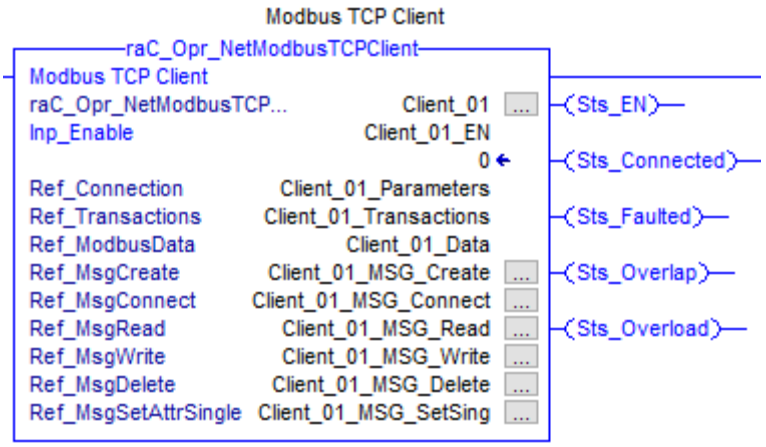
Modbus TCP Client Add-On Instruction based code for ControlLogix® and CompactLogix® controllers

Introduction

This document describes the application and use of the Modbus TCP Client Add-On Instruction.

Modbus TCP Client Add-On Instruction (AOI) allows users to implement Modbus TCP Client functionality into the Logix family of controllers. AOIs can be used standalone or can be added to an existing application following the directions outlined below.



Contents

Introduction 1

Requirements 3

- Hardware Requirements 3
- Software Requirements 3
- Memory Requirements 3

Functional Requirements and Description 4

- Supported Modbus Function Codes 4
  - Bit Level Commands 4
  - Word Level Commands 5

Implementation 6

- Modbus TCP Client AOI implementation 6
  - Using Periodic Task 6
  - Rung Import and tag naming changes 6
  - Rung Import process for Modbus TCP Client AOI 6
  - Configure Local Operational Parameters 9
  - Configure Data Transactions 11
- Implementation Restrictions 15

Monitoring Modbus TCP Server operations 16

Performance data 18

Revision History 19

Requirements

Hardware Requirements

The Modbus TCP Client code requires a ControlLogix or CompactLogix controller with an EtherNet/IP module that supports Logix Sockets functionality. See Knowledgebase technote 470690 for complete list of controllers and modules.
https://rockwellautomation.custhelp.com/app/answers/detail/a_id/470690

Software Requirements

The Modbus TCP Client AOI code supports Logix controller revisions 20 and higher.

Memory Requirements

First instance of the Modbus TCP Client AOI uses about 93 Kbytes of memory.
Each additional AOI instance requires about 20 Kbytes of memory.
These estimates based on the ControlLogix 5570 family of controllers.
Please note some Compactlogix controllers have a starting memory size as low as 384Kbytes. This code can take a significant amount of memory in smaller CompactLogix controllers.

Functional Requirements and Description

Supported Modbus Function Codes

Bit Level Commands

Function Code	Name	Description	Supported Values	Modbus Range
01	Read Coils	This function code is used to read contiguous status of coils in a remote device (0xxxx addresses). The coils in the response message are packed as one coil per bit of the data field.	Local Address: 0 to 1023 Server Address: 0 to 9999 Length: 1 to 256 coils	Local Address 00001-01024 Server Address 00001-09999
02	Read Discrete Inputs	This function code is used to read contiguous status of discrete inputs in a remote device (1xxxx addresses). The inputs in the response message are packed as one coil per bit of the data field.	Local Address: 0 to 1023 Server Address: 0 to 9999 Length: 1 to 256 Inputs	Local Address 10001-11024 Server Address 10001-19999
05	Write Single Coil	This function code is used to write a single coil to either ON or OFF in a remote device (0xxxx addresses).	Local Address: 0 to 1023 Server Address: 0 to 9999	Local Address 00001-01024 Server Address 00001-09999
15	Write Multiple Coils	This function code is used to write to one or more coils in a sequence of coils to either ON or OFF in a remote device (0xxxx addresses).	Local Address: 0 to 1023 Server Address: 0 to 9999 Length: 1 to 256 coils	Local Address 00001-01024 Server Address 00001-09999

