

## **Sample Ladder Overview**

The sample ladder performs several important tasks for the MVI56-AFC operation. For most of the applications, the user should not have to perform any modifications to the sample ladder logic.

The sample ladder logic performs the following tasks:

- Write the Wallclock from the processor to the MVI56-AFC
- Enable all 16 meters
- Disable all 16 meters
- Displays the current enable/disable status of all 16 meters
- Transfer the process variables from the processor to the MVI56-AFC for all 16 meters.
- Transfer the calculation results from the MVI56-AFC to the processor for all 16 meters.
- Transfer the molar analysis data (gas only) for each meter
- Reset all 4 Resettable Accumulators for each meter
- Write a Daily or Hourly Archive
- Read each meter profile ( meter type and product group)
- Transfer up to 2000 words of data between the processor and the Primary or Virtual Modbus Slaves.
- Set the processor date and time information using a controller tag as source
- Read the meter alarms
- Read the site status

## Using the Sample Ladder

This section shows how the user can extract important information from the sample ladder without explaining the details of how the sample ladder actually works. The user will realize that for most applications it will be enough to refer only to the controller tags in order to perform the tasks.

### - Enable/Disable Status

Each meter run will only perform flow calculation while it is enabled. However, the user cannot change a meter type, product group or units while the meter is enabled. In order to accomplish this, the user has to disable the meter, change meter type, product or units and then enable the meter again. The meters can be enabled or disabled from ladder logic or AFC Manager.

The ladder logic constantly reads each meter enable/disable status from the MVI56-AFC. The user should refer to *AFC56.EnableStatus* data structure for the each meter status. Each variable should be interpreted as:

*AFC56.EnableStatus.Meterx* = 0 => Meter x is Disabled

*AFC56.EnableStatus.Meterx* = 1 => Meter x is Enabled

AFC56.EnableStatus	{...}
AFC56.EnableStatus.Meter1	1
AFC56.EnableStatus.Meter2	1
AFC56.EnableStatus.Meter3	0
AFC56.EnableStatus.Meter4	0
AFC56.EnableStatus.Meter5	0
AFC56.EnableStatus.Meter6	0
AFC56.EnableStatus.Meter7	0
AFC56.EnableStatus.Meter8	0
AFC56.EnableStatus.Meter9	0
AFC56.EnableStatus.Meter10	0
AFC56.EnableStatus.Meter11	0
AFC56.EnableStatus.Meter12	0
AFC56.EnableStatus.Meter13	0
AFC56.EnableStatus.Meter14	0
AFC56.EnableStatus.Meter15	0
AFC56.EnableStatus.Meter16	0

In the previous example, Meter 1 and Meter 2 are enabled. All other meters are disabled.

## - Disable Meter

Each meter can be disabled through ladder logic. The user should refer to the *AFC56.DisableMeter* data structure. The user should toggle each *AFC56.DisableMeter.Meterx* controller tag in order to command the meter to be enabled.

The ladder logic will continuously analyze each meter enable status. Once the logic realizes that a specific meter is disabled (*AFC56.DisableStatus.Meterx = 1*) the command bit (*AFC56.DisableMeter.Meterx*) will be unlatched.

<input type="checkbox"/> AFC56.DisableMeter	{...}
<input type="checkbox"/> AFC56.DisableMeter.Meter1	0
<input type="checkbox"/> AFC56.DisableMeter.Meter2	0
<input type="checkbox"/> AFC56.DisableMeter.Meter3	0
<input type="checkbox"/> AFC56.DisableMeter.Meter4	0
<input type="checkbox"/> AFC56.DisableMeter.Meter5	0
<input type="checkbox"/> AFC56.DisableMeter.Meter6	0
<input type="checkbox"/> AFC56.DisableMeter.Meter7	0
<input type="checkbox"/> AFC56.DisableMeter.Meter8	0
<input type="checkbox"/> AFC56.DisableMeter.Meter9	0
<input type="checkbox"/> AFC56.DisableMeter.Meter10	0
<input type="checkbox"/> AFC56.DisableMeter.Meter11	0
<input type="checkbox"/> AFC56.DisableMeter.Meter12	0
<input type="checkbox"/> AFC56.DisableMeter.Meter13	0
<input type="checkbox"/> AFC56.DisableMeter.Meter14	0
<input type="checkbox"/> AFC56.DisableMeter.Meter15	0
<input type="checkbox"/> AFC56.DisableMeter.Meter16	0

Obs: DO NOT create a rung in ladder logic to constantly disable the meter. The command bit should be toggled only once in order to disable the meter.

## - Enable Meter

Each meter can be enabled through ladder logic. The user should refer to the *AFC56.EnableMeter* data structure. The user should toggle each *AFC56.EnableMeter.Meterx* controller tag in order to command the meter to be enabled.

The ladder logic will continuously analyze each meter enable status. Once the logic realizes that a specific meter is enabled (*AFC56.EnableStatus.Meterx = 1*) the command bit (*AFC56.EnableMeter.Meterx*) will be unlatched.

AFC56.EnableMeter	{...}
AFC56.EnableMeter.Meter1	0
AFC56.EnableMeter.Meter2	0
AFC56.EnableMeter.Meter3	0
AFC56.EnableMeter.Meter4	0
AFC56.EnableMeter.Meter5	0
AFC56.EnableMeter.Meter6	0
AFC56.EnableMeter.Meter7	0
AFC56.EnableMeter.Meter8	0
AFC56.EnableMeter.Meter9	0
AFC56.EnableMeter.Meter10	0
AFC56.EnableMeter.Meter11	0
AFC56.EnableMeter.Meter12	0
AFC56.EnableMeter.Meter13	0
AFC56.EnableMeter.Meter14	0
AFC56.EnableMeter.Meter15	0
AFC56.EnableMeter.Meter16	0

Obs: DO NOT create a rung in ladder logic to constantly enable the meter. The command bit should be toggled only once in order to enable the meter.

## - Wallclock

Once the module powers up it will not perform flow calculation until it receives a valid wallclock information from the Controllogix processor. The ladder logic uses the processor internal clock as the source of the wallclock information.

Initially the user should configure the processor time and date information:

- 1) Right-Click on Controller MVI56-AFC folder
- 2) Click on Properties
- 3) Select the Date/Time tab
- 4) Enter a valid date and time information.

Once the *AFC56.Flags.AFC\_Set\_Clock* bit is toggled, the logic will move the date and time information from the processor to the MVI56-AFC module.

[-] AFC56.Flags	{...}
- AFC56.Flags.AFC_Stopped	0
- AFC56.Flags.AFC_Set_Clock	0
- AFC56.Flags.AFC_OutputError	0
[+] AFC56.Flags.AFC_InputSequenceNum	0
- AFC56.Flags.plc_set_clock	0

The *AFC56.Flags.AFC\_Set\_Clock* bit is latched in the power up routine, in order to guarantee that the module will be up and running after power up.



Once the ladder logic receives the input block back from the module it unlatches the *AFC56.Flags.AFC\_Set\_Clock* bit.

The user may want to periodically synchronize the processor and the module's wallclock, specially when the date and time information is received from a remote station. In this case, further ladder logic is required from the user to periodically toggle the *AFC56.Flags.AFC\_Set\_Clock* bit.

## Meter Profile

The ladder logic constantly reads each meter profile. The meter profile informs each meter type (pulse or orifice) and each product group (gas or liquid). Actually this information is not used by most users, but it is very important to the ladder logic in order to perform other important tasks (write process variables and read calculation results).

The *AFC56.Meters[x].Profile* controller tag is used to store the profile information for each meter run:

*AFC56.Meters[0].Profile* - Meter 1 Profile  
*AFC56.Meters[1].Profile* - Meter 2 Profile  
 ...  
*AFC56.Meters[15].Profile* - Meter 16 Profile

The controller tags are interpreted as follows:

Controller Tag	Value	Description
<i>AFC56.Meters[x].Profile.MeterType</i>	0	Meter x is an Orifice Meter
<i>AFC56.Meters[x].Profile.MeterType</i>	1	Meter x is a Pulse Meter
<i>AFC56.Meters[x].Profile.ProductGroup</i>	0	Meter x uses a Gas product
<i>AFC56.Meters[x].Profile.ProductGroup</i>	1	Meter x uses a Liquid product

The example below shows a situation where Meter 0 is configured as a pulse meter and uses a gas product.

[-] AFC56.Meters[0].Profile	{ ... }
[-] AFC56.Meters[0].Profile.MeterType	1
[-] AFC56.Meters[0].Profile.ProductGroup	0

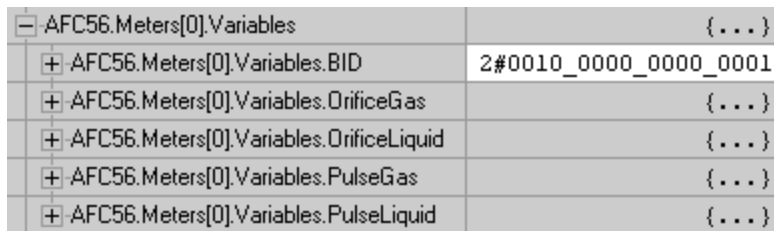
## - Meter Process Variables

In order to perform flow calculation the module must receive the meter process variables from the processor.

The process variables will depend on the meter type and product group. So the profile information (discussed before) is used to decide from which controller tag the variables will be copied from. The following options are used:

Meter Type	Product Group	Use this Controller Tag
Orifice	Gas	<i>AFC56.Meters[x].Variables.OrificeGas</i>
Orifice	Liquid	<i>AFC56.Meters[x].Variables.OrificeLiquid</i>
Pulse	Gas	<i>AFC56.Meters[x].Variables.PulseGas</i>
Pulse	Liquid	<i>AFC56.Meters[x].Variables.PulseLiquid</i>

Follows below the controller tag screenshot from RSLogix5000 :



[- AFC56.Meters[0].Variables	{...}
[+ AFC56.Meters[0].Variables.BID	2#0010_0000_0000_0001
[+ AFC56.Meters[0].Variables.OrificeGas	{...}
[+ AFC56.Meters[0].Variables.OrificeLiquid	{...}
[+ AFC56.Meters[0].Variables.PulseGas	{...}
[+ AFC56.Meters[0].Variables.PulseLiquid	{...}

So this means that all the user has to do is to identify where the variables will be copied from (based on the meter type and product group) and the ladder logic will automatically select the correct controller tags.

Obs: In order to configure each meter type and product group the user should refer to the AFC manager software tool.

For each possible combination, the following variables are used:

**1) Meter Type = Orifice & Product Group = Gas**

Process Input	Controller Tag	Data Type
Temperature	AFC56.Meters[x].Variables.OrificeGas.Temperature	REAL
Pressure	AFC56.Meters[x].Variables.OrificeGas.Pressure	REAL
Differential Pressure	AFC56.Meters[x].Variables.OrificeGas.DifferentialPressure	REAL

Where x can assume values between 0 and 15.

