Value/Register

When **Value** is selected, provide a fixed number in the following entry box.

When **Register** is selected, provide a data store register address in the following entry box.

The contents of the data store register are used in the threshold rule. When **Same as Source** is selected (else statement), the contents of the register defined in the **If** comparison is used as the result.

### Hysteresis

Hysteresis is an optional parameter that is enabled only when values are non-zero. How hysteresis is applied depends on the comparison.

For a test that becomes true **if greater than**, the test will not return to false until the register is less than the test value by a margin of at least this hysteresis value.

If a test becomes true **if less than**, the test will not return to false until the register is greater than the test value by a margin of at least this hysteresis value.

**Minimum On Time** and **Minimum Off Time** are time-based parameters that govern how long a statement must be true or false to activate the output register.

### On Time

Selecting **On Time** stores in a register how long (in minutes) the threshold rule has been true.

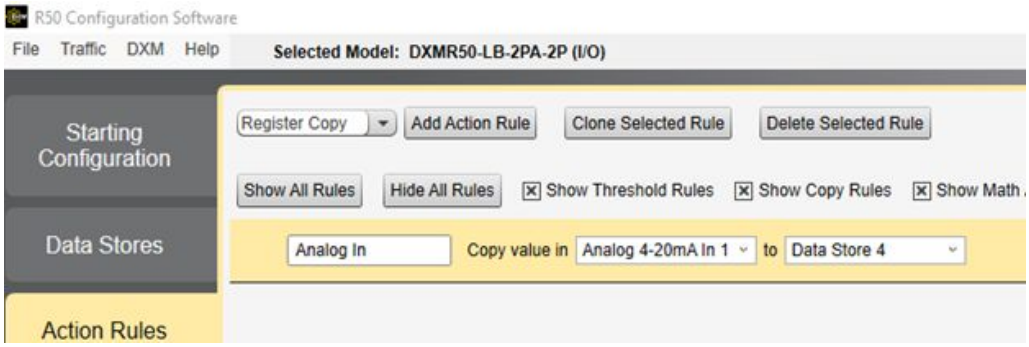
The **On-Time** value is shown in the data store register in minutes but maintains the internal counter in seconds. This results in the accuracy of seconds, not rounding of the value stored in the register.


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

## Register Copy Rules

Use the Register Copy screen to copy one data store register into another data store register.

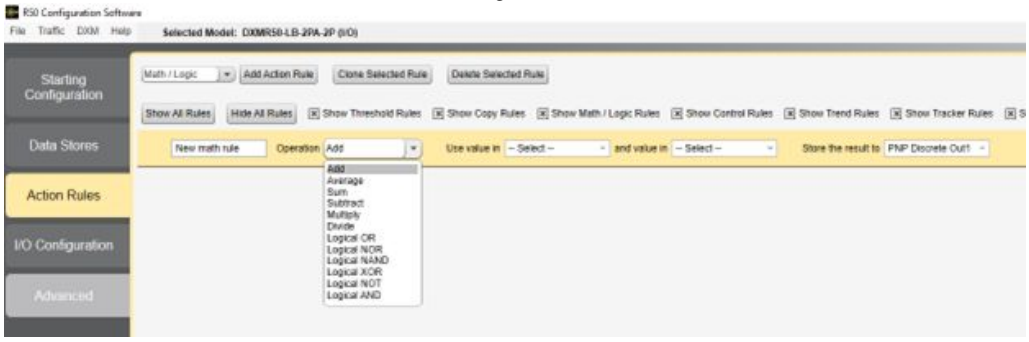
*Register Copy screen*




## Math and Logic Rules

Math and logic rules deal with 32-bit register logic. In contrast, control logic rules are binary rules, with the results being either 0 or 1.

*Math and Logic Rules screen*




The average, sum, and logic operations are valid for ranges of registers. The data store registers are unsigned 32-bit integers. All math and logic functions can operate on all 32-bits.

The data store register operations are:

- Add. Adds two data store registers and stores the result in a data store register.
- Average. Averages the values of multiple registers and stores the result.
- Divide. Divides data register 1 by data store register 2 and stores the result. Dividing by zero results in a zero.
- Logic NOT. Performs a bit-wise one's complement on a register and stores the result.
- Logic OR, AND, NOR, NAND, XOR. Performs bit-wise logic functions on multiple registers and stores the result. Bit-wise logic functions operate on all 32-bits of the registers. To perform a logic function on a simple 0 or 1 in the registers, use the control logic rules.
- Multiply. Multiplies one register by another and stores the result.
- Sum. Adds multiple registers and stores the result in a data store register.
- Subtract. Subtracts register 2 from register 1 and stores the result. For negative numbers, the results are in two's complement form.

## Control Logic Rules

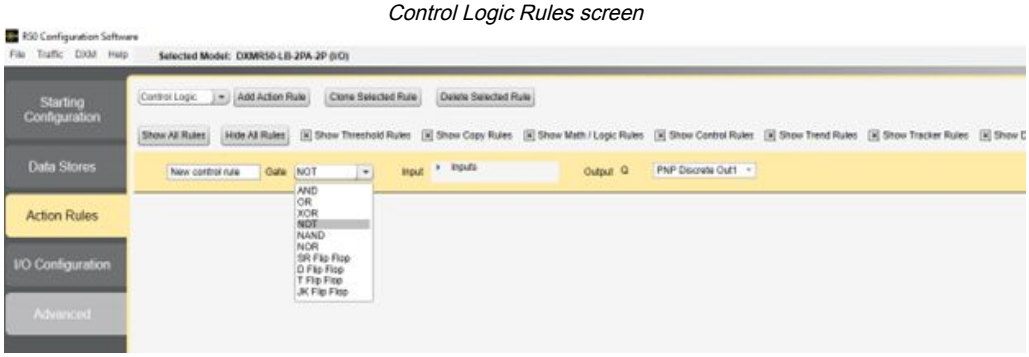
Control logic rules are binary rules, with the results either 0 or 1.

Some control logic rules create memory elements using the data store registers as inputs and outputs. A value of zero in a register is defined as 0 in the control logic rule, and a non-zero value in a register is defined as a 1 value. Control logic rules are evaluated when an input value changes state.

Memory elements include J-K Flip/Flop, S/R Flip/Flop, D Flip/Flop, and T Flip/ Flop.