Word Level Commands

Function Code	Name	Description	Supported Values	Modbus Range
03	Read Holding Registers	This function code is used to read the contents of a contiguous block of holding registers (4xxxx addresses) in a remote device.	Local Address: 0 to 1023 Server Address: 0 to 9999 Length: 1 to 120 registers	Local Address 40001-41024 Server Address 40001-49999
04	Read Input Registers	This function code is used to read the contents of a contiguous block of input registers (3xxxx addresses) in a remote device.	Local Address: 0 to 1023 Server Address: 0 to 9999 Length: 1 to 120 input registers	Local Address 30001-31024 Server Address 30001-39999
06	Write a Single Holding Register	This function code is used to write to single holding register (4xxxx addresses) in a remote device	Local Address: 0 to 1023 Server Address: 0 to 9999	Local Address 40001-41024 Server Address 40001-49999
16	Write Multiple Holding Registers	This function code is used to write to contiguous holding registers (4xxxx addresses) in a remote device.	Local Address: 0 to 1023 Server Address: 0 to 9999 Length: 1 to 120 registers	Local Address 40001-41024 Server Address 40001-49999

Implementation

Modbus TCP Client AOI implementation

Using Periodic Task

It's recommended to add AOIs into a Periodic task with Rate of 10ms (or higher).
Slower rates will reduce controller load and reduce performance.
Faster task rates will increase performance but will add a significant load to the controller.

See Performance Data section for details.

Rung Import and tag naming changes

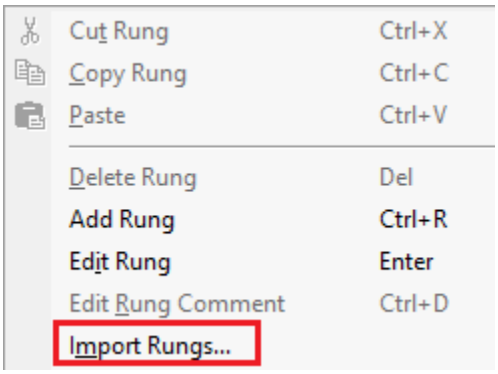
The pre-configured Add-On Instructions are supplied in a Rung format.

The Rung Import format must be used to implement the AOI.

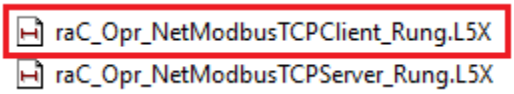
***Important**:*
Use only the Rung Import process.
Do not use Copy/Paste functionality or add these AOIs using Instructions tool bar. Doing this will remove configurations from pre-configured message instructions, making AOIs non-functional.

Rung Import process for Modbus TCP Client AOI

1. Open a Ladder Routine within your application
2. Right click on any empty area and select ***Import Rungs***



3. Select ***raC_Opr_NetModbusTCPClient_Rung.L5X*** file and click ***Import***



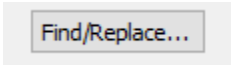
4. When Import Configuration Dialog opens, select ***Tags***