Follows below an example for Meter 1:

- AFC56.Meters[0].Variables.OrificeGas	{...}
+ AFC56.Meters[0].Variables.OrificeGas.Reserved1	0
- AFC56.Meters[0].Variables.OrificeGas.Temperature	35.98
- AFC56.Meters[0].Variables.OrificeGas.Pressure	53.23
- AFC56.Meters[0].Variables.OrificeGas.Differential_Pressure	22.58
+ AFC56.Meters[0].Variables.OrificeGas.Reserved2	0
+ AFC56.Meters[0].Variables.OrificeGas.Reserved3	0
+ AFC56.Meters[0].Variables.OrificeGas.Reserved4	0
+ AFC56.Meters[0].Variables.OrificeGas.Reserved5	0

**2) Meter Type = Orifice & Product Group = Liquid**

Process Input	Controller Tag	Data Type
Water %	AFC56.Meters[x].Variables.OrificeLiquid.Water_Percent	INT
Temperature	AFC56.Meters[x].Variables.OrificeLiquid.Temperature	REAL
Pressure	AFC56.Meters[x].Variables.OrificeLiquid.Pressure	REAL
Diff Pressure	AFC56.Meters[x].Variables.OrificeLiquid.DifferentialPressure	REAL
Density	AFC56.Meters[x].Variables.OrificeLiquid.Density	REAL



**Corporate Office:**  
 1675 Chester Avenue  
 Fourth Floor  
 Bakersfield, CA 93301

661/716-5100

Fax: 661/716-5101

e-mail: prosoft@prosoft-technology.com

[-] AFC56.Meters[0].Variables.OrificeLiquid	{...}
[+] AFC56.Meters[0].Variables.OrificeLiquid.Water_Percent	12
- AFC56.Meters[0].Variables.OrificeLiquid.Temperature	12.1
- AFC56.Meters[0].Variables.OrificeLiquid.Pressure	42.877998
- AFC56.Meters[0].Variables.OrificeLiquid.Differential_Press...	13.22
- AFC56.Meters[0].Variables.OrificeLiquid.Density	32.099998
[+] AFC56.Meters[0].Variables.OrificeLiquid.Reserved1	0
[+] AFC56.Meters[0].Variables.OrificeLiquid.Reserved2	0

### 3) Meter Type = Pulse & Product Group = Gas

Process Input	Controller Tag	Data Type
Temperature	AFC56.Meters[x].Variables.PulseGas.Temperature	REAL
Pressure	AFC56.Meters[x].Variables.PulseGas.Pressure	REAL
Pulse Count	AFC56.Meters[x].Variables.PulseGas.Meter_Pulses	DINT
Pulse Frequency	AFC56.Meters[x].Variables.PulseGas.Frequency	REAL

Where x can assume values between 0 and 15.

Follows below an example for Meter 1:

[-] AFC56.Meters[0].Variables.PulseGas	{...}
[+] AFC56.Meters[0].Variables.PulseGas.Reserved1	0
- AFC56.Meters[0].Variables.PulseGas.Temperature	13.32
- AFC56.Meters[0].Variables.PulseGas.Pressure	52.119999
[+] AFC56.Meters[0].Variables.PulseGas.Meter_Pulses	10
[+] AFC56.Meters[0].Variables.PulseGas.Reserved2	0
[+] AFC56.Meters[0].Variables.PulseGas.Reserved3	0
- AFC56.Meters[0].Variables.PulseGas.Pulse_Frequency	42.98

### 4) Meter Type = Pulse & Product Group = Liquid

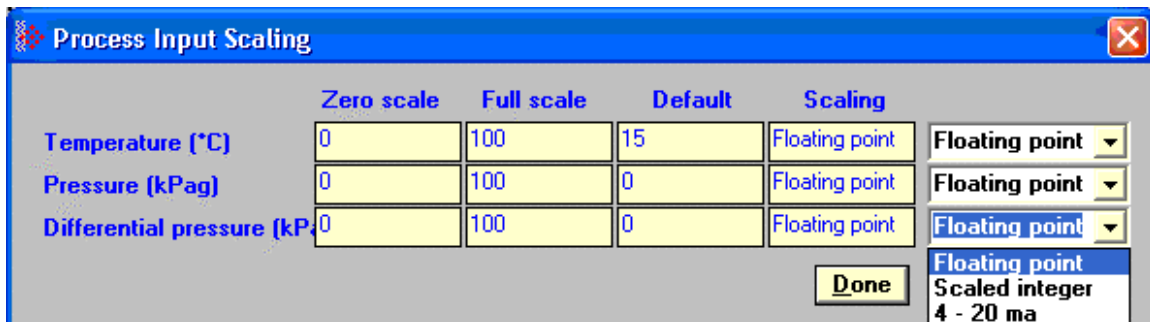
Process Input	Controller Tag	Data Type
Water Percent	AFC56.Meters[x].Variables.PulseLiquid.Water_Percent	INT
Temperature	AFC56.Meters[x].Variables.PulseLiquid.Temperature	REAL
Pressure	AFC56.Meters[x].Variables.PulseLiquid.Pressure	REAL
Pulse Count	AFC56.Meters[x].Variables.PulseLiquid.Meter_Pulses	DINT
Density	AFC56.Meters[x].Variables.PulseLiquid.Density	REAL
Pulse Frequency	AFC56.Meters[x].Variables.PulseLiquid.Pulse_Frequency	REAL

Where x can assume values between 0 and 15.

Follows below an example for Meter 1:

[-] AFC56.Meters[0].Variables.PulseLiquid	{...}
[+] AFC56.Meters[0].Variables.PulseLiquid.Water_Percent	13
- AFC56.Meters[0].Variables.PulseLiquid.Temperature	32.98
- AFC56.Meters[0].Variables.PulseLiquid.Pressure	23.780001
[+] AFC56.Meters[0].Variables.PulseLiquid.Meter_Pulses	13222
- AFC56.Meters[0].Variables.PulseLiquid.Density	23.42
- AFC56.Meters[0].Variables.PulseLiquid.Pulse_Frequency	64.199997

Important: The sample ladder logic is configured considering the input variables with floating point format (default from AFC Manager). However, if Scaled Integer or 4-20 mA formats are used, the user should change the meter variables format from floating point (REAL) to 32-bit long integer (DINT) in the ladder logic.



The *AFC56.Meters[0].Variables.BID* is the function block ID for each meter. This value is automatically calculated by the ladder logic, so the user does not have to move any value to this controller tag.

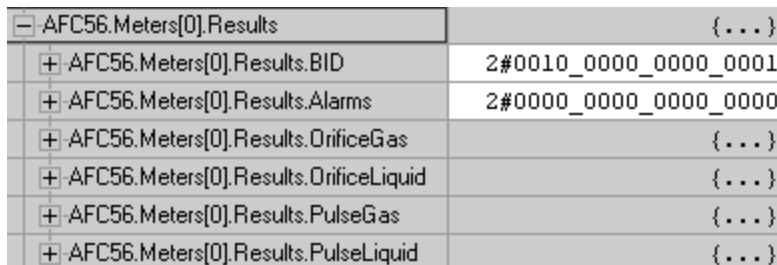
## - Meter Calculation Results

Once the module has performed the AGA/API calculation, all results are moved to the processor. The ladder logic will move the calculation results to the controller tags described in this section.

The calculation results will depend on the meter type and product group. So the profile information (discussed before) is used to decide to which controller tag the results will be copied to. The following options are used:

Meter Type	Product Group	Use this Controller Tag
Orifice	Gas	<i>AFC56.Meters[x].Results.OrificeGas</i>
Orifice	Liquid	<i>AFC56.Meters[x].Results.OrificeLiquid</i>
Pulse	Gas	<i>AFC56.Meters[x].Results.PulseGas</i>
Pulse	Liquid	<i>AFC56.Meters[x].Results.PulseLiquid</i>

Follows below the controller tag screenshot from RSLogix5000 :



- AFC56.Meters[0].Results	{ ... }
+ AFC56.Meters[0].Results.BID	2#0010_0000_0000_0001
+ AFC56.Meters[0].Results.Alarms	2#0000_0000_0000_0000
+ AFC56.Meters[0].Results.OrificeGas	{ ... }
+ AFC56.Meters[0].Results.OrificeLiquid	{ ... }
+ AFC56.Meters[0].Results.PulseGas	{ ... }
+ AFC56.Meters[0].Results.PulseLiquid	{ ... }

So this means that all the user has to do is to identify where the results will be copied to (based on the meter type and product group) since the ladder logic will automatically select the correct controller tags.

Obs: In order to configure each meter type and product group the user should refer to the AFC manager software tool.



**Corporate Office:**  
 1675 Chester Avenue  
 Fourth Floor  
 Bakersfield, CA 93301

661/716-5100  
 Fax: 661/716-5101  
 e-mail: prosoft@prosoft-technology.com

For each possible combination, the following variables are used:

**1) Meter Type = Orifice & Product Group = Gas**

Calculation Result	Controller Tag	Data Type
Net Accumulator	AFC56.Meters[x].Results.OrificeGas.Net_Accumulator	DINT
Net Flow Rate	AFC56.Meters[x].Results.OrificeGas.Net_Flowrate	REAL
Gross Flow Rate	AFC56.Meters[x].Results.OrificeGas.Gross_Flowrate	REAL
Fpv	AFC56.Meters[x].Results.OrificeGas.Fpv	REAL
Cprime	AFC56.Meters[x].Results.OrificeGas.Cprime	REAL

Where x can assume values between 0 and 15.

Follows below an example for Meter 1:

- AFC56.Meters[0].Results.OrificeGas	{...}
+ AFC56.Meters[0].Results.OrificeGas.Net_Accumulator	129777
- AFC56.Meters[0].Results.OrificeGas.Net_Flowrate	13.98
- AFC56.Meters[0].Results.OrificeGas.Gross_Flowrate	15.64
- AFC56.Meters[0].Results.OrificeGas.Fpv	0.97000003
- AFC56.Meters[0].Results.OrificeGas.Cprime	0.86000001

**2) Meter Type = Orifice & Product Group = Liquid**

Calculation Result	Controller Tag	Data Type
Net Accumulator	AFC56.Meters[x].Results.OrificeLiquid.Net_Accumulator	DINT
Net Flow Rate	AFC56.Meters[x].Results.OrificeLiquid.Net_Flowrate	REAL
Gross Accumulator	AFC56.Meters[x].Results.OrificeLiquid.Gross_Accumulator	DINT
Gross Standard Accumulator	AFC56.Meters[x].Results.OrificeLiquid.Standard_Accumulator	DINT

Mass Accumulator	AFC56.Meters[x].Results.OrificeLiquid.Mass_Accumulator	DINT
------------------	--	------

Where x can assume values between 0 and 15.

