Application Note

C007 Modbus TCP Master Library

HA502487C007 Issue B

AC30 V1.13 onwards

AC30P/D V2.13, V3.13, V4.16 onwards

Library V1.1.4.1 onwards

Safety Information

Requirements

Intended Users

This Application Note is to be made available to all persons who are required to install, configure or service equipment described herein, or any other associated operation.

The information given is intended to enable the user to obtain maximum benefit from the equipment.

Application Area

The equipment described is intended for industrial motor speed control utilising AC induction or AC synchronous machines.

Personnel

Installation, operation and maintenance of the equipment should be carried out by qualified personnel. A qualified person is someone who is technically competent and familiar with all safety information and established safety practices; with the installation process, operation and maintenance of this equipment; and with all the hazards involved.

Hazards

Refer to the Safety Information given at the front of the Product Manual supplied with every Parker SSD Drives product.

C007 MODBUS TCP MASTER LIBRARY

Abstract

This application note explains how to use the **AC30 Modbus TCP Master** CoDeSys library on an AC30 drive.

Pre-Requisite

The pre-requisites are an AC30 drive and the PC tool PDQ or PDD.

Introduction

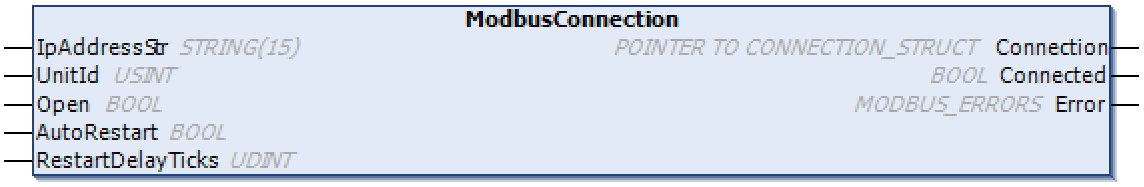
The CoDeSys **AC30 Modbus TCP Master** library allows the AC30 drive to be a Modbus master (client) to a connected Modbus slave (server) on the Ethernet network.   The slave, for example, could be additional IO or another AC30.

Two function blocks are available: **ModbusConnection** and **ModbusRequest**.

The function block **ModbusConnection** allows for a TCP connection to be made to a Modbus TCP slave.

The function block **ModbusRequest** allows for Modbus requests to be made to the connected slave and then processes the response. One or more ModbusRequest function blocks may be associated with a ModbusConnection function block.

Modbus Connection Function Block Description

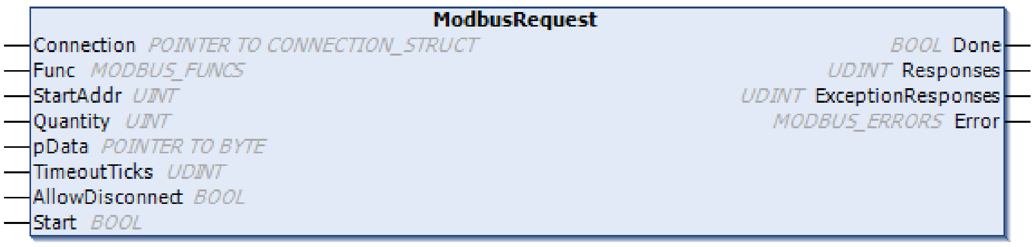


Inputs	**Default**	**Description**
IPAddressStr	‘0.0.0.0’	IP address of the Modbus TCP slave with which to connect.
UnitId	0	Unit ID with which to identify a remote slave. Normally this can be left as zero.
Open	FALSE	If TRUE, the master we attempt to connect to the Modbus slave. Subsequently set to FALSE to disconnect. Note that if the connection is lost, the input **Open** needs to be first set to FALSE before setting back to TRUE, unless the input **AutoRestart** is set to TRUE.
AutoRestart	FALSE	If TRUE, and the input **Open** is also TRUE, the master will automatically attempt to reconnect to the slave if the connection was lost, without having to set the input **Open** to FALSE first.
RestartDelayTicks	0	Number of task ticks to delay before attempting to automatically reconnect to the slave, when the input **AutoRestart** is TRUE.