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

*Control Logic Rules screen*




*Control Logic*

| Control Logic | | | |
|---|---|---|---|
| **Function** | **In 1** | **In 2** | **Q out** |
| AND | 0 | 0 | 0 |
| | 1 | 0 | 0 |
| | 0 | 1 | 0 |
| | 1 | 1 | 1 |
| OR | 0 | 0 | 0 |
| | 1 | 0 | 1 |
| | 0 | 1 | 1 |
| | 1 | 1 | 1 |
| XOR | 0 | 0 | 0 |
| | 1 | 0 | 1 |
| | 0 | 1 | 1 |
| | 1 | 1 | 0 |
| NOT | 0 | - | 1 |
| | 1 | - | 0 |
| NAND | 0 | 0 | 1 |
| | 1 | 0 | 1 |
| | 0 | 1 | 1 |
| | 1 | 1 | 0 |
| NOR | 0 | 0 | 1 |
| | 1 | 0 | 0 |
| | 0 | 1 | 0 |
| | 1 | 1 | 0 |

**NOTE:** * Output Qn is an inverted output from the Q output and represents optional inputs or outputs. Leave the optional inputs or outputs connected to 0 when not used.

### J-K Flip/Flop

If J and K are different, the output Q takes the value of J. The **Enable**, **Clock**, and **Qn** connections are not required for operation.


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

*J-K Flip/Flop*

| *Enable | J | K | * Clock | Q | Qn * |
|---|---|---|---|---|---|
| 0 | X | X | X | No change | No change |
| 1 | X | X | 0 or 1 or ↓ | No change | No change |
| 1 | 0 | 0 | ↑ | No change | No change |
| 1 | 1 | 0 | ↑ | 1 | 0 |
| 1 | 0 | 1 | ↑ | 0 | 1 |
| 1 | 1 | 1 | ↑ | No change | No change |

If an input is zero it is false (0), and if a value is non-zero (anything but zero) it is considered true (1). All but one requires two inputs; the NOT rule only has one input. Some of the control logic rules provide basic non-memory logic functions: AND, OR, XOR, NOT, NAND and NOR. These basic logic functions are based on zero and non-zero values and result in a 0 or 1.

Optionally, the rules can have a **Clock** input and an **Enable** input. The **Clock** input determines the time (0-1 rising edge) the rule is evaluated. The **Enable** input enables or disables the logic function.

- AND—Output is high (1) only if both inputs are non-zero
- OR—Output is high (1) if either input is non-zero
- XOR—Output is high (1) if either, but not both, inputs are non-zero
- NOT—Input is inverted to the output
- NAND—Output is low (0) if both inputs are high (> 0)
- NOR—Output is high (1) if both inputs are zero (0)

### Set/Reset (S/R) Flip/Flop

The Set/Reset Flip/Flop, or S/R Latch, is the most basic memory device. The S input sets the output to 1, and the R input sets the output to 0. Selecting **Insert input** turns the inputs to low active, so a 0 on the S input set the Q output to 1. The truth table below is defined without the invert flag.

*Set/Reset (S/R) Flip/Flop*

| S | R | Q | Qn * |
|---|---|---|---|
| 1 | 0 | 1 | 0 |
| 0 | 0 | No change | No change |
| 0 | 1 | 0 | 1 |
| 1 | 1 | Invalid [1] | |

### Delay (D) Flip/Flop

The Delay (D) Flip/Flop takes the D input to the outputs at the next rising edge of the clock. All other states of clock have no effect on the outputs. The **Enable** input is optional. A **Clock** input and a **D** input changing at the same time results in an undetermined state.

**Invert Inputs**—Typically a input value of "1" will be evaluated as true (active), selecting "Invert inputs" will result in a "0" value to be considered true(active).

**Flip/Flop is R dominated**—xxx

Table references:

- X = doesn't matter what the value is.
- <up arrow>, <down arrow> = signify a rising or falling edge of the clock input.
- * = optional input or output

[1] The Set input and Reset input active at the same time is an illegal condition and should not be used.


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

*Delay (D) Flip/Flop*

| *Enable | D | Clock | Q | Qn * |
|---|---|---|---|---|
| 0 | X | X | No change | No change |
| 1 | X | 0 or 1 or ↓ | No change | No change |
| 1 | 0 | ↑ | 0 | 1 |
| 1 | 1 | ↑ | 1 | 0 |

### Toggle (T) Flip/Flop

The Toggle (T) Flip/Flop toggles the state of the output if the input is 1. The **Enable** input is optional. A **Clock** input and a **T** input changing at the same time results in an undetermined state.

*Toggle (T) Flip/Flop*

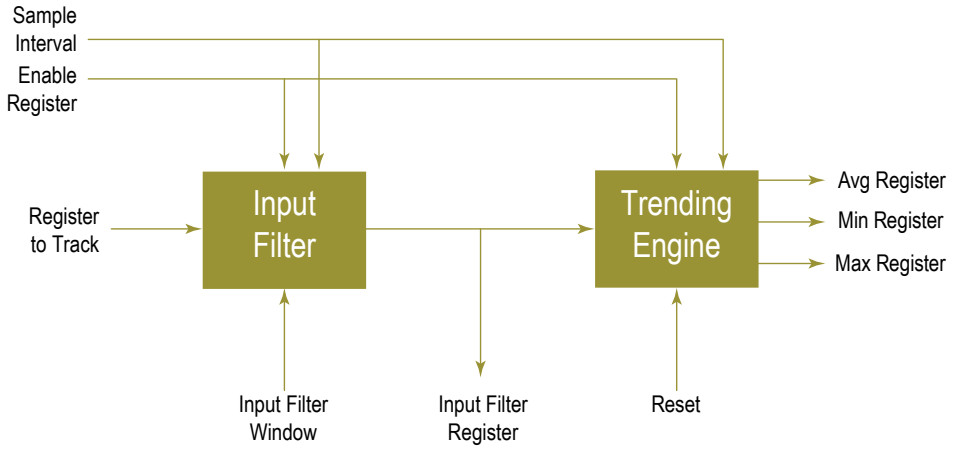
| *Enable | T | Clock | Q | Qn * |
|---|---|---|---|---|
| 0 | X | X | No change | No change |
| 1 | X | 0 or 1 or ↓ | No change | No change |
| 1 | 0 | ↑ | 0 | 1 |
| 1 | 1 | ↑ | (toggle) | (toggle) |

## Trending and Filtering Rules

Use **Trending** rules to find the average, minimum, and maximum values for a specific register at a user-specified time interval. The Trend function collects data when the device boots up and accumulates the average, minimum, and maximum values for a register.