Follows an example for Meter 1:

- AFC56.Meters[0].Results.OrificeLiquid	{...}
+ AFC56.Meters[0].Results.OrificeLiquid.Net_Accumulator	533233
- AFC56.Meters[0].Results.OrificeLiquid.Net_Flowrate	13.43
+ AFC56.Meters[0].Results.OrificeLiquid.Gross_Accumulator	602522
+ AFC56.Meters[0].Results.OrificeLiquid.Gross_Standard_Accumulator	540644
+ AFC56.Meters[0].Results.OrificeLiquid.Mass_Accumulator	300127

### 3) Meter Type = Pulse & Product Group = Gas

Calculation Result	Controller Tag	Data Type
Net Accumulator	AFC56.Meters[x].Results.PulseGas.Net_Accumulator	DINT
Net Flow Rate	AFC56.Meters[x].Results.PulseGas.Net_Flowrate	REAL
Gross Flow Rate	AFC56.Meters[x].Results.PulseGas.Gross_Flowrate	REAL
Fpv	AFC56.Meters[x].Results.PulseGas.Fpv	REAL
Cprime	AFC56.Meters[x].Results.PulseGas.Cprime	REAL

Where x can assume values between 0 and 15.

Follows an example for Meter 1:

- AFC56.Meters[0].Results.PulseGas	{...}
+ AFC56.Meters[0].Results.PulseGas.Net_Accumulator	455233
- AFC56.Meters[0].Results.PulseGas.Net_Flowrate	32.233002
- AFC56.Meters[0].Results.PulseGas.Gross_Flowrate	40.743999
- AFC56.Meters[0].Results.PulseGas.Fpv	0.97860003
- AFC56.Meters[0].Results.PulseGas.CPrime	0.75749999

#### 4) Meter Type = Pulse & Product Group = Liquid

Calculation Result	Controller Tag	Data Type
Net Accumulator	AFC56.Meters[x].Results.PulseLiquid.Net_Accumulator	DINT
Net Flow Rate	AFC56.Meters[x].Results.PulseLiquid.Net_Flowrate	REAL
Gross Accumulator	AFC56.Meters[x].Results.PulseLiquid.Gross_Accumulator	DINT
Gross Standard Accumulator	AFC56.Meters[x].Results.PulseLiquid.Gross_Standard_Accumulator	DINT
Mass Accumulator	AFC56.Meters[x].Results.PulseLiquid.Mass_Accumulator	DINT

Where x can assume values between 0 and 15.

Follows an example for Meter 1:

[-] AFC56.Meters[0].Results.PulseLiquid	{ ... }
[+] AFC56.Meters[0].Results.PulseLiquid.Net_Accumulator	466896
[-] AFC56.Meters[0].Results.PulseLiquid.Net_Flowrate	32.630001
[+] AFC56.Meters[0].Results.PulseLiquid.Gross_Accumulator	500233
[+] AFC56.Meters[0].Results.PulseLiquid.Gross_Standard_Accumulator	480422
[+] AFC56.Meters[0].Results.PulseLiquid.Mass_Accumulator	400533

## - Molar Analysis (for gas product only)

If the application uses a chromatograph to send the molar concentrations to the module, the sample ladder may dynamically supply all molar concentrations to the MVI56-AFC.

Initially, the user should check the "Selected" checkboxes for all elements using the AFC Manager (clicking on the Analysis button in the Meter Configuration window).

In order to write the molar concentration values from the ladder logic, the user should set the *AFC56.Analysis.Enable* bit. After that, any molar concentration configuration performed through AFC Manager will be overwritten by the ladder logic.

[-] AFC56.Analysis	{...}
[-] AFC56.Analysis.Enable	1
[+] AFC56.Analysis.MeterNumber	0
[+] AFC56.Analysis.BID	0

All the user has to do is to refer to the Meters[x].Analysis controller tag in order to move the concentrations for each meter (x assumes values between 0 and 15).

- AFC56.Meters[0].Analysis	{...}
+ AFC56.Meters[0].Analysis.C1	5000
+ AFC56.Meters[0].Analysis.N2	1700
+ AFC56.Meters[0].Analysis.CO2	300
+ AFC56.Meters[0].Analysis.C2	0
+ AFC56.Meters[0].Analysis.C3	0
+ AFC56.Meters[0].Analysis.H2O	0
+ AFC56.Meters[0].Analysis.H2S	0
+ AFC56.Meters[0].Analysis.H2	0
+ AFC56.Meters[0].Analysis.CO	0
+ AFC56.Meters[0].Analysis.O2	0
+ AFC56.Meters[0].Analysis.IC4	3000
+ AFC56.Meters[0].Analysis.NC4	0
+ AFC56.Meters[0].Analysis.IC5	0
+ AFC56.Meters[0].Analysis.NC5	0
+ AFC56.Meters[0].Analysis.C6	0
+ AFC56.Meters[0].Analysis.C7	0
+ AFC56.Meters[0].Analysis.C8	0
+ AFC56.Meters[0].Analysis.C9	0
+ AFC56.Meters[0].Analysis.C10	0
+ AFC56.Meters[0].Analysis.He	0
+ AFC56.Meters[0].Analysis.Ar	0
+ AFC56.Meters[0].Analysis.NeoC5	0
+ AFC56.Meters[0].Analysis.Ux_U...	0
+ AFC56.Meters[0].Analysis.Uy_U...	0

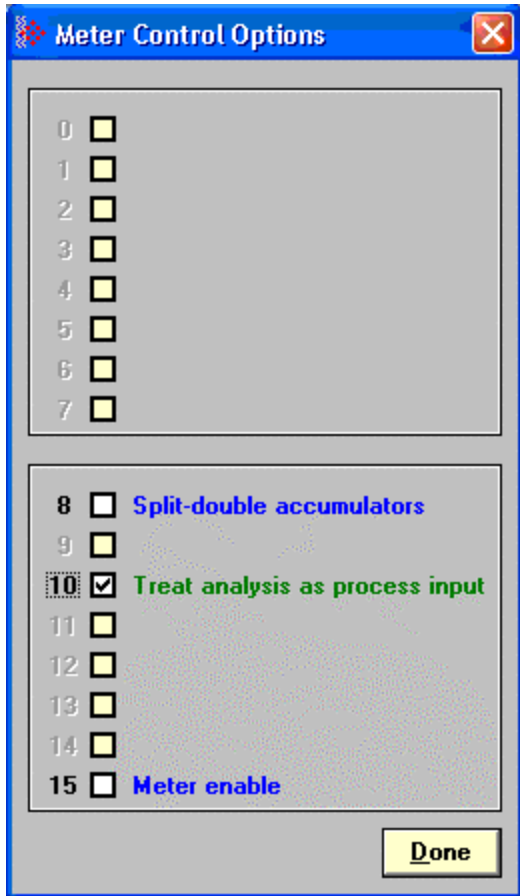
The concentrations are entered as scaled integer format where 10000 = 100%. For example:

$$C1 = 9168 \Rightarrow 91.68\%$$

The sum of all concentration should be 100%. Sometimes the chromatograph can generate values which total is slightly less (or more) than 100%. In this case the user should configure the Normalization Error Tolerance parameter in the AFC Manager in order to make sure that the module will not generate any alarms.

When the module detects that a molar concentration value has changed it will generate an event. However, when the values are updated from ladder logic using a chromatograph device it is not convenient to generate an alarm every time a concentration value changes. In this case the user may configure the module to not generate any alarms when a molar value is modified.

So the user should check the "Meter Configuration-> Ctrl Options->Treat Analysis as Process Input" checkbox:



### - Reset Resettable Accumulator

The sample ladder logic shows how to reset all resettable accumulators for a meter. However, most customers will prefer to reset the resettable accumulators when the archives are created. The module can be configured to automatically reset the resettable accumulators upon period end using Meter-Relative words Mh00341 (daily archive) and Mh00421 (hourly archive) in the Primary Modbus Slave. In order to use this feature, no ladder logic is required.

Please refer to Appendix B for further information about this subject.

Some customers may decide to reset the resettable accumulators from the ladder logic (specially if using batch operation).



Corporate Office:  
1675 Chester Avenue  
Fourth Floor  
Bakersfield, CA 93301

661/716-5100  
Fax: 661/716-5101  
e-mail: prosoft@prosoft-technology.com

In this case the user should toggle the following bit commands:

Command	Use this Controller Tag	Description
Enable	<i>AFC56.Reset.Enable</i>	This value should be set as 1 in order to enable the command. If this value is 0 the module will ignore the reset commands from the ladder
Select Meter	<i>AFC56.Reset.Meter</i>	Select the meter number (1 to 16) for the reset command
Reset Resttable Accumulator 1	<i>AFC56.Reset.Signals.Res_Acc1</i>	If this bit is set to 1 the module will reset Acc1 for the selected meter. The ladder logic will reset this command.
Reset Resttable Accumulator 2	<i>AFC56.Reset.Signals.Res_Acc2</i>	If this bit is set to 1 the module will reset Acc2 for the selected meter. The ladder logic will reset this command.
Reset Resttable Accumulator 3	<i>AFC56.Reset.Signals.Res_Acc3</i>	If this bit is set to 1 the module will reset Acc3 for the selected meter. The ladder logic will reset this command.
Reset Resttable Accumulator 4	<i>AFC56.Reset.Signals.Res_Acc4</i>	If this bit is set to 1 the module will reset Acc4 for the selected meter. The ladder logic will reset this command.

Obs: the *AFC56.Reset.BID* and *AFC56.Reset.Action* tags are automatically updated by the ladder logic s the user does not have to enter any value for this tags.

Follows the screenshot for this data structure. Note that the Sel\_Stream bits are not used for this MVI56-AFC version.



**Corporate Office:**  
1675 Chester Avenue  
Fourth Floor  
Bakersfield, CA 93301

661/716-5100  
Fax: 661/716-5101  
e-mail: prosoft@prosoft-technology.com

[-] AFC56.Reset	{...}
[-] AFC56.Reset.Enable	1
[+] AFC56.Reset.BID	0
[+] AFC56.Reset.Meter	1
[+] AFC56.Reset.Action	0
[-] AFC56.Reset.Signals	{...}
[-] AFC56.Reset.Signals.Sel_Stream1	0
[-] AFC56.Reset.Signals.Sel_Stream2	0
[-] AFC56.Reset.Signals.Sel_Stream3	0
[-] AFC56.Reset.Signals.Sel_Stream4	0
[-] AFC56.Reset.Signals.Res_Acc1	0
[-] AFC56.Reset.Signals.Res_Acc2	0
[-] AFC56.Reset.Signals.Res_Acc3	0
[-] AFC56.Reset.Signals.Res_Acc4	0
[-] AFC56.Reset.Signals.Wr_Daily_Archive	0
[-] AFC56.Reset.Signals.Wr_Hrly_Archive	0



**Corporate Office:**  
1675 Chester Avenue  
Fourth Floor  
Bakersfield, CA 93301

661/716-5100  
Fax: 661/716-5101  
e-mail: prosoft@prosoft-technology.com

## - Write Hourly/Daily Archive

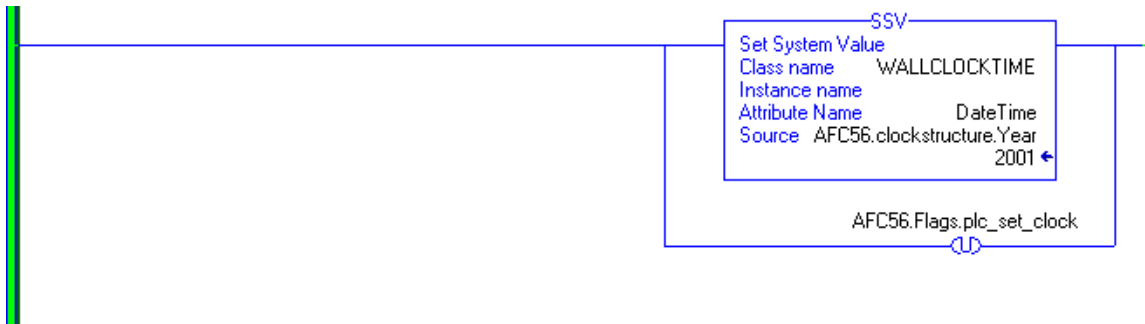
The sample ladder logic shows how to create hourly or daily archive. Its very important to notice that the sample ladder automatically creates both archives, depending on the configured **End-of-day minute** and **End-of-hour minute** parameters entered in the AFC Manager. So most of the users will not need to generate the archives from ladder logic.