5. You can leave final names as-is or change them to accommodate your application.

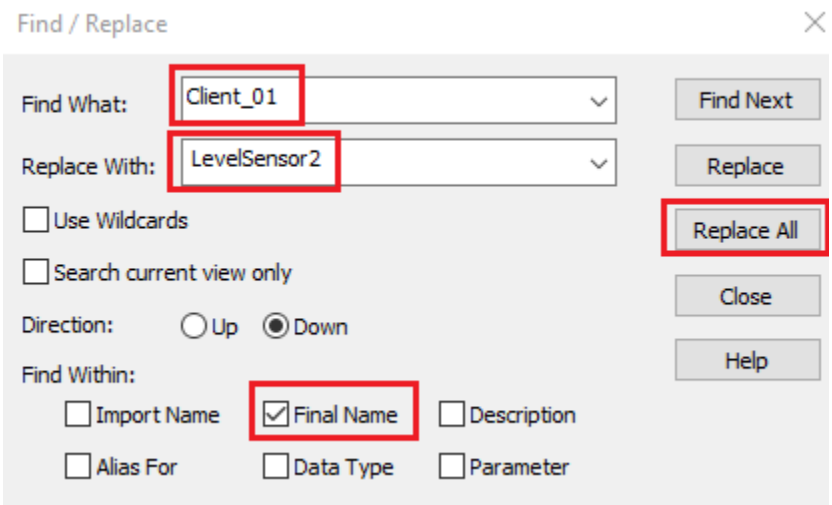
Configure Tag References

Import Name	Operation	Final Name
Client_01	Create	Client_01
Client_01_Data	Create	Client_01_Data
Client_01_EN	Create	Client_01_EN
Client_01_MSG_Connect	Create	Client_01_MSG_Connect
Client_01_MSG_Create	Create	Client_01_MSG_Create
Client_01_MSG_Delete	Create	Client_01_MSG_Delete
Client_01_MSG_Read	Create	Client_01_MSG_Read
Client_01_MSG_SetSing	Create	Client_01_MSG_SetSing
Client_01_MSG_Write	Create	Client_01_MSG_Write
Client_01_Parameters	Create	Client_01_Parameters
Client_01_Transactions	Create	Client_01_Transactions

6. To change Final Names click ***Find/Replace*** button



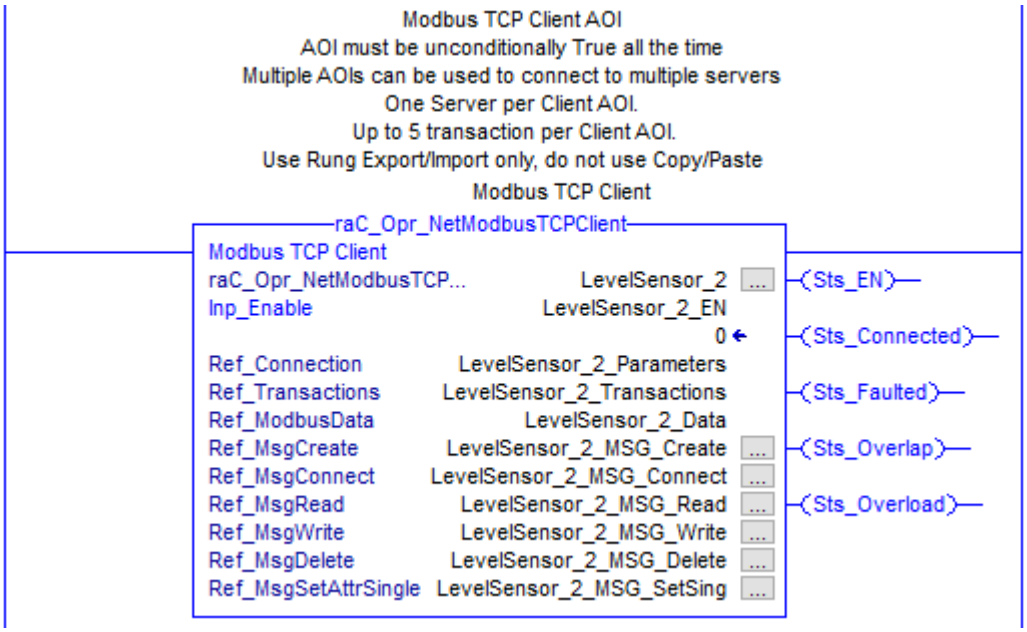
When Dialog opens, replace default name **Client_01** with desired prefix, verify that Final Names Box is checked then click ***Replace All***



Close Find/Replace dialog and verify **Final Names**

Final Name
LevelSensor2
LevelSensor2_Data
LevelSensor2_EN
LevelSensor2_MSG_Connect
LevelSensor2_MSG_Create
LevelSensor2_MSG_Delete
LevelSensor2_MSG_Read
LevelSensor2_MSG_SetSing
LevelSensor2_MSG_Write
LevelSensor2_Parameters
LevelSensor2_Transactions