Outputs	**Description**
Connection	Used to connect to one or more **ModbusRequest** function blocks.
Connected	TRUE indicates that a connection has been made to the slave.
Error	Enumerated error values related to this function block.

Value	**Enumerated Error**	**Description**
16#0000	ERROR_NONE	No errors detected.
16#0102	ERROR_NO_SOCKET	Could not create a socket due to lack of resources.
16#0103	ERROR_CONNECTION_REFUSED	Connection was refused by the slave.
16#0104	ERROR_CONNECTION_CLOSED	The slave has closed the connection.
16#0105	ERROR_CONNECTION_FAILED	General connection failure.

Modbus Request Function Block



Inputs	**Default**	**Description**
Connection	0	Connection back to a **ModbusConnection** function block.
Func	0	Enumerated Modbus function. See Mobus Functions section.
StartAddr	0	Modbus starting address. This assumes the starting addresses start at 1.
Quantity	0	Quantity of Modbus registers, coils or discrete inputs.
pData	0	Pointer to the Modbus read or write data, which may be a simple data type or a structure. A pointer can be created from the data using the **ADR** function. The size of the data pointed to **MUST** be large enough to hold all the Modbus data for the request or response.
TimeoutTicks	0	Number of task ticks to wait for the Modbus response before allowing another request. A value of 0 indicates an indefinite wait until the connection is closed.
AllowDisconnect	TRUE	If set to TRUE, the master will disconnect from the slave if a Modbus response has not been received within the period indicated by the input **TimeoutTicks**.
Start	FALSE	On a rising edge of this input, a Modbus request will be sent on the connection. This could be used with the **BLINK** function to produce a cyclic request-response. Note that the rate requests are sent should be limited to 10ms. A new request cannot be started until the previous request has completed (see **Done** below) or timed out.

Outputs	**Description**
Done	When TRUE this indicates a valid Modbus response has been received from the slave. This output will remain TRUE until the next rising edge of the input **Start**, or the connection is closed.
Responses	Number of valid Modbus responses received since the connection was made to the slave.
ExceptionResponses	Number of Modbus exception responses received since the connection was made to the slave.
Error	Enumerated error values of standard Modbus exception codes or errors related to this function block. The error will be cleared on the next rising edge of the input **Start**,

Value	**Enumerated Error**	**Description**
16#0000	ERROR_NONE	No errors detected.
16#0001 to 16#000B	See Modbus Exception Codes section	Modbus exceptions
16#0101	ERROR_PARAMETER	Invalid function block inputs: **pData** is set to zero **StartAddr** is set to zero **Quantity** is set to zero (excludes function codes 5 and 6) or has an invalid value for the Modbus function **Func** is set to an unsupported function
16#0105	ERROR_CONNECTION_FAILED	General connection failure.
16#0106	ERROR_RECEIVE_TIMEOUT	A Modbus response has not been received by the slave within the time specified by the input **TimeoutTicks**.

Modbus Functions

The following Modbus functions are supported with the Modbus Request function block.

Read Coils (01)

This function code is used to read from 1 to 2000 contiguous status of coils.

Required Input	**Description**	**Value**
Func	Function Code	MB_FUNC_READ_COILS (01)
StartAddr	Starting Address	16#0001 to 16#FFFF
Quantity	Quantity of Coils (N)	1 to 2000
pData	Pointer to data to which the response data will be placed.	The data **MUST** be at least N/8 (plus 1 if the remainder is different to 0) bytes in size.

Notes:

- The first least-significant 8 bits from the response will be placed in the first byte of the data pointed to. Thus in an AC30 application, if, for example, 16 coils are read into a UINT type of data then the byte order will need to be swapped.
- If the quantity of coils is not an even number of 8, then the remaining most-significant bits of the last byte will be filled with zeros.

Read Discrete Inputs (02)

The function code is used to read from 1 to 2000 contiguous status of discrete inputs.

Required Input	**Description**	**Value**
Func	Function Code	MB_FUNC_READ_DISCRETE_INPUTS (02)
StartAddr	Starting Address	16#0001 to 16#FFFF
Quantity	Quantity of Inputs (N)	1 to 2000
pData	Pointer to data to which the response data will be placed.	The **MUST** data be at least N/8 (plus 1 if the remainder is different to 0) bytes in size.