In order to create archives from ladder logic, the user should refer to the following data tags:

<b>Command</b>	<b>Use this Controller Tag</b>	<b>Description</b>
Enable	<i>AFC56.Reset.Enable</i>	This value should be set as 1 in order to enable the command. If this value is 0 the module will ignore the commands from the ladder
Select Meter	<i>AFC56.Reset.Meter</i>	Select the meter number (1 to 16) for the archive command
Write Daily Archive	<i>AFC56.Reset.Signals. Wr_Daily_Archive</i>	If this bit is set to 1 the module will generate a daily archive
Write Hourly Archive	<i>AFC56.Reset.Signals. Wr_Hrly_Archive</i>	If this bit is set to 1 the module will generate an hourly archive

## - Set the Processor Time

This logic was added for convenience purposes only. It shows how to set the processor time and date information using a controller tag as the source:



The source of the information is:

Value	Controller Tag
Year	<i>AFC56.clockstructure.Year</i>
Month	<i>AFC56.clockstructure.Month</i>
Day	<i>AFC56.clockstructure.Day</i>
Hour	<i>AFC56.clockstructure.Hour</i>
Minute	<i>AFC56.clockstructure.Minute</i>
Seconds	<i>AFC56.clockstructure.Second</i>
Miliseconds	<i>AFC56.clockstructure.msec</i>

[-] AFC56.clockstructure	{...}
[+] AFC56.clockstructure.Year	2001
[+] AFC56.clockstructure.Month	3
[+] AFC56.clockstructure.Day	30
[+] AFC56.clockstructure.Hour	8
[+] AFC56.clockstructure.Minute	8
[+] AFC56.clockstructure.second	8
[+] AFC56.clockstructure.msec	0



**Corporate Office:**  
1675 Chester Avenue  
Fourth Floor  
Bakersfield, CA 93301

661/716-5100  
Fax: 661/716-5101  
e-mail: [prosoft@prosoft-technology.com](mailto:prosoft@prosoft-technology.com)

The ladder logic should toggle the following bit in order to transfer the date and time information to the processor.

<b>Command</b>	<b>Controller Tag</b>
Write Clock	<i>AFC56.Flags.plc_set_clock</i>



**Corporate Office:**  
1675 Chester Avenue  
Fourth Floor  
Bakersfield, CA 93301

661/716-5100  
Fax: 661/716-5101  
e-mail: prosoft@prosoft-technology.com

## - Checking Meter Alarms

The ladder logic continuously informs if a meter has an alarm or not. Refer to the following controller tags for the meter alarm status:

<b>Information</b>	<b>Controller Tag</b>	<b>Values</b>
Meter 1 Alarm Status	<i>AFC56.Site_Alarms.Meter1</i>	0 = Meter 1 does not have alarm 1 = Meter 1 has alarm
Meter 2 Alarm Status	<i>AFC56.Site_Alarms.Meter2</i>	0 = Meter 2 does not have alarm 1 = Meter 2 has alarm
Meter 3 Alarm Status	<i>AFC56.Site_Alarms.Meter3</i>	0 = Meter 3 does not have alarm 1 = Meter 3 has alarm
Meter 4 Alarm Status	<i>AFC56.Site_Alarms.Meter4</i>	0 = Meter 4 does not have alarm 1 = Meter 4 has alarm
...	...	...
Meter 16 Alarm Status	<i>AFC56.Site_Alarms.Meter16</i>	0 = Meter 16 does not have alarm 1 = Meter 16 has alarm

For the example below, meter 1 and meter 2 has alarms. All other meter does not have any alarms:



**Corporate Office:**  
 1675 Chester Avenue  
 Fourth Floor  
 Bakersfield, CA 93301

661/716-5100  
 Fax: 661/716-5101  
 e-mail: prosoft@prosoft-technology.com

-AFC56.Site_Alarms	{...}
-AFC56.Site_Alarms.Meter1	1
-AFC56.Site_Alarms.Meter2	1
-AFC56.Site_Alarms.Meter3	0
-AFC56.Site_Alarms.Meter4	0
-AFC56.Site_Alarms.Meter5	0
-AFC56.Site_Alarms.Meter6	0
-AFC56.Site_Alarms.Meter7	0
-AFC56.Site_Alarms.Meter8	0
-AFC56.Site_Alarms.Meter9	0
-AFC56.Site_Alarms.Meter10	0
-AFC56.Site_Alarms.Meter11	0
-AFC56.Site_Alarms.Meter12	0
-AFC56.Site_Alarms.Meter13	0
-AFC56.Site_Alarms.Meter14	0
-AFC56.Site_Alarms.Meter15	0
-AFC56.Site_Alarms.Meter16	0

For each meter, the ladder logic also informs which alarm was generated. Refer to the following controller tags for the meter alarm information:

<b>Information</b>	<b>Controller Tag</b>	<b>Values</b>
Meter 1 Alarm	<i>AFC56.Meters[0].Results.Alarms</i>	Please see table below
Meter 2 Alarm	<i>AFC56.Meters[1].Results.Alarms</i>	Please see table below
Meter 3 Alarm	<i>AFC56.Meters[2].Results.Alarms</i>	Please see table below
Meter 4 Alarm	<i>AFC56.Meters[3].Results.Alarms</i>	Please see table below
...	...	...
Meter 16 Alarm	<i>AFC56.Meters[15].Results.Alarms</i>	Please see table below

Each Alarm word is interpreted as follows:

<b>Bit Number</b>	<b>Description</b>
0	Input out of range: Temperature
1	Input out of range: Pressure
2	Input out of range: Differential Pressure
3	Input out of range: Flowing Density
4	Input out of range: Water Content



**Corporate Office:**  
1675 Chester Avenue  
Fourth Floor  
Bakersfield, CA 93301

661/716-5100  
Fax: 661/716-5101  
e-mail: prosoft@prosoft-technology.com

5	Deifferential Pressure Low
6	Reserved
7	Reserved
8	Orifice Characterization Error
9	Analysis Total Zero
10	Analysis Total Not Normalized
11	AGA 8 Calculation Error
12	API Calculation Error: Density Correction
13	API Calculation Error: Ctl
14	API Calculation Error: Vapor Pressure
15	API Calculation Error: Cpl

The following screenshot shows an example where the meter 1 has an “ **Input Out of Range: Temperature** ” alarm

- AFC56.Meters[0].Results	{...}
+ AFC56.Meters[0].Results.BID	2#0010_0010_0000_0001
+ AFC56.Meters[0].Results.Alarms	2#0000_0000_0000_0001
+ AFC56.Meters[0].Results.OrificeGas	{...}
+ AFC56.Meters[0].Results.OrificeLiquid	{...}
+ AFC56.Meters[0].Results.PulseGas	{...}
+ AFC56.Meters[0].Results.PulseLiquid	{...}



**Corporate Office:**  
 1675 Chester Avenue  
 Fourth Floor  
 Bakersfield, CA 93301

661/716-5100  
 Fax: 661/716-5101  
 e-mail: prosoft@prosoft-technology.com

## - Site Status

The ladder logic continuously reads the site status from the MVI56-AFC. Follows the controller tags that are used:

<b>Information</b>	<b>Controller Tag</b>
AFC56(16) Released	<i>AFC56.Site_Status.AFC_ACTIVE</i>
Checksum Alarms	<i>AFC56.Site_Status.Checksum_Alarms</i>
PLC Halted	<i>AFC56.Site_StatusPLC_Halted</i>
Cfg Changed	<i>AFC56.Site_Status.Cfg_changed</i>
Power Up	<i>AFC56.Site_Status.Powerup</i>
Cold Start	<i>AFC56.Site_Status.ColdStart</i>

[-] AFC56.Site_Status	{...}
[-] AFC56.Site_Status.AFC_ACTIVE	1
[-] AFC56.Site_Status.Checksum_Alarms	0
[-] AFC56.Site_Status.reserved2	0
[-] AFC56.Site_Status.reserved3	0
[-] AFC56.Site_Status.PLC_HALTED	0
[-] AFC56.Site_Status.Cfg_changed	0
[-] AFC56.Site_Status.Powerup	0
[-] AFC56.Site_Status.ColdStart	0
[-] AFC56.Site_Status.reserved8	0
[-] AFC56.Site_Status.reserved9	0
[-] AFC56.Site_Status.reserved10	0
[-] AFC56.Site_Status.reserved11	0
[-] AFC56.Site_Status.reserved12	0
[-] AFC56.Site_Status.reserved13	0
[-] AFC56.Site_Status.reserved14	0
[-] AFC56.Site_Status.reserved15	0

## - Modbus Gateway

The ladder logic can be used to read or write data from one of the internal Modbus Slaves. Any data that is not part of the AFC56.Meters[.].Results will have to be read or written through the Modbus Gateway blocks.

Each block can transfer up to 200 words of data and uses a specific *AFC56.Modbus.Gateway[.]* controller tag. Each one of these tags must be configurable in order to read or write data between the module and the processor.

Follows below the simple steps for the user in order to use the Modbus Gateway blocks:

- 1) First of all the user should identify how many words (total) will be transferred. The sample ladder supports up to 2000 words.
- 2) Based on the number of registers to be transferred, the user should calculate how many Modbus Gateway blocks will be necessary. Each block contains up to 200 registers. For example: if 700 registers will be used to transfer data then 4 Modbus Gateway blocks will be required.
- 3) Based on the number of Modbus Gateway blocks, the user should configure the *AFC56.Modbus.BlockCount* controller tag.

- AFC56.Modbus	{...}
+ AFC56.Modbus.BlockCount	4
+ AFC56.Modbus.Gateway	{...}

For example: if the user configures the number of blocks as 4, the ladder logic will automatically send the following Modbus Gateway blocks to the module:

```
AFC56.Modbus.Gateway[0]
AFC56.Modbus.Gateway[1]
AFC56.Modbus.Gateway[2]
AFC56.Modbus.Gateway[3]
```

The maximum number of blocks is 10.