Local registers locations are defined include: Register to Track, Filter Register, Average Value Register, Minimum Value Register, Maximum Value Register, and Enable Register. When not using the local register definitions, leave them set to zero.

The filter and tracking is applied according to this flowchart.




The trending rule has an optional filter input, and the filter output is the input to the trending operation. The filter result is used before applying the trending rule when you define the **Filter** parameter. The other trending parameters are listed. The Average, Minimum, and Maximum register fields are optional and can be left set to zero when using the Filter output register.

**Register to Track**

Local register to provide the input to the filter and trending rule.

**Sample Interval**

Defines the time interval of the samples to collect from the register.


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

**Filter**

Defines the specific filter algorithm to apply to the front end of the trending rule.

**Cumulative Average**. Sums the samples and divides them by the number of samples.
Exponential Moving Average. A moving average filter that applies weighting factors that decrease exponentially. The weighting for each older sample decreases exponentially, never reaching zero.
**Lulu Smoother**. Takes the minimum and maximum values of mini sequences of sample points. The mini-sequence length is defined by the filter window parameter.
Median Average. Takes a least three sample points, sorts the sample points numerically, and selects the middle value as the result. A new sample point entered into the filter removes the oldest sample point.
**Recursive Filter**. Created from a percentage of the current sample plus a percentage of past samples. The number of samples is determined by the **Filter Window** parameter.
**Simple Moving Average**. Takes the number of samples defined by the **Filter Window** parameter and averages the samples for the result. When a new sample is taken, it becomes the latest sample point, and the oldest sample point is taken out of the filter averaging.
**Weighted Moving Average**. A moving average filter that applies a weighting factor to the data based on when the data was captured. More recent data has a higher weighted factor.

**Filter Window**

Defines the number of samples for a filter algorithm. For example, if the sample interval is 5 minutes and the Filter Window is 100, this creates a 100-point moving average over a 500-minute time period.

**Filter Register**

The local register that stores the output of the first-stage filter. This is not required to connect the filter front-end to the trending rule processing.

**Average Value Register**

The local register number to store the average value of the trending rule. The average is calculated from the beginning of the power-up or from the **Reset Trending** time. (Hourly, Daily) Defining this local register is optional and can be left at zero.

**Minimum Value Register**

The local register number to store the minimum value of the trending rule. The minimum is captured from the beginning of power-up or from the **Reset Trending** time. (Hourly, Daily) Defining this local register is optional and can be left at zero.

**Maximum Value Register**

The local register number to store the maximum value of the trending rule. The maximum is captured from the beginning of power-up or from the **Reset Trending** time. (Hourly, Daily) Defining this local register is optional and can be left at zero.

**Reset Trend Data**

Use to reset the trend data accumulation. (Hourly, Daily, or not at all)

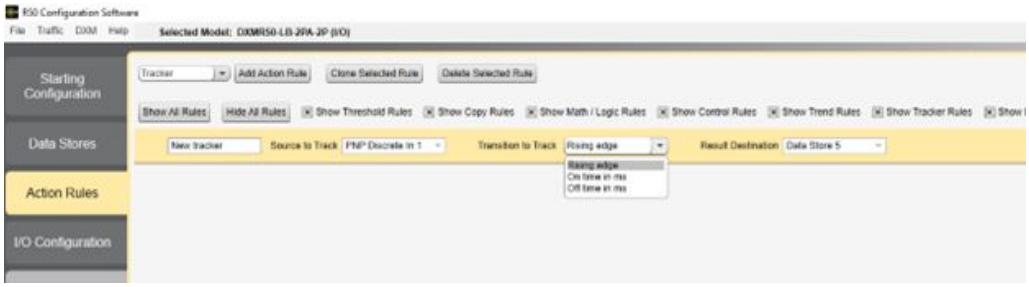
**Enable Register**

The **Enable Register** is an optional register and can be left set to zero. The **Enable Register** defines the address of a local register that can be used to turn on and off the trending function. Set the register value to 1 to turn on trending or to zero to turn off trending. When disabled, the current data is held until the rule is enabled again.

## Tracker Rules

Tracker rules monitor a data store register and store the result of a function in another register.

*Tracker Rules screen*




The possible functions are:

- **Rising Edge**. Counts the number of register transitions from 0 to non-zero value. The speed at which it can count depends on how much work the DXM is performing. Typical tracker rates should be around one to two times per second.
- **On time in ms**. Tracks the time (ms) that the register is in the non-zero state
- **Off time in ms**. Tracks the time (ms) that the register is in the zero state


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

## Decoder Rules

Use the Decoders screen to copy a subset of the bits contained in a data store register location into another register.

*Decoder Rules screen*




**Add Decoder Rule**
Click to create a new decoder rule.

**Name**
Name your rule.

**Copy from Source Register**
The register to copy data from.

**Number of bits**
The total number of bits (minimum of 1, maximum of 32) to copy from the source register.

The **Number of bits** field auto-decrements when either of the **Starting at bit index** fields is increased to ensure that the number of bits copied does not overflow the maximum bit index of 31.

**Starting at bit index**
The specified number of bits will be copied from the source register ascending from the specified index (minimum of 0, maximum of 31). Starting at bit 0 will capture the lowest-order portion of the register's value, while starting at index 31 will copy only the highest-order bit.

**Destination Register**
The register to copy data to.

**Starting at bit index**
The lowest-order bit of the data copied from the source register will occupy this bit index in the destination register.

On the DXMR50-LB, these rules are executed by performing a bit shift, applying a mask, and writing the result to the destination register.

## Delay and Scaler Rules

**Delay** rules allow for time-based logic between source and destination registers.

**Scaler** rules transform the domain of register values between source and destination registers.

# IO Configuration

Go to the IO Configuration tab to define read rules, write rules, and the Modbus settings.

Both the Read Rules and Write Rules screens include a **Show Internal Rules** checkbox. These internal rules load with your starting configuration files and are default rules specific to the discrete/analog inputs and outputs. When the checkbox is not selected, the rules are hidden from view. For normal operation, the internal rules do not need to be changed.

