Setting Multiple Thresholds Using **Action** Rules or ScriptBasic

Setting Multiple Thresholds

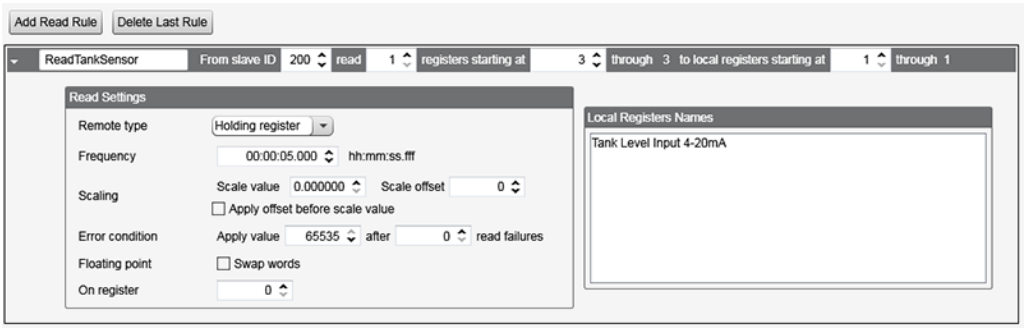
The DXM Controller uses Action Rules and the DXM Configuration Tool to do simple operations with local registers. ScriptBasic is a programming language that can handle the simple operations as well as the complex operations.

This example application creates multiple thresholds on a single analog input from an external 4-20 mA sensor to indicate tank levels. The first method uses Action Rules and the second method uses ScriptBasic.

Use **Action** Rules to Set **Multiple** Thresholds

This procedure assumes you have installed the DXM Configuration Tool on a Windows-based computer.

1. Launch the DXM Configuration Tool.
2. Define the Local Register.
   a) Go to the Local Registers > Local Register **Configuration** screen.
   b) Click on the arrow next to input 1 to display its parameters. Name this input.
   c) Set the Units to mA and set LCD Permissions to Read. Setting the Units to mA isn't necessary, but does make the field easier to understand. Setting LCD Permissions to Read displays the register value on the DXM's LCD.
3. Define the Read Rule.



   a) Go to the Register Mapping > Read Rules screen and click Add Read Rule to define the read rule that copies a DXM Controller register value into local register 1.
   b) Enter a name for the Read Rule. For our example, we have named the rule: ReadTankSensor.
   c) Set the parameters to read one register from Slave ID 200 (DXM Controller), starting at register 3, and writing the value to local register 1 every 5 seconds.
4. Define the Local Registers and Action Rules that check the thresholds.
   a) Name and define Local Register 2 as the 8 mA Threshold. Set the Units to on/off and the LCD Permissions to Read.
   b) Name and define Local Register 3 as the 12 mA Threshold. Set the Units to on/off and the LCD Permissions to Read.
   c) Name and define Local Register 4 as the 16 mA Threshold. Set the Units to on/off and the LCD Permissions to Read.
   d) Define an Action Rule to set Local Register 2 to 1 when the tank level sensor reading is above 8 mA.
   e) Define an Action Rule to set Local Register 2 to 1 when the tank level sensor reading is above 12 mA.
   f) Define an Action Rule to set Local Register 3 to 1 when the tank level sensor reading is above 16 mA.

To view the results of the Action Rules, go to the DXM Controller. On the menu system, select Registers and click Enter. Any registers configured with LCD Permissions set to Read displays on the screen.

Save and Upload the **Configuration** File

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

1. Save the XML configuration file to your hard drive by going to the File > Save As menu.
2. Upload the configuration file to your DXM Controller.
   a) Verify your DXM Controller is connected to the Windows-based PC running the DXM Configuration Tool.
   b) Go to the Device > Send XML **Configuration** to DXM menu.

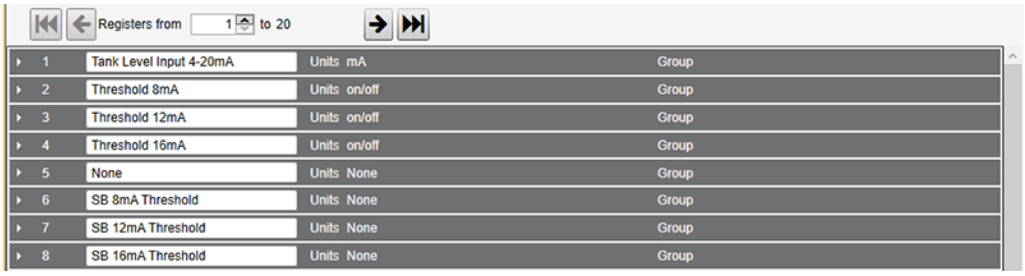
   The DXM Controller reboots and begins running the new configuration.

Use ScriptBasic to Set **Multiple** Thresholds

ScriptBasic is the programming language integrated into the DXM Controller. ScriptBasic has access to all local register data and can send Modbus read or write commands just like the Read or Write Rules.

Where ScriptBasic is useful is when timing needs to be controlled or an application is becoming more complex than a few action rules. ScriptBasic comments document what the program is trying to accomplish, so someone else can look at the program and understand what the program is trying to accomplish.

To write the ScriptBasic program, begin with the existing configuration and add names for registers 6, 7, and 8 that will be controlled by ScriptBasic. The rest will be done in the ScriptBasic program.



ScriptBasic program files must have the suffix .SB and can be created from any text editor. Text editors that color key words are easier to read.