If the *AFC56.Modbus.BlockCount* controller tag is configured as 0 the module will not send any Modbus Gateway blocks.

4) Refer to the Modbus Map document and identify the addresses of all registers in the Primary Slave.

5) Using the AFC Manager, remmap the registers from the Primary Slave to the Virtual Slave (refer to AFC Manager User Manual for further information about this subject). The user must also set a Virtual Slave Address greater than 0 in order to activate the Virtual Slave.

6) In the sample ladder logic, configure each Modbus Gateway Block using the following controller tags:

<b>Parameter</b>	<b>Controller Tag</b>	<b>Values</b>
Enable Transaction	<i>AFC56.Modbus.Gateway[0].Config.Enable</i>	0 = The module will ignore this request 1 = The module will process this request.
Start Register	<i>AFC56.Modbus.Gateway[0].Config.StartRegister</i>	Start register in the Modbus Slave to be written or read from
Register Count	<i>AFC56.Modbus.Gateway[0].Config.RegisterCount</i>	Number of words to be written or read between the module and the processor.
Function Type	<i>AFC56.Modbus.Gateway[0].Config.FunctionType_Write</i>	0 = Read from MVI56-AFC 1 = Write to MVI56-AFC
Register Type	<i>AFC56.Modbus.Gateway[0].Config.RegisterType_Input</i>	0 = Holding Register 1 = Input Register
Slave Type	<i>AFC56.Modbus.Gateway[0].Config.SlaveType_Virtual</i>	0 = Primary Slave 1 = Virtual Slave

Obs: It is strongly suggested that the user should first configure all parameters having the Enable bit set to 0. Once the configuration is finished than the Enable bit can be set to 1.

### **Example:**

In order to write 200 words from the processor to the Primary Modbus Slave starting at holding register address 2000, the *AFC56.Modbus.Gateway[0]* block should be configured as follows:

[-] AFC56.Modbus.Gateway[0].Config	{...}
[-] AFC56.Modbus.Gateway[0].Config.Enable	1
[+] AFC56.Modbus.Gateway[0].Config.StartRegister	2000
[+] AFC56.Modbus.Gateway[0].Config.RegisterCount	200
[-] AFC56.Modbus.Gateway[0].Config.FunctionType_Write	1
[-] AFC56.Modbus.Gateway[0].Config.RegisterType_Input	0
[-] AFC56.Modbus.Gateway[0].Config.SlaveType_Virtual	0

7) Refer to the *AFC56.Modbus.Gateway[x].WriteData* or *AFC56.Modbus.Gateway[x].ReadData* controller tags depending on the configured function type:

- If the Modbus Gateway block uses a **READ** function type:

The data will be read to the *AFC56.Modbus.Gateway[x].ReadData[ ]* array.

- If the Modbus Gateway block uses a **WRITE** function type:

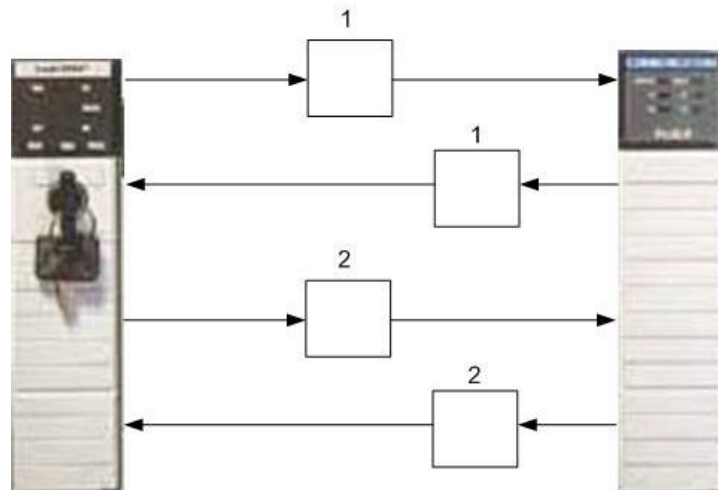
The data will be written from the *AFC56.Modbus.Gateway[x].WriteData[ ]* array.

Obs: The BID and BOD controller tags are automatically generated by the ladder logic so the user does not have to write any value to these controller tags.

## How does the sample ladder works?

This section presents a general overview on how the sample ladder logic works.. For further information about specific rungs, please refer to the ladder logic comments in the ladder file.

The ladder logic consists on basically sending output blocks to the MVI56-AFC and receiving the input blocks from the module. Each block contains a Block Sequence number that identifies the block. The input response block sent by the module will also contain the same Block Sequence Number as the previous output block.



So the ladder logic basically performs the following sequence:

- 1 – Receives the input block from the MVI56-AFC
- 2 – Copies the input block content from the input buffer to the controller tags based on the Block Sequence Number
- 3 – Increments the next Block Sequence Number
- 4 – Builds the next output block with the new Sequence Number
- 5 – Sends the new output block to the module.



**Corporate Office:**  
1675 Chester Avenue  
Fourth Floor  
Bakersfield, CA 93301

661/716-5100  
Fax: 661/716-5101  
e-mail: prosoft@prosoft-technology.com

The sample ladder uses the following Block Sequence Number to generate output blocks:

<b>Block Sequence Number</b>	<b>Description</b>
1	Process variables, enable/disable (all Meters). Write Molar Analysis (Meter 1) and read profile (Meter 1)
2	Process variables, enable/disable (all Meters). Write Molar Analysis (Meter 2) and read profile (Meter 2)
3	Process variables, enable/disable (all Meters). Write Molar Analysis (Meter 3) and read profile (Meter 3)
4	Process variables, enable/disable (all Meters). Write Molar Analysis (Meter 4) and read profile (Meter 4)
5	Process variables, enable/disable (all Meters). Write Molar Analysis (Meter 5) and read profile (Meter 5)
6	Process variables, enable/disable (all Meters). Write Molar Analysis (Meter 6) and read profile (Meter 6)
7	Process variables, enable/disable (all Meters). Write Molar Analysis (Meter 7) and read profile (Meter 7)
8	Process variables, enable/disable (all Meters). Write Molar Analysis (Meter 8) and read profile (Meter 8)
9	Process variables, enable/disable (all Meters). Write Molar Analysis (Meter 9) and read profile (Meter 9)
10	Process variables, enable/disable (all Meters). Write Molar Analysis (Meter 10) and read profile (Meter 10)
11	Process variables, enable/disable (all Meters). Write Molar Analysis (Meter 11) and read profile (Meter 11)
12	Process variables, enable/disable (all Meters). Write Molar Analysis (Meter 12) and read profile (Meter 12)
13	Process variables, enable/disable (all Meters). Write Molar Analysis (Meter 13) and read profile (Meter 13)
14	Process variables, enable/disable (all Meters). Write Molar Analysis (Meter 14) and read profile (Meter 14)
15	Process variables, enable/disable (all Meters). Write Molar Analysis (Meter 15) and read profile (Meter 15)
16	Process variables, enable/disable (all Meters). Write Molar Analysis (Meter 16) and read profile (Meter 16)



**Corporate Office:**  
 1675 Chester Avenue  
 Fourth Floor  
 Bakersfield, CA 93301

661/716-5100

Fax: 661/716-5101

e-mail: prosoft@prosoft-technology.com

17	Modbus Gateway Block 0 (if configured)
18	Modbus Gateway Block 1 (if configured)
19	Modbus Gateway Block 2 (if configured)
20	Modbus Gateway Block 3 (if configured)
21	Modbus Gateway Block 4 (if configured)
22	Modbus Gateway Block 5 (if configured)
23	Modbus Gateway Block 6 (if configured)
24	Modbus Gateway Block 7 (if configured)
25	Modbus Gateway Block 8 (if configured)
25	Modbus Gateway Block 9 (if configured)

Each Process Block (Block Sequence Number 1 to 16) is organized as follows:

**Process Block ( uses Transaction Numbers from 0 to 16)**

<b>Controller Tag Begin</b>	<b>Controller Tag End</b>	<b>Description</b>
AFC56.outputbuffer[0]		Transaction Number [x]
AFC56.outputbuffer[1]		Block Length (226)
AFC56.outputbuffer[2]	AFC56.outputbuffer[13]	Meter 1 Process Variables
AFC56.outputbuffer[14]	AFC56.outputbuffer[25]	Meter 2 Process Variables
AFC56.outputbuffer[26]	AFC56.outputbuffer[37]	Meter 3 Process Variables
AFC56.outputbuffer[38]	AFC56.outputbuffer[49]	Meter 4 Process Variables
AFC56.outputbuffer[50]	AFC56.outputbuffer[61]	Meter 5 Process Variables
AFC56.outputbuffer[62]	AFC56.outputbuffer[73]	Meter 6 Process Variables
AFC56.outputbuffer[74]	AFC56.outputbuffer[85]	Meter 7 Process Variables
AFC56.outputbuffer[86]	AFC56.outputbuffer[97]	Meter 8 Process Variables
AFC56.outputbuffer[98]	AFC56.outputbuffer[109]	Meter 9 Process Variables
AFC56.outputbuffer[110]	AFC56.outputbuffer[121]	Meter 10 Process Variables
AFC56.outputbuffer[122]	AFC56.outputbuffer[133]	Meter 11 Process Variables
AFC56.outputbuffer[134]	AFC56.outputbuffer[145]	Meter 12 Process Variables
AFC56.outputbuffer[146]	AFC56.outputbuffer[157]	Meter 13 Process Variables
AFC56.outputbuffer[158]	AFC56.outputbuffer[169]	Meter 14 Process Variables
AFC56.outputbuffer[170]	AFC56.outputbuffer[181]	Meter 15 Process Variables
AFC56.outputbuffer[182]	AFC56.outputbuffer[193]	Meter 16 Process Variables
AFC56.outputbuffer[194]	AFC56.outputbuffer[195]	Enable Meters
AFC56.outputbuffer[196]	AFC56.outputbuffer[197]	Disable Meters
AFC56.outputbuffer[198]	AFC56.outputbuffer[199]	Reset Acc, Write Archives
AFC56.outputbuffer[200]	AFC56.outputbuffer[224]	Molar Analysis for Meter [x]
AFC56.outputbuffer[225]	AFC56.outputbuffer[227]	Read Meter [x] Profile
AFC56.outputbuffer[226]	AFC56.outputbuffer[246]	Not Used

AFC56.outputbuffer[247]		Transaction Number [x]
-------------------------	--	------------------------

So once the module receives this output block, it will perform the required calculations and then build the input response block to be sent to the processor. The module will use the same offsets as the output block. For example: the calculation results for Meter 9 will be copied at offset 98 in the input block.