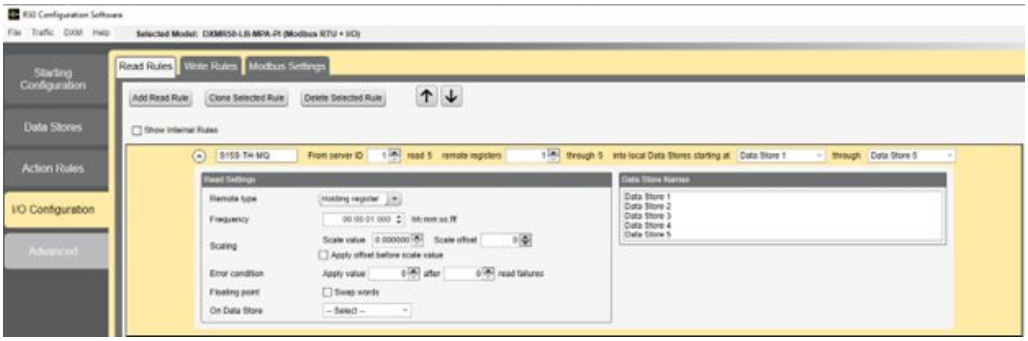
## Create a Read Rule for the DXMR50-LB-MPA Series

Each read rule defines a Modbus server ID and register range to read and then store in the selected data store registers. Use the **I/O Configuration › Read Rules** screen to create a new read rule.

1. Click **Add Read Rule**.
   A new read rule is created.
2. Click the arrow next to the first rule to view the parameters.
   The list of user-defined parameters displays.
3. Type in the rule's name in the **none** field.
4. Select the server ID of the source device.
5. Select the number of registers to read from the source device by modifying the data store register selectors.


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

*Read Rules screen*




## Create a Read Rule for the DXMR50-LB-2M Series

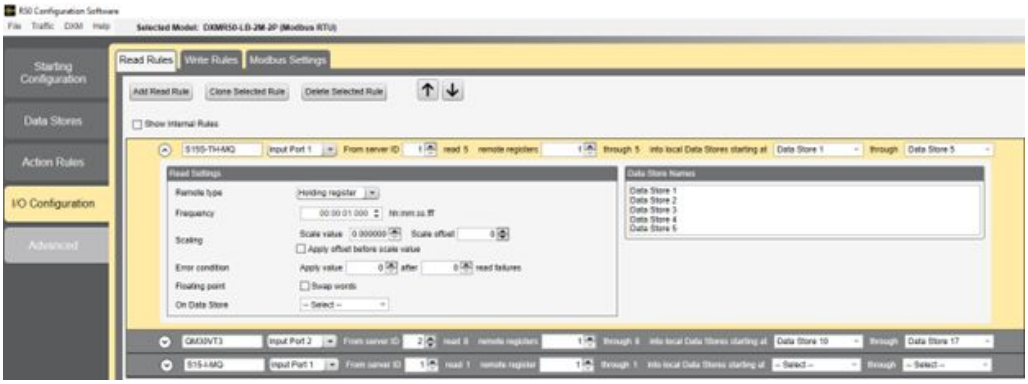
Each read rule defines a Modbus server ID and register range to read and then store in the selected data store registers. Use the **I/O Configuration › Read Rules** screen to create a new read rule.

1. Click **Add Read Rule**.

   A new read rule is created.
2. Click the arrow next to the first rule to view the parameters.

   The list of user-defined parameters displays.
3. Type in the rule's name in the **none** field.
4. Select the port number the Modbus server is connected to on the DXMR50. This can be either **Input Port 1** or **Input Port 2**.
5. Select the server ID of the source device.
6. Select the number of registers to read from the source device by modifying the data store register selectors.

*Read Rules screen*




## Create a Write Rule for the DXMR50-LB-MPA Series

The write rules write data store register data to the defined Modbus server ID and registers. Use the **I/O Configuration › Write Rules** screen to create a new write rule.

1. Click **Add Write Rule**.

   A new write rule is created.
2. Click the arrow next to the first rule to view the parameters.

   The list of user-defined parameters displays.
3. Type in the rule's name in the **none** field.
4. Select the server ID of the target device.
5. Select the number of registers to write to the target device by modifying the data store register selectors.
6. Select the starting register of the target device.


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

*Write Rules screen*




The ending register automatically fills in based on the starting register and the number of registers selected.

## Create a Write Rule for the DXMR50-LB-2M Series

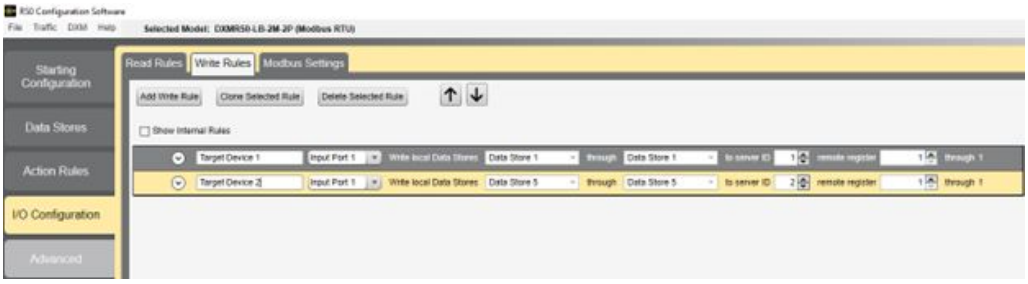
The write rules write data store register data to the defined Modbus server ID and registers. Use the **I/O Configuration › Write Rules** screen to create a new write rule.

1. Click **Add Write Rule**.

   A new write rule is created.
2. Click the arrow next to the first rule to view the parameters.

   The list of user-defined parameters displays.
3. Type in the rule's name in the **none** field.
4. Select the port number the Modbus server is connected to on the DXMR50. This can be either **Input Port 1** or **Input Port 2**.
5. Select the server ID of the target device.
6. Select the number of registers to write to the target device by modifying the data store register selectors.
7. Select the starting register of the target device.

*Write Rules screen*




The ending register automatically fills in based on the starting register and the number of registers selected.

## Modbus Settings

Use the **I/O Configuration › Modbus Settings** screen to view and configure the DXMR50 server settings. Select the **Get** and **Set** buttons to view and adjust these settings.