An example ScriptBasic program is shown. Key words are in bold, and comments are in italics. In the first half of the program, constants are declared and variables are assigned. This isn't necessary, but it is much easier to read the program with names that explain the values.

```
'ScriptBasic program creating multiple thresholds
'
CONST LocalRegSID        = 199
CONST IO_BoardSID        = 200
CONST DisplaySID        = 201
'
CONST HoldingReg        = 0
'
CONST  AnalogIn_reg        = 3
CONST  Threshold8mA_reg    = 6
CONST  Threshold12mA_reg    = 7
CONST  Threshold16mA_reg    = 8
'
CONST LED1_reg            = 1102
CONST LED2_reg            = 1103
CONST LED3_reg            = 1104
'
RdData               = 0
WrErr               = 0
'
WHILE(1)
    'Read the I/O board universal input 1, Modbus register 3
    RdData    = GETREG(AnalogIn_reg, IO_BoardSID, HoldingReg)
    '
    Threshold8mA        = 0
    Threshold12mA        = 0
    Threshold16mA        = 0
    ' Analog input value is in uA, so 8000 = 8mA
    '
    IF RdData >= 8000 AND RdData < 12000 THEN
        Threshold8mA     = 1
    ELSEIF RdData >= 12000 AND RdData < 16000 THEN
        Threshold12mA    = 1
    ELSEIF RdData >= 16000 THEN
        Threshold16mA    = 1
    END IF
    '
    ' Output to the console to show the values...
    PRINT " Analog In Value: ",RdData, "\t Threshold 8mA: ",Threshold8mA
    PRINT "\t Threshold 12mA: ",Threshold12mA, "\t Threshold 16mA: ",Threshold16mA, "\n\r"
    '
    ' Write the ScriptBasic variables back to the local registers.
    WrErr    = SETREG(Threshold8mA_reg, Threshold8mA, LocalRegSID, HoldingReg)
    WrErr    = SETREG(Threshold12mA_reg, Threshold12mA, LocalRegSID, HoldingReg)
    WrErr    = SETREG(Threshold16mA_reg, Threshold16mA, LocalRegSID, HoldingReg)
    ' Write the ScriptBasic variables to the Display LEDs.
    WrErr    = SETREG(LED1_reg, Threshold8mA, DisplaySID, HoldingReg)
    WrErr    = SETREG(LED2_reg, Threshold12mA, DisplaySID, HoldingReg)
    WrErr    = SETREG(LED3_reg, Threshold16mA, DisplaySID, HoldingReg)
    '
    SLEEP(5)
```

```
WEND

END
```

The main program starts with the WHILE(1) command and ends with the WEND key word. This creates a loop between these two statements that will loop forever. For more information about ScriptBasic commands see the DXM ScriptBasic Quick Start Guide.

The first command is GETREG. As the name implies it is getting a Modbus register from the I/O board, the analog input and putting it into variable *RdData.*

```
RdData = GETREG(AnalogIn_reg, IO_BoardSID, HoldingReg)
```

The IF THEN ELSE structure is checking the value of the variable *RdData*. If the value is greater or equal to 8000 and less than 12000 then the variable Thresold8mA is set to 1. If the value is greater or equal to 12000 and less than 16000 then the variable Thresold12mA is set to 1. If the value is greater or equal to 16000 then the variable Thresold16mA is set to 1. This is similar to the three Action rules created earlier, but not quite.

To make the equivalent Action rules function to this ScriptBasic program would require us to double the number of action rules needed (6) and math Action rules (3) to AND the Action rule results. To exactly match the one Action rule earlier the equivalent ScriptBasic statement would be:

```
IF RdData >= 8000 THEN
Threshold8mA = 1
```

This shows how quickly the Action rules can get complicated, but the ScriptBasic implementation remains simple to understand.

The next command is PRINT, which sends the statement out to the console. It will print out the analog value read from the I/O board and the results of each variable. This is the most common way to debug a program, print out program variables and view the live output.

The next series of six SETREG functions writes the variables to the three local registers and writes the threshold variables to the LEDs on the DXM Controller display. Using the LEDs on the DXM display helps to debug ScriptBasic programs.

The SLEEP command pauses the program for 5 seconds before continuing. Then the program reads the WEND command word and loops back the WHILE command and the program runs again.

Save the program (*.sb) then use the DXM Configuration Tool to load the program into the DXM Controller. To load the program:

1. Go to the **Settings** > **Scripting** screen.
2. Click on the Upload Script button. A pop-up box appears to load the program.
3. After the program finishes loading, click on the file name and then click on Add selected to startup. The program name appears in the box Startup Scripts.

The configuration file has now been changed to run the ScriptBasic program at startup time. Save the configuration file and then load the new configuration file to the DXM Controller by clicking on Upload **Configuration** to Device under the Device menu.

Running the ScriptBasic Program

After the DXM Controller powers up or reboots, the DXM's processor loads the XML configuration file. The XML configuration file specifies the ScriptBasic program to run.

To view the ScriptBasic program operation, use the console output and a terminal program, such as PuTTY[1]. Watch the same COMM port as the DXM Configuration Tool software uses.

The ScriptBasic program sends output to the console with every PRINT statement. After the DXM boots, the console output should print out "ScriptBasic Script Started". If this messages does not appear, the XML file was not properly configured or the file was not loaded on the DXM Controller.

[1] PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers. Download PuTTY here: http://www.putty.org/