### **Modbus Gateway Block ( uses Transaction Numbers from 17 to 26)**

Each Modbus Gateway block has the following structure:

Controller Tag Begin	Controller Tag End	Description
AFC56.outputbuffer[0]		Transaction Number [y]
AFC56.outputbuffer[1]		Block Length <sup>1</sup>
AFC56.outputbuffer[2]		Modbus Gateway BOD
AFC56.outputbuffer[3]		Start Register
AFC56.outputbuffer[4]		Register Count
AFC56.outputbuffer[5]	AFC56.outputbuffer[204]	Reserved for Modbus Gateway
AFC56.outputbuffer[205]	AFC56.outputbuffer[246]	Not Used
AFC56.outputbuffer[247]		Transaction Number [y]

<sup>1</sup> The block length will depend on the Register Count parameter entered by the user.

### **Wallclock Block (uses Transaction Number =99)**

The wallclock block has the following structure:

Controller Tag Begin	Controller Tag End	Description
AFC56.outputbuffer[0]		Transaction Number = 99
AFC56.outputbuffer[1]		Block Length = 7
AFC56.outputbuffer[2]		Wallclock BOD
AFC56.outputbuffer[3]		Year
AFC56.outputbuffer[4]		Month



**Corporate Office:**  
 1675 Chester Avenue  
 Fourth Floor  
 Bakersfield, CA 93301

661/716-5100  
 Fax: 661/716-5101  
 e-mail: prosoft@prosoft-technology.com

AFC56.outputbuffer[5]		Day
AFC56.outputbuffer[6]		Hour
AFC56.outputbuffer[7]		Minute
AFC56.outputbuffer[8]		Seconds
AFC56.outputbuffer[9]		Transaction Number = 99

## APPENDIX A - Modbus Gateway Application

This appendix shows a sample application that shows how the user would read specific data from the MVI56-AFC using the Modbus Gateway blocks.

Q: I want to read the Net Accumulator totalizer values from yesterday's archive (Meters 1 to 4) from the MVI56-AFC to the Controllogix processor. These are all orifice meters (gas product).

Follow the steps below:

- 1) This application will only transfer 8 words (each totalizer occupies 2 words).
- 2) Since the number of words is less than 200, it means that only one Modbus Gateway block will be used (AFC56.Modbus.Gateway[0])
- 3) Configure the *AFC56.Modbus.BlockCount* tag with a value of 1.
- 4) In order to identify the register addresses in the Primary Modbus Slave, the user should refer to the following spreadsheet that shows the addresses for the hourly and daily archives:

### Input Registers

Meter	Start Daily Archive	End Daily Archive	Start Hourly Archive	End Hourly Archive
1	0	1059	1060	2499
2	2500	3559	3560	4999
3	5000	6059	6060	7499
4	7500	8559	8560	9999
5	10000	11059	11060	12499
6	12500	13559	13560	14999
7	15000	16059	16060	17499
8	17500	18559	18560	19999



**Corporate Office:**  
1675 Chester Avenue  
Fourth Floor  
Bakersfield, CA 93301

661/716-5100  
Fax: 661/716-5101  
e-mail: prosoft@prosoft-technology.com

9	20000	21059	21060	22499
10	22500	23559	23560	24999
11	25000	26059	26060	27499
12	27500	28559	28560	29999
13	30000	31059	31060	32499
14	32500	33559	33560	34999
15	35000	36059	36060	37499
16	37500	38559	38560	39999

Each archive occupies a 30-word block. For example, for meter 1:

Archive	Start Register	End Register
Yesterday	0	29
2 days ago	30	59
3 days ago	60	89
4 days ago	90	119
5 days ago	120	149
...	...	

Follows the structure of each archive. The first 10 words are common for all archives. The rest of the archive structure will depend on the meter type or product group:

### **Pre-defined Overhead**

Start Offset	End Offset	Data Type	Description
00	01	Dt	Closing timestamp of archive
02		Wd	Flowing period
03		Bm	Cumulative meter alarms
04		Bm	Cumulative status
05		Wd	Event counter
06	07	Dw	Flowing period, seconds
08	09	Dt	Opening timestamp of archive

### **Orifice Meter with Gas Product**

Start Offset	End Offset	Data Type	Description
10	11	Acc	Accumulator totalizer, net/liqevq
12	13	Fp	Accumulator residue, net/liqevq
14	15	Fp	Flow rate, net/liqevq



**Corporate Office:**  
 1675 Chester Avenue  
 Fourth Floor  
 Bakersfield, CA 93301

661/716-5100

Fax: 661/716-5101

e-mail: prosoft@prosoft-technology.com

16	17	Fp	Temperature, CU
18	19	Fp	Pressure, CUG
20	21	Fp	Differential pressure, CU
22		Wd	Relative density 15øC/15øC, e-4
23		Wd	Compressibility, reference, e-4
24		Wd	Compressibility, flowing, e-4
25		Wd	Fpv, e-4
26		Wd	Velocity of approach factor, Ev, e-4
27		Wd	Expansion factor, Y, e-4
28		Wd	Coefficient of discharge, Cd, e-4
29		Wd	Reserved

### **Pulse Meter with Gas Product**

Start Offset	End Offset	Data Type	Description
10	11	Acc	Accumulator totalizer, net/liqev
12	13	Fp	Accumulator residue, net/liqev
14	15	Fp	Flow rate, net/liqev
16	17	Fp	Temperature, CU
18	19	Fp	Pressure, CUG
20	21	Fp	K-Factor
22	23	Fp	Meter Factor
24		Wd	Relative density 15øC/15øC, e-4
25		Wd	Compressibility, reference, e-4
26		Wd	Compressibility, flowing, e-4
27		Wd	Fpv, e-4
28	29	Wd	Reserved

### **Orifice Meter with Liquid Product**

Start Offset	End Offset	Data Type	Description
10	11	Acc	Accumulator totalizer, net/liqev
12	13	Fp	Accumulator residue, net/liqev
14	15	Fp	Flow rate, net/liqev
16	17	Fp	Temperature, CU
18	19	Fp	Pressure, CUG



**Corporate Office:**  
 1675 Chester Avenue  
 Fourth Floor  
 Bakersfield, CA 93301

661/716-5100  
 Fax: 661/716-5101

e-mail: prosoft@prosoft-technology.com

20	21	Fp	Differential pressure, CU
22	23	Fp	Flowing density, CU
24		Wd	Corrected density, kg/m <sup>3</sup> e-1
25		Wd	Ctl e-4
26		Wd	Cpl e-4
27		Wd	Velocity of approach factor, Ev, e-4
28		Wd	Expansion factor, Y, e-4
29		Wd	Coefficient of discharge, Cd, e-4

**Pulse Meter with Liquid Product**

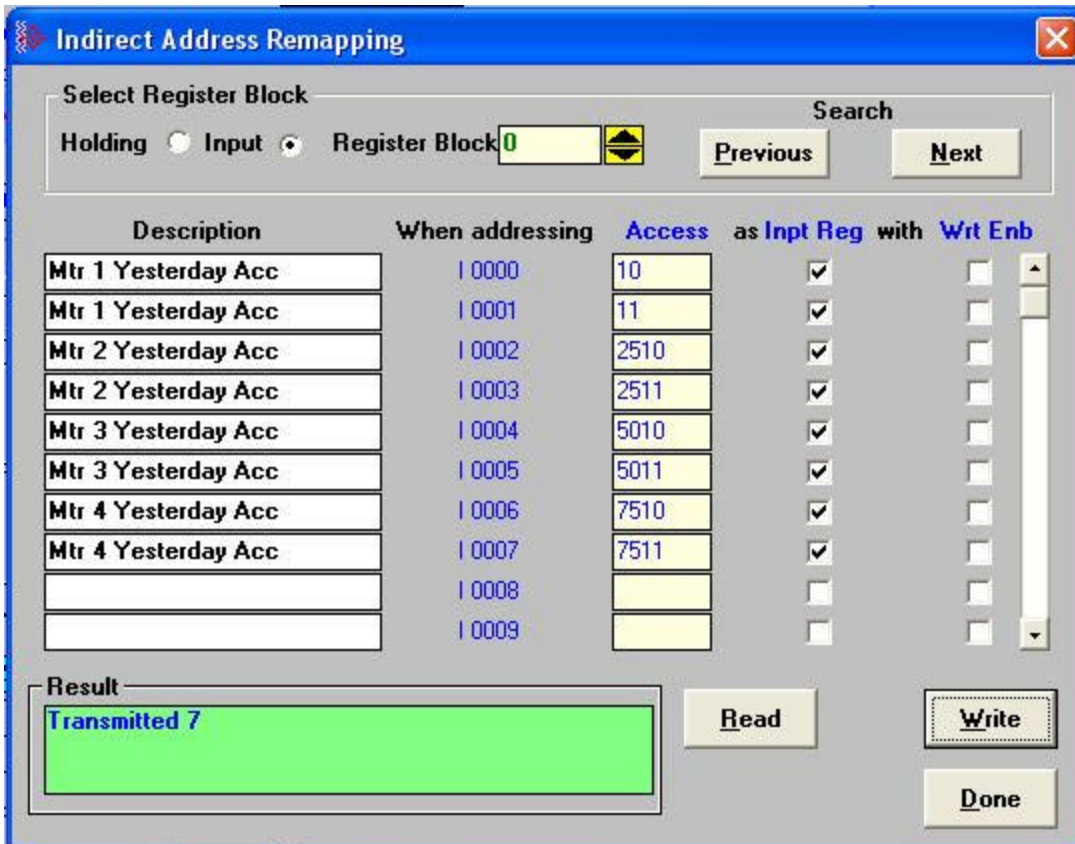
Start Offset	End Offset	Data Type	Description
10	11	Acc	Accumulator totalizer, net/liqeqv
12	13	Fp	Accumulator residue, net/liqeqv
14	15	Fp	Flow rate, net/liqeqv
16	17	Fp	Temperature, CU
18	19	Fp	Pressure, CUg
20	21	Fp	K-Factor
22	23	Fp	Meter Factor
24	25	Fp	Flowing density, CU
26		Wd	Water content, % e-2
27		Wd	Corrected density, kg/m <sup>3</sup> e-1
28		Wd	Ctl e-4
29		Wd	Cpl e-4

So the user is interested in the following input registers:

Primary Modbus Slave Input Register Address	Description
10	Net Acc Totalizer from Yesterday archive – Meter 1
11	Net Acc Totalizer from Yesterday archive – Meter 1
2510	Net Acc Totalizer from Yesterday archive – Meter 2
2511	Net Acc Totalizer from Yesterday archive – Meter 2
5010	Net Acc Totalizer from Yesterday archive – Meter 3
5011	Net Acc Totalizer from Yesterday archive – Meter 3
7510	Net Acc Totalizer from Yesterday archive – Meter 4
7511	Net Acc Totalizer from Yesterday archive – Meter 4

5) Using the AFC Manager the user should remmap the registers above to the Virtual Slave. In this example we will remmap the addresses as follows:

Virtual Modbus Slave Input Register Address	Primary Modbus Slave Input Register Address
0	10
1	11
2	2510
3	2511
4	5010
5	5011
6	7510
7	7511



**Indirect Address Remapping**

Select Register Block  
 Holding  Input  Register Block 0   Search

Description	When addressing	Access	as Inpt Reg	with Wrt Enb
Mtr 1 Yesterday Acc	I 0000	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Mtr 1 Yesterday Acc	I 0001	11	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Mtr 2 Yesterday Acc	I 0002	2510	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Mtr 2 Yesterday Acc	I 0003	2511	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Mtr 3 Yesterday Acc	I 0004	5010	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Mtr 3 Yesterday Acc	I 0005	5011	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Mtr 4 Yesterday Acc	I 0006	7510	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Mtr 4 Yesterday Acc	I 0007	7511	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	I 0008		<input type="checkbox"/>	<input type="checkbox"/>
	I 0009		<input type="checkbox"/>	<input type="checkbox"/>

Result  
 Transmitted 7

Obs: the past archives and events are always input registers. All other data should be remapped as holding registers.