**Baud Rate**

Defined for both the Modbus client and server

Settings include: 19200 (default), 1200, 2400, 9600, 38400, 57600, and 115200

**Parity**

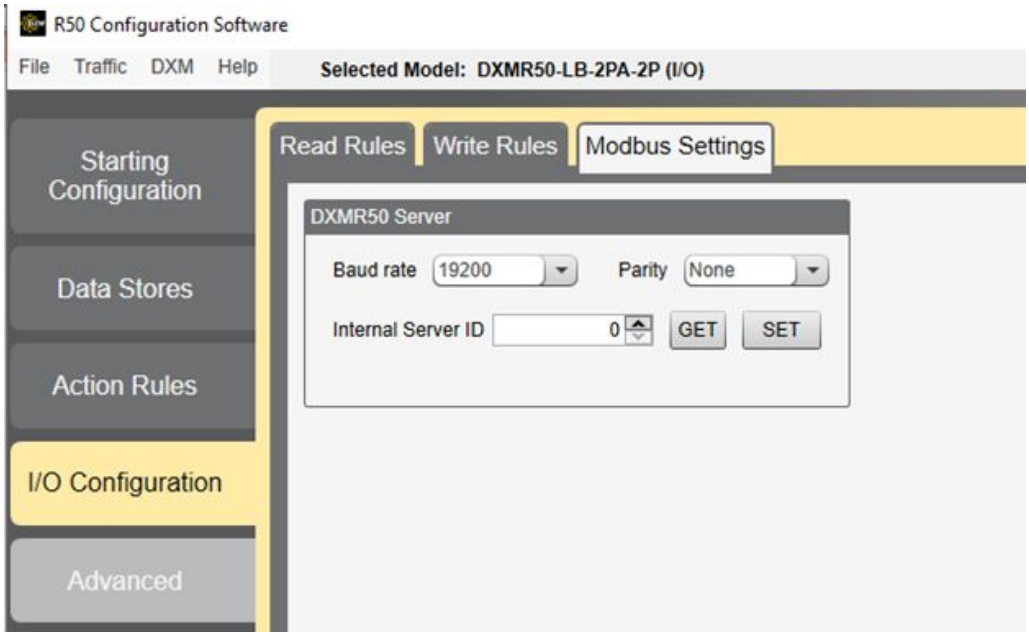
Defined for both the Modbus client and server

Settings include: None (default), odd, even, space, and mark

**Internal Server ID**

The internal server ID is the Modbus ID that an external Modbus client accesses to read/write to the local registers on the DXMR50


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

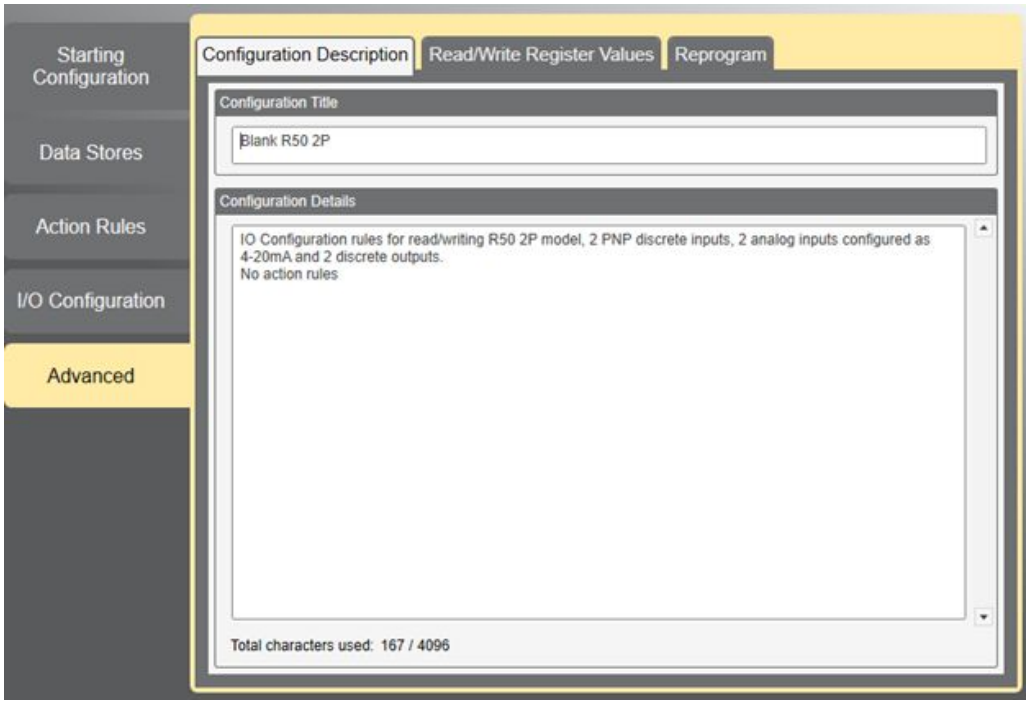



# Advanced Tab

Access the Configuration Description, Read/Writer Register Values, and Reprogram screens from the Advanced tab.

## Configuration Description

Use the **Configuration Description** screen to set and save the **Configuration Title** and **Configuration Details**.

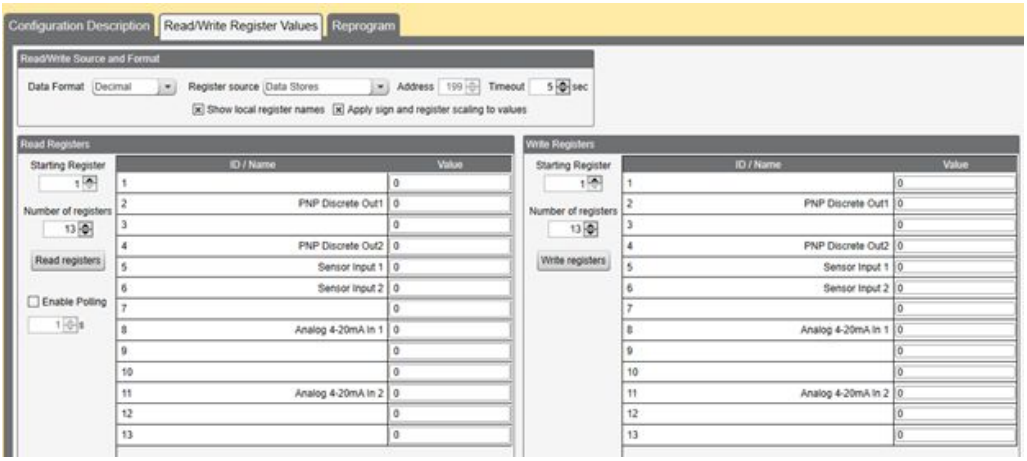




© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

## Read/Write Register Values

The Register View screen is a Modbus utility to help debug configurations and view register data for devices connected to the DXMR50.