7. Click ***Ok*** to finish the Import process
   The new rung should look like shown below without any errors



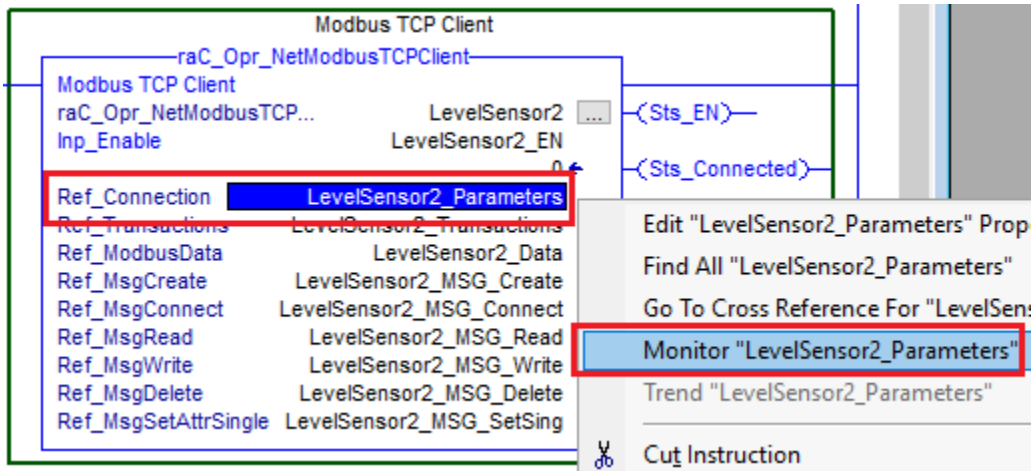
8. Repeat Steps 2-7 if application requires additional Clients. DO NOT Copy/Paste.

Configure Local Operational Parameters

Modbus TCP Client requires a local EtherNet/IP module that supports Logix Sockets. See Requirements section for details.

In this section we will link Modbus TCP Client AOI to this EtherNet/IP module.

1. Right Click on the tag attached to the **Ref_Connection** parameter and select ***Monitor "…"***



2. Expand Parameters tag. Specify the slot of the Local EtherNet/IP module.

LevelSensor2_Parameters	raC_UDT_Modbus...	{...}
LevelSensor2_Parameters.LocalSlot	SINT	0
LevelSensor2_Parameters.LocalAddress	STR0016	''
LevelSensor2_Parameters.DestAddress	STR0016	''
LevelSensor2_Parameters.DestinationPort	DINT	502

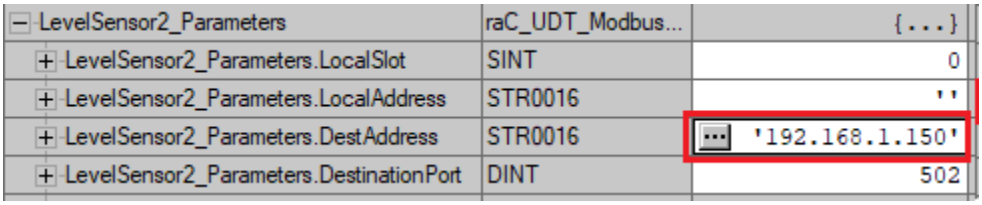
For 1756 ControlLogix processors specify the actual slot of desired 1756-EN2T(R) module.
For 1756-L8xE controllers using the built in Ethernet port specify the 1756-L8xE controller slot.
For CompactLogix 5370, 5380, 5480 controllers leave **.LocalSlot** at 0.

3. Specify the **.LocalAddress** of the EtherNet/IP module.

LevelSensor2_Parameters	raC_UDT_Modbus...	{...}
LevelSensor2_Parameters.LocalSlot	SINT	0
LevelSensor2_Parameters.LocalAddress	STR0016	''
LevelSensor2_Parameters.DestAddress	STR0016	''
LevelSensor2_Parameters.DestinationPort	DINT	502

For CompactLogix 5380 and 5480 in **Dual IP Mode only**, specify the IP Address of the Local Ethernet connection used for Modbus TCP communications. **Leave this field blank for all other cases.**

4. Specify Ethernet IP Address of the Modbus Server device. This address must be specified and cannot remain blank.



5. Leave Default Modbus TCP port at 502. This value is Modbus TCP protocol standard.
6. If you change any of these Parameters during operation be sure to reset and then set the AOI **Inp_Enable** parameter tag.

Configure Data Transactions

1. Expand Transactions tags to expose Transaction 0 member tags

LevelSensor2_Transactions	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[0]	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[0].Enabled	BOOL	0
LevelSensor2_Transactions[0].PollInterval	DINT	1000
LevelSensor2_Transactions[0].TransType	INT	3
LevelSensor2_Transactions[0].UID	SINT	0
LevelSensor2_Transactions[0].BeginAddress	INT	1
LevelSensor2_Transactions[0].Count	INT	100
LevelSensor2_Transactions[0].LocalAddress	INT	6
LevelSensor2_Transactions[0].TransComplete	BOOL	1
LevelSensor2_Transactions[0].TransStat	INT	0
LevelSensor2_Transactions[0].Diagnostic	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[1]	raC_UDT_ModbusCl...	{...}

2. Set Polling Interval value in milliseconds.
   Default value is 1000 (1 second).
   Minimum value is 80 msec. Any transaction with polling rate below 80ms will be polled at 1 second rate.