6) In the sample ladder we can now configure the Modbus Gateway block to read 6 input registers words from the Virtual slave starting at address 0:

- AFC56.Modbus.Gateway[0].Config	{...}
- AFC56.Modbus.Gateway[0].Config.Enable	1
+ AFC56.Modbus.Gateway[0].Config.StartRegister	0
+ AFC56.Modbus.Gateway[0].Config.RegisterCount	8
- AFC56.Modbus.Gateway[0].Config.FunctionType_Write	0
- AFC56.Modbus.Gateway[0].Config.RegisterType_Input	1
- AFC56.Modbus.Gateway[0].Config.SlaveType_Virtual	1

7) The data will be available in the *AFC56.Modbus.Gateway[0].ReadData[0]* controller tag.

The following screenshots shows that, for this example, the totalizer values are:

	<b>Totalizer for yesterday's archive</b>
<b>Meter 1</b>	5
<b>Meter 2</b>	7
<b>Meter 3</b>	11
<b>Meter 4</b>	14

Meter 1 – Yesterday Archive

**Meter Archive** ✖

Site Name

Project

Meter Tag

**Select Meter**

Meter  1

---

**Select Archives**

Daily  
 Hourly

**Result**

Success

<b>86308</b>	Flowing period	<b>2003-09-03 00:00:00</b>	Closing timestamp of archive
<b>152</b>	Event counter	<b>2003-09-02 00:01:32</b>	Opening timestamp of archive
<b>205.7708</b>	Net flow rate (x m3/h)	<b>00h</b>	Status (bits 8-15)
<b>35.97999</b>	Temperature (°C)	<b>0000h</b>	Alarms
<b>53.23013</b>	Pressure (kPag)	<b>5</b>	Net accumulator (x m3)
<b>0.6</b>	Relative density (15°C/15°C)	<b>0.3294477</b>	Accumulator residue
<b>1</b>	Reference compressibility	<b>22.58004</b>	Differential Pressure (kPa)
<b>1</b>	Flowing compressibility	<b>1.0008</b>	Velocity of approach factor Ev
<b>1</b>	Fpv	<b>0.9603</b>	Expansion factor Y
		<b>0.598</b>	Coefficient of discharge Cd

Meter 2 – Yesterday Archive

**Meter Archive**

Site Name: **MVI Flow Station** Project: **AFC**

Meter Tag: **M02**

Select Meter: Meter **2**

Select Archives: **1**  Daily  Hourly

**Read** **Update**

Result: **Success**

<b>86400</b>	Flowing period	<b>2003-09-03 00:00:00</b>	Closing timestamp of archive
<b>152</b>	Event counter	<b>2003-09-02 00:00:00</b>	Opening timestamp of archive
<b>140.4606</b>	Net flow rate (x m3/h)	<b>00h</b>	Status (bits 8-15)
<b>13.09995</b>	Temperature (°C)	<b>0000h</b>	Alarms
<b>13.20002</b>	Pressure (kPag)	<b>7</b>	Net accumulator (x m3)
<b>0.6</b>	Relative density (15°C/15°C)	<b>0.2659223</b>	Accumulator residue
<b>1</b>	Reference compressibility	<b>13.29996</b>	Differential Pressure (kPa)
<b>1</b>	Flowing compressibility	<b>1.0008</b>	Velocity of approach factor Ev
<b>1</b>	Fpv	<b>0.9676</b>	Expansion factor Y
		<b>0.5985</b>	Coefficient of discharge Cd

**Close** **Print** **Log**

Meter 3 – Yesterday Archive

**Meter Archive**

Site Name: **MVI Flow Station** Project: **AFC**

Meter Tag: **M03**

Select Meter: Meter **3**

Select Archives: **1**  Daily  Hourly

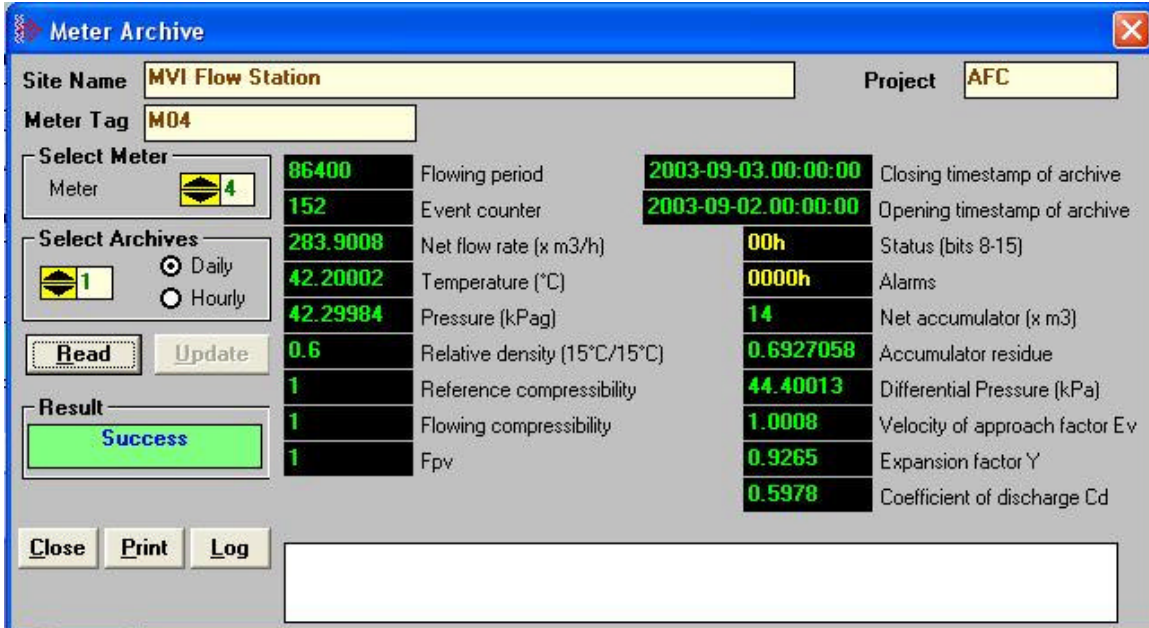
**Read** **Update**

Result: **Success**

<b>86400</b>	Flowing period	<b>2003-09-03 00:00:00</b>	Closing timestamp of archive
<b>152</b>	Event counter	<b>2003-09-02 00:00:00</b>	Opening timestamp of archive
<b>229.9332</b>	Net flow rate (x m3/h)	<b>00h</b>	Status (bits 8-15)
<b>31.09979</b>	Temperature (°C)	<b>0000h</b>	Alarms
<b>31.19998</b>	Pressure (kPag)	<b>11</b>	Net accumulator (x m3)
<b>0.6</b>	Relative density (15°C/15°C)	<b>0.8533873</b>	Accumulator residue
<b>1</b>	Reference compressibility	<b>31.29995</b>	Differential Pressure (kPa)
<b>1</b>	Flowing compressibility	<b>1.0008</b>	Velocity of approach factor Ev
<b>1</b>	Fpv	<b>0.9406</b>	Expansion factor Y
		<b>0.5979</b>	Coefficient of discharge Cd

**Close** **Print** **Log**

Meter 4 - Yesterday Archive



**Meter Archive**

Site Name: **MVI Flow Station** Project: **AFC**

Meter Tag: **M04**

Select Meter: Meter **4**

Select Archives: **1**  Daily  Hourly

Buttons: **Read** **Update**

Result: **Success**

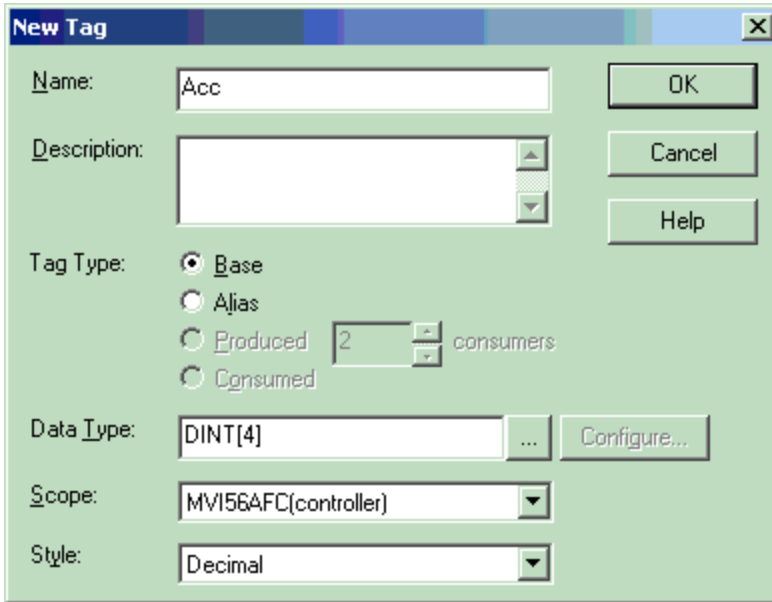
<b>86400</b>	Flowing period	<b>2003-09-03 00:00:00</b>	Closing timestamp of archive
<b>152</b>	Event counter	<b>2003-09-02 00:00:00</b>	Opening timestamp of archive
<b>283.9008</b>	Net flow rate (x m3/h)	<b>00h</b>	Status (bits 8-15)
<b>42.20002</b>	Temperature (°C)	<b>0000h</b>	Alarms
<b>42.29984</b>	Pressure (kPag)	<b>14</b>	Net accumulator (x m3)
<b>0.6</b>	Relative density (15°C/15°C)	<b>0.6927058</b>	Accumulator residue
<b>1</b>	Reference compressibility	<b>44.40013</b>	Differential Pressure (kPa)
<b>1</b>	Flowing compressibility	<b>1.0008</b>	Velocity of approach factor Ev
<b>1</b>	Fpv	<b>0.9265</b>	Expansion factor Y
		<b>0.5978</b>	Coefficient of discharge Cd

Buttons: **Close** **Print** **Log**

The controller tags show:

- AFC56.Modbus.Gateway[0].ReadData	{ ... }
+ AFC56.Modbus.Gateway[0].ReadData[0]	5
+ AFC56.Modbus.Gateway[0].ReadData[1]	0
+ AFC56.Modbus.Gateway[0].ReadData[2]	7
+ AFC56.Modbus.Gateway[0].ReadData[3]	0
+ AFC56.Modbus.Gateway[0].ReadData[4]	11
+ AFC56.Modbus.Gateway[0].ReadData[5]	0
+ AFC56.Modbus.Gateway[0].ReadData[6]	14
+ AFC56.Modbus.Gateway[0].ReadData[7]	0

The ReadData[] tag is an array of integers. Since the totalizer is displayed as 32-bit long integer, the user may create an array of DINT elements and then add a rung to copy the values to this new array.