Follow these steps to select the registers to view:

1. Connect to the DXMR50-LB using the BWA-UCT-900 adaptor.
2. From the **Register source** drop-down list, select between Remote Device, Data Stores, or DXM inputs/outputs.
3. Enter a Modbus ID when accessing remote devices.
4. Select the **Starting Register** to display.
5. Select the number of registers to display.

To read the contents of a specific register or range of registers:

1. Select the starting register.
2. Select the number of registers to read from.
3. Click **Read Registers** to read the data once. The register values display in the **Read Registers** section.
4. Select **Enable Polling**.
5. Specify a polling rate.
6. Click **Read Registers** to create a constant polling loop.

To write values to a specific register or range of registers:

1. Select the starting register.
2. Select the number of registers to write to.
3. Enter the value to write to these registers.
4. Click **Write Registers** to send these defined values to the selected registers.

## Reprogram

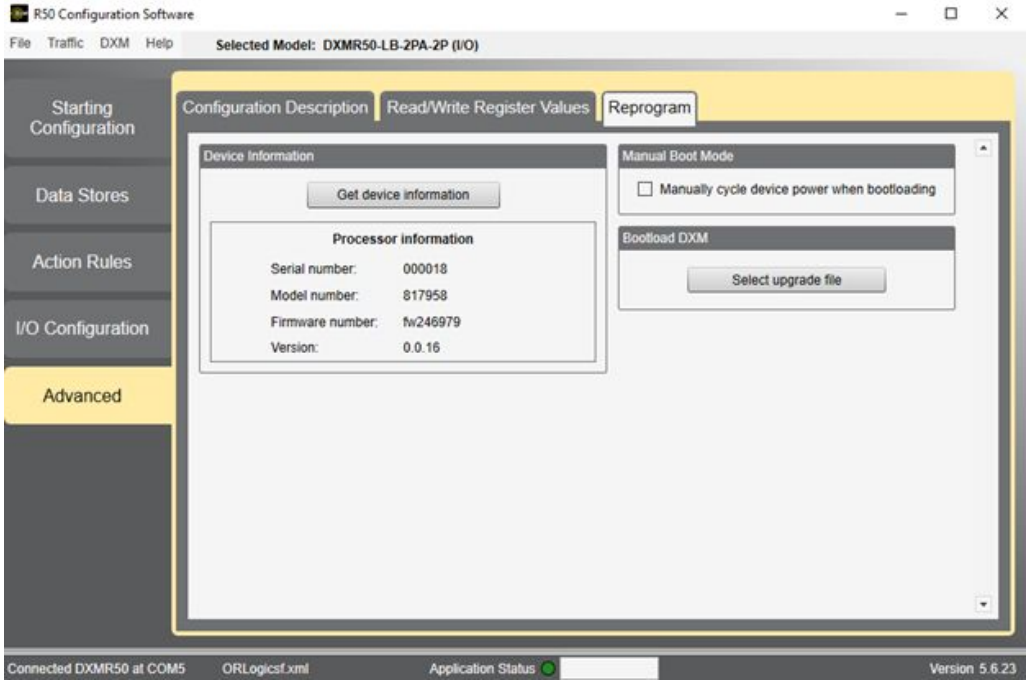
Use the Reprogram screen to update the DXM's firmware.

Click **Get Device Information** to have the DXM read and display its serial number, model number, version, and firmware numbers.

To reprogram the DXM, follow these instructions:

1. Connect the DXM to your PC using the BWA-UCT-900 adaptor cordset.
2. Go to the **Advanced › Reprogram** screen.
3. Click **Select upgrade file**.
4. Browse to the firmware HEX file location and select the file.


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com




# Examples

## Configure the DXMR50-LB Logic Block

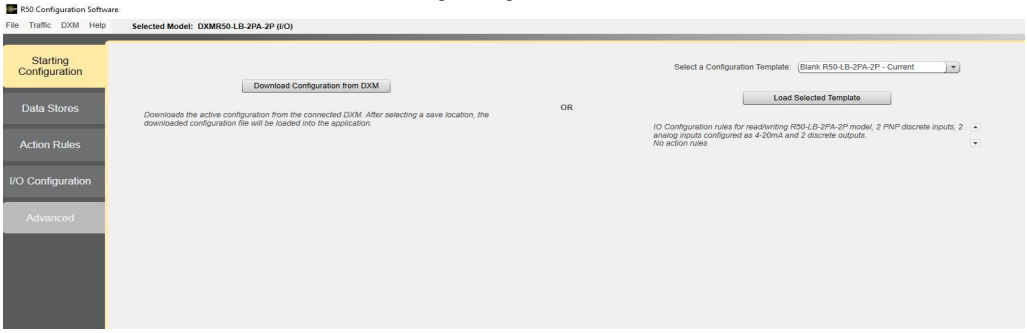
This section will walk you through the method of setting up the DXMR50 Configuration Software and communicating with a connected DXMR50-LB device.

After your DXMR50-LB is connected to your computer, select **Auto-Detect** so that the software automatically detects the correct model and loads the appropriate screens.

You may also manually select which model of DXM you are configuring if you intend to create a configuration file without connecting a device. This ensures that the interface and the configuration file use the correct features.

1. Download and install the DXMR50 Configuration Software from www.bannerengineering.com/something.
2. Use the BWA-UCT-900 adapter cordset to connect the DXMR50-LB to your PC.
3. Launch the configuration software.
4. Identify your DXMR50-LB model.
    - If your DXMR50-LB is connected to your PC, click **Auto-Detect**.
    - If the software did not detect the correct model or if you want to create a configuration file without connecting to your DXMR50-LB, manually select which model of DXMR50-LB you are configuring.

*Starting Configuration screen*




Click **Download the Configuration from DXM** to view the current configuration of the connected DXMR50-LB. Note that the **Download Configuration from DXM** option is available only if an DXMR50-LB device is connected.


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

The configuration software can also be used in an offline mode (no device connected) by manually selecting a device model from the starting screen, loading a template to build, and saving a configuration file locally.

Click **Load Selected Template** to open the **Data Stores** screen with the Modbus registers pre-labeled. Use these registers to create custom logic in the **Action Rules** screen. You can select a configuration template from the drop-down list and review the description to ensure you are starting with the baseline configuration that's most applicable before clicking the **Load Selected Template** button.

## Save and Upload the Configuration File

After making any changes to the configuration, you must save the configuration files to your computer, then upload it to the device.