LevelSensor2_Transactions	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[0]	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[0].Enabled	BOOL	0
LevelSensor2_Transactions[0].PollInterval	DINT	1000
LevelSensor2_Transactions[0].TransType	INT	3
LevelSensor2_Transactions[0].UID	SINT	0
LevelSensor2_Transactions[0].BeginAddress	INT	1
LevelSensor2_Transactions[0].Count	INT	100
LevelSensor2_Transactions[0].LocalAddress	INT	6
LevelSensor2_Transactions[0].TransComplete	BOOL	1
LevelSensor2_Transactions[0].TransStat	INT	0
LevelSensor2_Transactions[0].Diagnostic	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[1]	raC_UDT_ModbusCl...	{...}

3. Set Modbus Function code in **TransType** tag. See Supported Modbus Function Codes section for the list of supported commands.

LevelSensor2_Transactions	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[0]	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[0].Enabled	BOOL	0
LevelSensor2_Transactions[0].PollInterval	DINT	1000
LevelSensor2_Transactions[0].TransType	INT	3
LevelSensor2_Transactions[0].UID	SINT	0
LevelSensor2_Transactions[0].BeginAddress	INT	1
LevelSensor2_Transactions[0].Count	INT	100
LevelSensor2_Transactions[0].LocalAddress	INT	6
LevelSensor2_Transactions[0].TransComplete	BOOL	1
LevelSensor2_Transactions[0].TransStat	INT	0
LevelSensor2_Transactions[0].Diagnostic	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[1]	raC_UDT_ModbusCl...	{...}

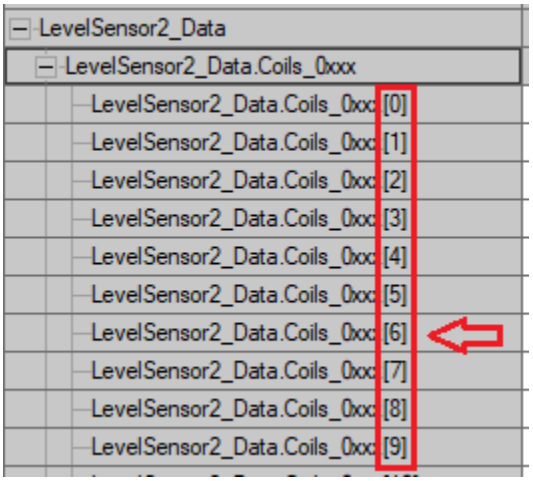
4. Set **BeginAddress** tag. The value represents the beginning address in remote device (Modbus TCP Server) to read from or write to.
   Depending on the function used above, values 0…9998 represent Modbus addresses 00001-09999, 10001-19999, 30001-39999, 40001-49999 respectively.

LevelSensor2_Transactions	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[0]	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[0].Enabled	BOOL	0
LevelSensor2_Transactions[0].PollInterval	DINT	1000
LevelSensor2_Transactions[0].TransType	INT	3
LevelSensor2_Transactions[0].UID	SINT	0
LevelSensor2_Transactions[0].BeginAddress	INT	1
LevelSensor2_Transactions[0].Count	INT	100
LevelSensor2_Transactions[0].LocalAddress	INT	6
LevelSensor2_Transactions[0].TransComplete	BOOL	1
LevelSensor2_Transactions[0].TransStat	INT	0
LevelSensor2_Transactions[0].Diagnostic	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[1]	raC_UDT_ModbusCl...	{...}

5. Set **Count** tag. The value represents the number of items in remote device (Modbus TCP Server) to read from or write to.