**New Tag**

Name:

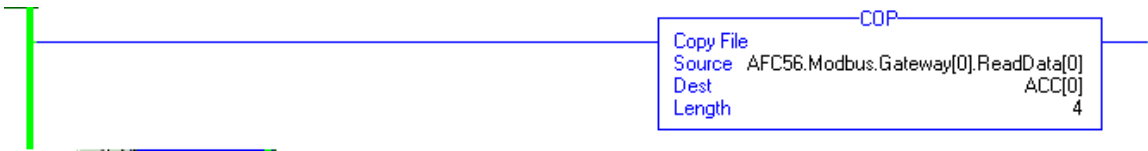
Description:

Tag Type:  Base  
 Alias  
 Produced  consumers  
 Consumed

Data Type:  ...

Scope:

Style:



The totalizers will be copied to the new ACC[] array:

[-] ACC	{ ... }
[+] ACC[0]	5
[+] ACC[1]	7
[+] ACC[2]	11
[+] ACC[3]	14

## APPENDIX B – HOW TO CONFIGURE ARCHIVES TO RESET ACCUMULATORS

The module can be configured to automatically reset the resettable accumulators upon period end using Meter-Relative words Mh00341 (daily archive) and Mh00421 (hourly archive) in the Primary Modbus Slave. In order to use this feature, no ladder logic is required.

Please refer to Appendix B for further information about this subject.

The Meter-Relative addresses refers to the offset within each meter range. Each meter uses 2000 holding register words (starting at address 8000):

Address	Description
8341	Meter 1 daily archive configuration word
8421	Meter 1 hourly archive configuration word
10341	Meter 2 daily archive configuration word
10421	Meter 2 hourly archive configuration word
12341	Meter 3 daily archive configuration word
12421	Meter 3 hourly archive configuration word
14341	Meter 4 daily archive configuration word
14421	Meter 4 hourly archive configuration word
16341	Meter 5 daily archive configuration word
16421	Meter 5 hourly archive configuration word
...	...

Each configuration word is a bitmap word which has the following meaning:

Bit	Description
0	Period select, hourly
1	Archive upon period end
2	Archive upon event
3	Reserved
4	Reset resettable accumulator 1 upon period end
5	Reset resettable accumulator 2 upon period end
6	Reset resettable accumulator 3 upon period end
7	Reset resettable accumulator 4 upon period end



**Corporate Office:**  
1675 Chester Avenue  
Fourth Floor  
Bakersfield, CA 93301

661/716-5100

Fax: 661/716-5101

e-mail: [prosoft@prosoft-technology.com](mailto:prosoft@prosoft-technology.com)

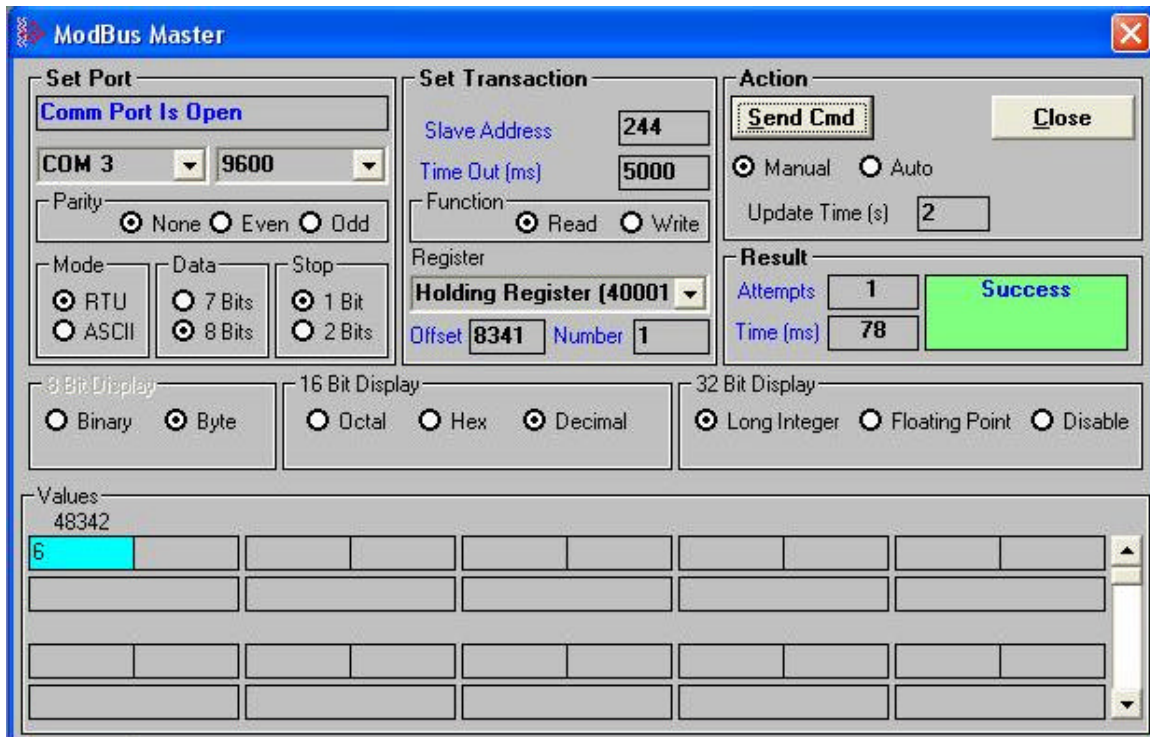
8	Reset resettable accumulator 1 upon event
9	Reset resettable accumulator 2 upon event
10	Reset resettable accumulator 3 upon event
11	Reset resettable accumulator 4 upon event
12	Reserved
13	Reserved
14	Reserved
15	Reserved

The default value for address Mh00341 is 6 and for Mh00421 is 7. It means that by default the archives are configured to be generated upon period end and if an event occurs.

**Example:** Configure the Meter 1 to reset all accumulators once the daily archive is generated.

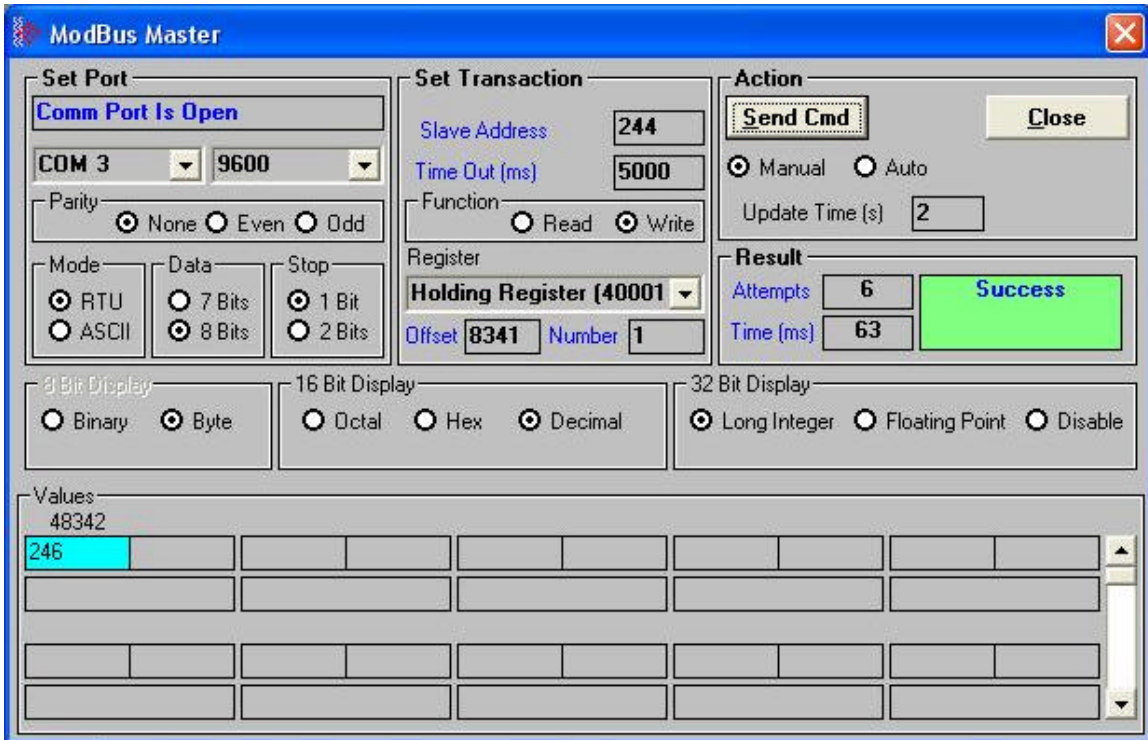
The user has to set bits 4 to 7 in the holding register 8341 from the Primary Modbus Slave. It means that the user has to write a value of 246 to address 80341.

The easiest way to accomplish this is referring to the AFC Manager->ModbusMaster interface and read the content of address 8341:



Set Port		Set Transaction		Action	
Comm Port Is Open		Slave Address	244	Send Cmd	Close
COM 3	9600	Time Out (ms)	5000	<input checked="" type="radio"/> Manual <input type="radio"/> Auto	Update Time (s) 2
Parity: <input checked="" type="radio"/> None <input type="radio"/> Even <input type="radio"/> Odd		Function: <input checked="" type="radio"/> Read <input type="radio"/> Write			
Mode: <input checked="" type="radio"/> RTU <input type="radio"/> ASCII		Register: Holding Register (40001)			
Data: <input type="radio"/> 7 Bits <input checked="" type="radio"/> 8 Bits	Stop: <input checked="" type="radio"/> 1 Bit <input type="radio"/> 2 Bits	Offset: 8341	Number: 1	Attempts: 1	Time (ms): 78
9 Bit Display: <input type="radio"/> Binary <input checked="" type="radio"/> Byte		16 Bit Display: <input type="radio"/> Octal <input type="radio"/> Hex <input checked="" type="radio"/> Decimal		32 Bit Display: <input checked="" type="radio"/> Long Integer <input type="radio"/> Floating Point <input type="radio"/> Disable	
Values					
48342					
6					

Change the value from 6 to 246 and also change the function type from *Read* to *Write*. Click the **Send Cmd** button to write the new value to the module.



The daily archive is now configured. The resettable accumulators will be reset everytime the archive period ends.

The user should now perform an Upload operation in order to refresh the AFC configuration file in the local PC with the new values.