Changes to the XML file are not automatically saved. Save your configuration file before exiting the software and before sending the XML file to the device to avoid losing data. If you select **DXM › Send XML Configuration to DXM** before saving the configuration file, the software will prompt you to choose between saving the file or continuing without saving the file.

1. Save the XML configuration file to your hard drive by going to the **File › Save As** menu.
2. Go to the **DXM › Send XML Configuration to DXM** menu.

*Status indicator bar*




   - If the Application Status indicator is red, close and restart the DXMR50-LB Logic Block Configuration Software, unplug and re-plug in the cable and reconnect the DXM to the software.
   - If the Application Status indicator is green, the file upload is complete.
   - If the Application Status indicator is gray and the green status bar is in motion, the file transfer is in progress.

After the file transfer is complete, the device reboots and begins running the new configuration.

## Configure a DXMR50-LB-2PA-2P for AND Logic

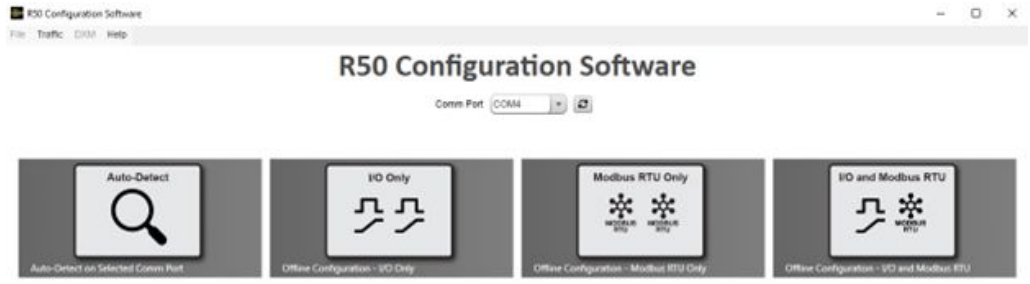
*Configure a DXMR50-LB-2PA-2P for AND Logic*



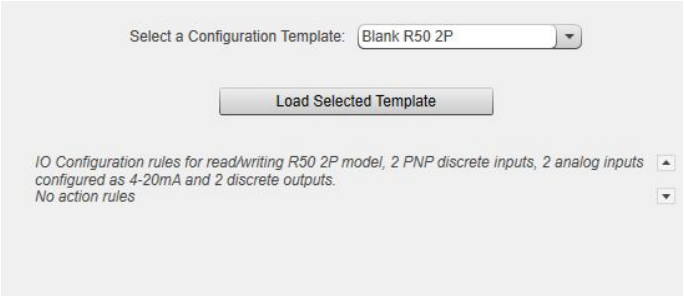
For this configuration example, we have two discrete sensors connected to the input ports on the DXMR50-LB-2PA-2P and a light connected to the output port (pin 2 on the M12 connector).

1. Using the BWA-UCT-900 Serial RS-485 to USB adaptor cordset, connect the DXMR50-LB-2PA-2P to your PC and launch the DXMR50-LB Logic Block Configuration Software.
2. In the configuration software, select **Auto-Detect**.

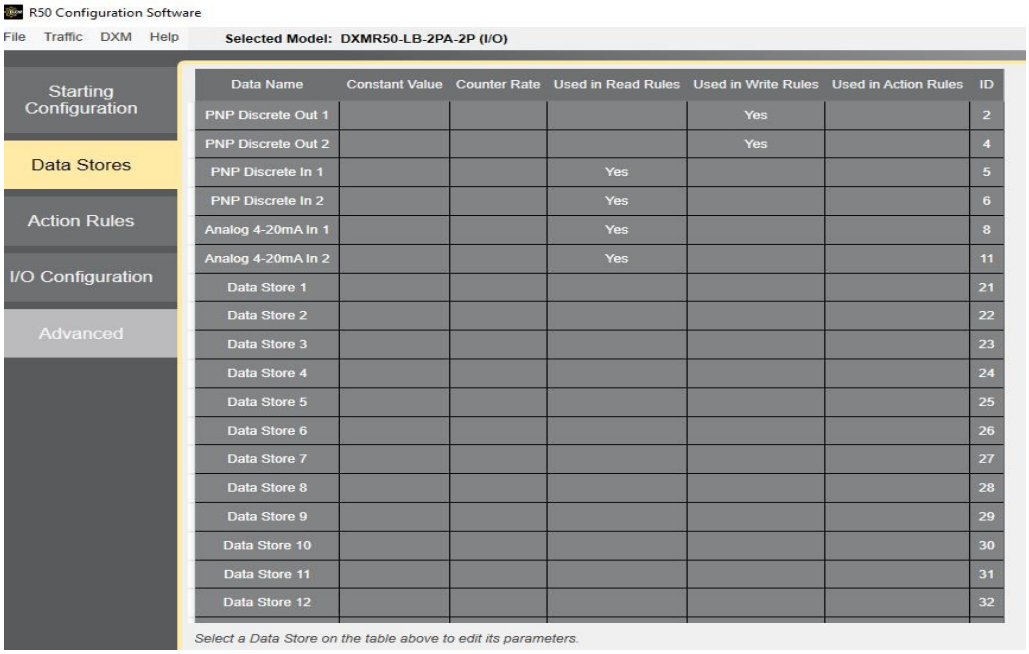

© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com




3. In the next window, select **Load Selected Template**.

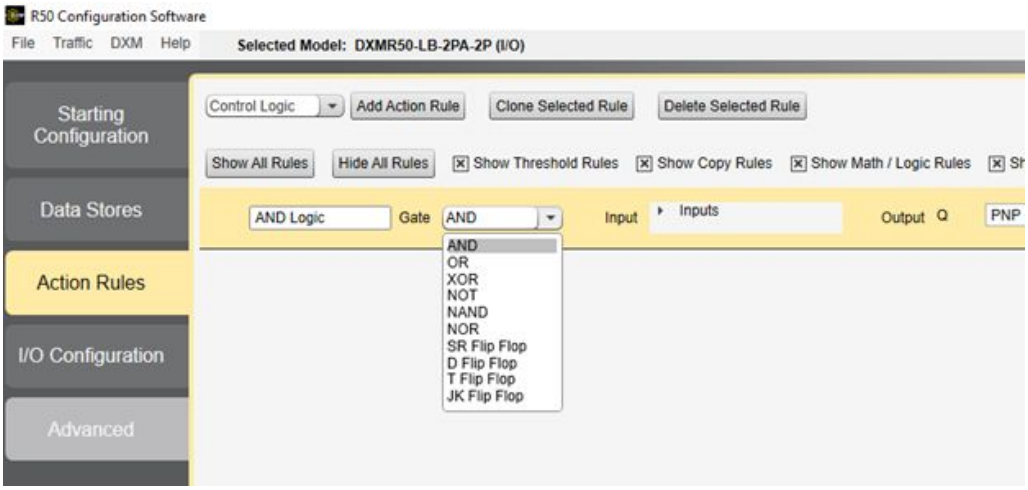



The **Data Stores** screen opens. In this screen, we can rename and edit each of these data stores, but in this example, we will leave the values as is.