LevelSensor2_Transactions	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[0]	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[0].Enabled	BOOL	0
LevelSensor2_Transactions[0].PollInterval	DINT	1000
LevelSensor2_Transactions[0].TransType	INT	3
LevelSensor2_Transactions[0].UID	SINT	0
LevelSensor2_Transactions[0].BeginAddress	INT	1
LevelSensor2_Transactions[0].Count	INT	100
LevelSensor2_Transactions[0].LocalAddress	INT	6
LevelSensor2_Transactions[0].TransComplete	BOOL	1
LevelSensor2_Transactions[0].TransStat	INT	0
LevelSensor2_Transactions[0].Diagnostic	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[1]	raC_UDT_ModbusCl...	{...}

6. Set **LocalAddress** tag. The value represents the beginning address in this programs' "_Data" arrays.



LevelSensor2_Transactions	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[0]	raC_UDT_ModbusCl...	{...}
LevelSensor2_Transactions[0].Enabled	BOOL	0
LevelSensor2_Transactions[0].PollInterval	DINT	1000
LevelSensor2_Transactions[0].TransType	INT	3
LevelSensor2_Transactions[0].UID	SINT	0
LevelSensor2_Transactions[0].BeginAddress	INT	1
LevelSensor2_Transactions[0].Count	INT	100
LevelSensor2_Transactions[0].LocalAddress	INT	6
LevelSensor2_Transactions[0].TransComplete	BOOL	1
LevelSensor2_Transactions[0].TransStat	INT	0

7. Start Modbus TCP Client by setting tag attached to **Inp_Enable** parameter to 1.

Implementation Restrictions

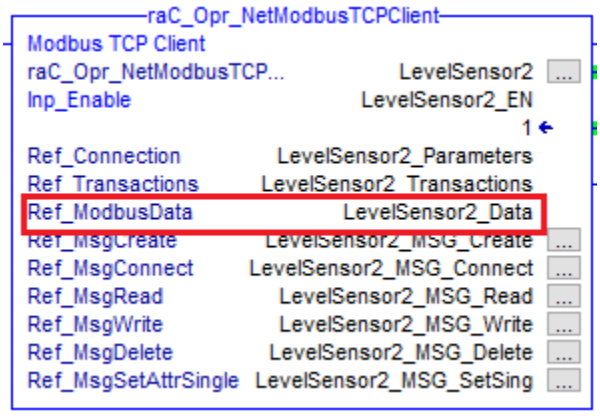
1. Implementation must be done using Import Rung function only to preserve Message instruction configurations. Do not use Copy/Paste as it will not bring complete Message instruction configurations and tags. Do not use Search/Replace tags once rung is implemented. All replacement can be done only during rung import.

2. Multiple instances of Client AOI are supported per controller.
   Each instance must use own set of backing and Message tags, however **"…_Data"** tag structure can be shared between AOIs.

3. Modbus TCP Server and Modbus TCP Client AOIs can reside in the same program. However Server applications may cause a temporary Client disconnection due to the shared Logix Sockets object.

Monitoring Modbus TCP Server operations

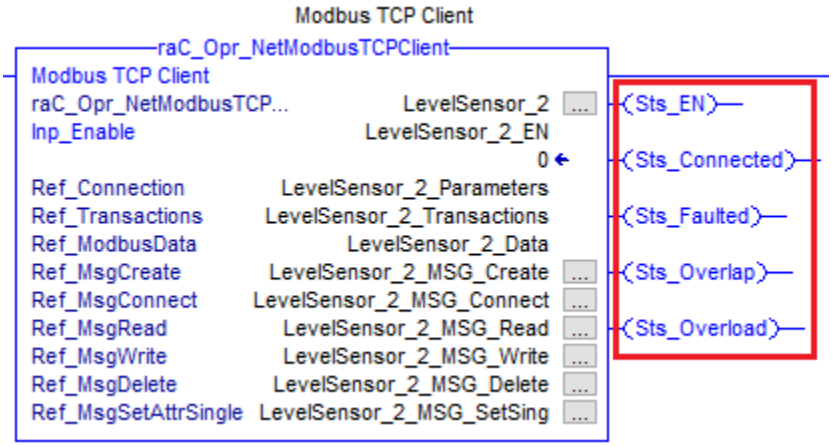
1. Modbus Tags are located under **Ref_ModbusData** parameter tag.



This tag contains four separate data areas for coils (0xxxx), discrete inputs (1xxxx), input registers (3xxxx) and holding registers (4xxxx). These tag values can be read and populated by the user application without any restrictions.