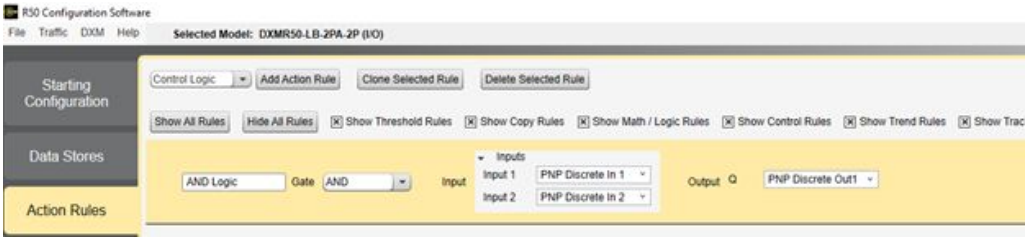



4. On the **Action Rules** screen, select **Control Logic** from the drop-down menu, then click **Add Action Rule**.
   The new action rule populates.
5. Rename this action rule **AND Logic** and select **AND** from the **Gate** drop-down list.


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com



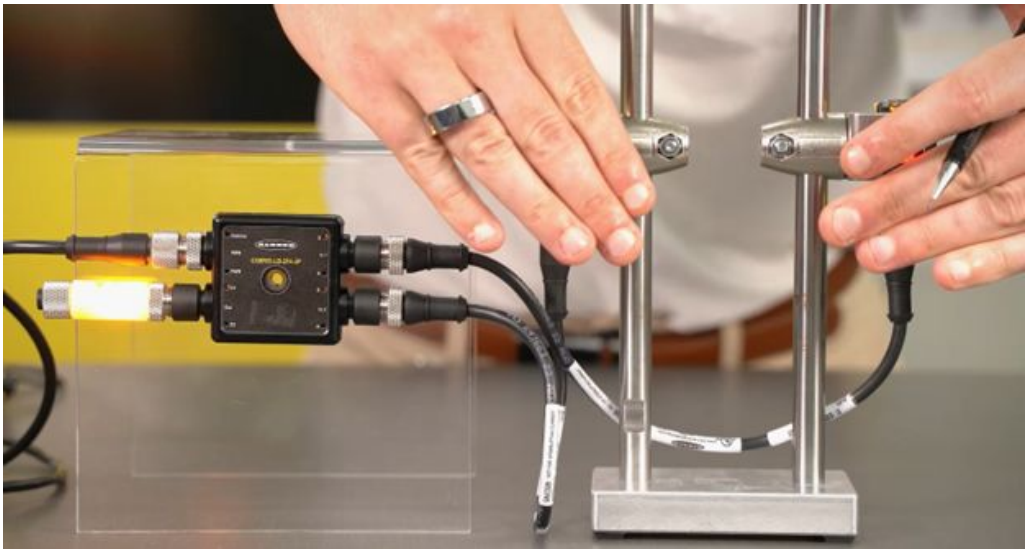

6. From the **Inputs** drop-down list, select **PNP Discrete In 1** for input 1 and **PNP Discrete In 2** for input 2.
7. From the **Output** drop-down list, select **PNP Discrete Out 1**.




8. On the menu bar, go to **File › Save As** and give your XML file a name. Save this file to your PC.

   Note that changes to the XML file are not automatically saved. Save your configuration file before exiting the software and before sending the XML file to the device to avoid losing data. If you select **DXM › Send XML Configuration to DXM** before saving the configuration file, the software will prompt you to choose between saving the file or continuing without saving the file.
9. Go to **DXM › Send Configuration to DXM**.

   After the file transfer is complete, the device reboots and begins running the new configuration.
10. Test the logic. With the new configuration, when the two sensors are activated simultaneously, the light turns on.




© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

Chapter Contents

DXMR50-LB Configuration Software Release Notes ..... 25
Banner Engineering Corp. Software Copyright Notice ..... 25

# Chapter 3 Product Support

## DXMR50-LB Configuration Software Release Notes

The following updates are included in the DXMR50-LB Configuration Software (content ID ).

| Date | Version | Release Notes |
|---|---|---|
| 5 March 2026 | 5.7.0 | Initial release of the DXMR50-LB Configuration Software |
| | | |

## Banner Engineering Corp. Software Copyright Notice

© Banner Engineering Corp., All Rights Reserved.

https://www.bannerengineering.com/us/en/company/terms-and-conditions.html

**Disclaimer of Warranties**. This software is provided "AS-IS." To the maximum extent permitted by applicable law, Banner, it affiliates, and its channel partners disclaim all warranties, expressed or implied, including any warranty that the software is fit for a particular purpose, title, merchantability, data loss, non-interference with or non-infringement of any intellectual property rights, or the accuracy, reliability, quality or content in or linked to the services. Banner and its affiliates and channel partners do not warrant that the services are secure, free from bugs, viruses, interruption, errors, theft or destruction. If the exclusions for implied warranties do not apply to you, any implied warranties are limited to 60 days from the date of first use of this software.

**Limitation of Liability and Indemnity**. Banner, its affiliates and channel partners are not liable for indirect, special, incidental, punitive or consequential damages, damages relating to corruption, security, loss or theft of data, viruses, spyware, loss of business, revenue, profits, or investment, or use of software or hardware that does not meet Banner minimum systems requirements. The above limitations apply even if Banner and its affiliates and channel partners have been advised of the possibility of such damages. This Agreement sets forth the entire liability of Banner, its affiliates and your exclusive remedy with respect to the software use.


© Banner Engineering Corp. All rights reserved.
www.bannerengineering.com

LinkedIn

X

Facebook

Instagram




© 2026. All rights reserved.