LevelSensor_2_Data	raC_UDT_Mod...
LevelSensor_2_Data.Coils_0xxx	BOOL[1024]
LevelSensor_2_Data.DiscInputs_1xxx	BOOL[1024]
LevelSensor_2_Data.InpRegisters_3xxx	INT[1024]
LevelSensor_2_Data.HoldRegisters_4xxx	INT[1024]

2. Modbus TCP AOI Status Bits

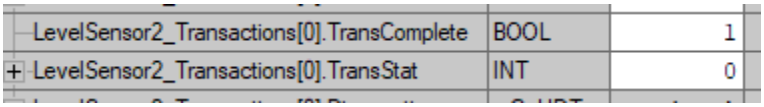


   a. **Sts_EN** output indicates that Modbus TCP Client functionality is enabled.

b. **Sts_Connected** output indicates that Client connection request was accepted by the Server. It does not indicate the active data flow. The status of individual transactions should be checked to verify the data exchange.
c. **Sts_Faulted** output indicates that one of the message instructions is faulted.
d. **Sts_Overlap** output indicates that one or more transaction is not completing before next trigger
e. **Sts_Overload** output indicates excessive Overlaps in one or more transactions.

3. Individual Transaction status information. These tags exist for each of the 5 built in transactions. Transaction 0-4).

   TransComplete bit is set when requested transaction was completed. It's cleared by the program when next transaction is requested.
   TransStatus value indicates the current status of transaction.
   0 = success, 1 = in process, 2 = retry, -1 = exception



LevelSensor2_Transactions[0].TransComplete	BOOL	1
+ LevelSensor2_Transactions[0].TransStat	INT	0

4. Transaction Diagnostic data structure provides internal dynamic information while the transaction is active. **Do not write to these tags.**

− LevelSensor_2_Transactions[0].Diagnostic	raC_UDT_Mod...	{...}
+ LevelSensor_2_Transactions[0].Diagnostic.Request	STR0462	'$0F$A8$00$00$0...
LevelSensor_2_Transactions[0].Diagnostic.ReqBuilt	BOOL	0
LevelSensor_2_Transactions[0].Diagnostic.Overlap	BOOL	0
LevelSensor_2_Transactions[0].Diagnostic.Overload	BOOL	0
+ LevelSensor_2_Transactions[0].Diagnostic.TransID	INT	4008
+ LevelSensor_2_Transactions[0].Diagnostic.TransLastError	INT	0
+ LevelSensor_2_Transactions[0].Diagnostic.IntervalTimer	TIMER	{...}
+ LevelSensor_2_Transactions[0].Diagnostic.NoReplyCounter	COUNTER	{...}
+ LevelSensor_2_Transactions[0].Diagnostic.PendingTimer	TIMER	{...}

Performance data

Modbus Client performance can be affected by many factors including: periodic task rate, performance of the Server device, speed of Client controller, how busy the Client controller is, network performance, network card, number of Clients in the controller, the number of active transactions etc.

Below are typical performance expectations when using an L7x Controller with a **10ms** periodic task:

Number of Transaction Enabled	**Recommended Minimum *PollInterval* for all Transactions**
1	80 mS
2	130 mS
3	220 mS
4	300 mS
5	380 mS

When using the recommended **PollInterval** settings or greater, you can expect the actual data delivery to be very close to the **PollInterval**.

NOTE:
If selected **PollInterval** settings are marginally too fast you will likely see occasional **Sts_Overlap** errors and your system should work reasonably well.
If selected **PollInterval** settings are extremely out of line, you will see both **Sts_Overlap** and **Sts_Overload** errors and your system will not work reliably. If you are getting **Sts_Overload** errors, you must make adjustments to your **PollInterval** settings.

Revision History

1. Revision 1.02 – Initial Release in Ladder Program format. If you are currently using this code in an existing application, you may continue to do so.

2. Client Revision 2.00.00 – Initial Release in Add-On Instruction format. This version is recommended for use in all new applications.
   - 2.1. Enhancements
     - 2.1.1. Re-written code in Add-On instruction format
     - 2.1.2. Reduced memory requirements
     - 2.1.3. Simplified implementation and configuration
   - 2.2. Corrected Anomalies
     - 2.2.1. None
   - 2.3. Known Anomalies
     - 2.3.1.None

3. Client Revision 2.00.01 - Update
   - 3.1. Enhancements
     - 3.1.1. None
   - 3.2. Corrected Anomalies
     - 3.2.1. Minor logic correction related to the local IP Address
   - 3.3. Known Anomalies
     - 3.3.1.None