Notes:

- The first least-significant 8 bits from the response will be placed in the first byte of the data pointed to. Thus in an AC30 application, if, for example, 16 discrete inputs are read into a UINT type of data then the byte order will need to be swapped.
- If the quantity of inputs is not an even number of 8, then the remaining most-significant bits of the last byte will be filled with zeros.

Read Holding Registers (03)

The function code is used to read the contents of a contiguous block of holding registers.

Required Input	**Description**	**Value**
Func	Function Code	MB_FUNC_READ_HOLDING_REGS (03)
StartAddr	Starting Address	16#0001 to 16#FFFF
Quantity	Quantity of Registers (N)	1 to 125
pData	Pointer to data to which the response data will be placed.	The data **MUST** be at least N*2 bytes in size.

Read Input Registers (04)

The function code is used to read the contents of a contiguous block of input registers.

Required Input	Description	Value
Func	Function Code	MB_FUNC_READ_INPUT_REGS (04)
StartAddr	Starting Address	16#0001 to 16#FFFF
Quantity	Quantity of Input Registers (N)	1 to 125
pData	Pointer to data to which the response data will be placed.	The data **MUST** be at least N*2 bytes in size.

Write Single Coil (05)

The function code is used to write a single output coil to either ON or OFF.

Required Input	Description	Value
Func	Function Code	MB_FUNC_WRITE_SINGLE_COIL (05)
StartAddr	Output Address	16#0001 to 16#FFFF
pData	Pointer to the byte data that will set or reset the coil output.	The data **MUST** be 1 byte in size. A value of 0 represents OFF A value of 1 represents ON

Write Single Register (06)

The function code is used to write a single holding register.

Required Input	Description	Value
Func	Function Code	MB_FUNC_WRITE_SINGLE_REG (06)
StartAddr	Register Address	16#0001 to 16#FFFF
pData	Pointer to the 16-bit data that will be sent.	The data **MUST** be at least 2 bytes in size.

Write Multiple Coils (15)

The function code is used to write each coil in a sequence of coils to either ON or OFF.

Required Input	Description	Value
Func	Function Code	MB_FUNC_WRITE_MULT_COILS (15)
StartAddr	Starting Address	16#0001 to 16#FFFF
Quantity	Quantity of Input Registers (N)	1 to 1968
pData	Pointer to data to the data that will be sent.	The data **MUST** be at least N/8 (plus 1 if the remainder is different to 0) bytes in size.

Notes:

- The first least-significant 8 bits of the request will be from the first byte of the data pointed to. Thus in an AC30 application, if, for example, 16 coils are sent from a UINT type of data then the byte order will need to be swapped.
- If the quantity of outputs is not an even number of 8, then the remaining most-significant bits of the last byte will be filled with zeros.

Write Multiple Registers (16)

The function code is used to write a block of contiguous registers.

Required Input	**Description**	**Value**
Func	Function Code	MB_FUNC_WRITE_MULT_REGS (16)
StartAddr	Starting Address	16#0001 to 16#FFFF
Quantity	Quantity of Input Registers (N)	1 to 123
pData	Pointer to data to the data that will be sent.	The data **MUST** be at least N*2 bytes in size.

Modbus Exception Codes

Modbus exception codes sent by the slave will be shown at the output **Error** of the Modbus Request function block. These are represented by the following enumerated values.

Exception code	**Enumnerated value**
16#01	ERROR_ILLEGAL_FUNCTION
16#02	ERROR_ILLEGAL_DATA_ADDR
16#03	ERROR_ILLEGAL_DATA_VALUE
16#04	ERROR_SLAVE_DEVICE_FAILURE
16#05	ERROR_ACKNOWLEDGE
16#06	ERROR_SLAVE_DEVICE_BUSY
16#08	ERROR_MEMORY_PARITY_ERROR
16#0A	ERROR_GATEWAY_PATH_UNAVAIL
16#0B	ERROR_GATEWAY_TARGET_FAILED

Example

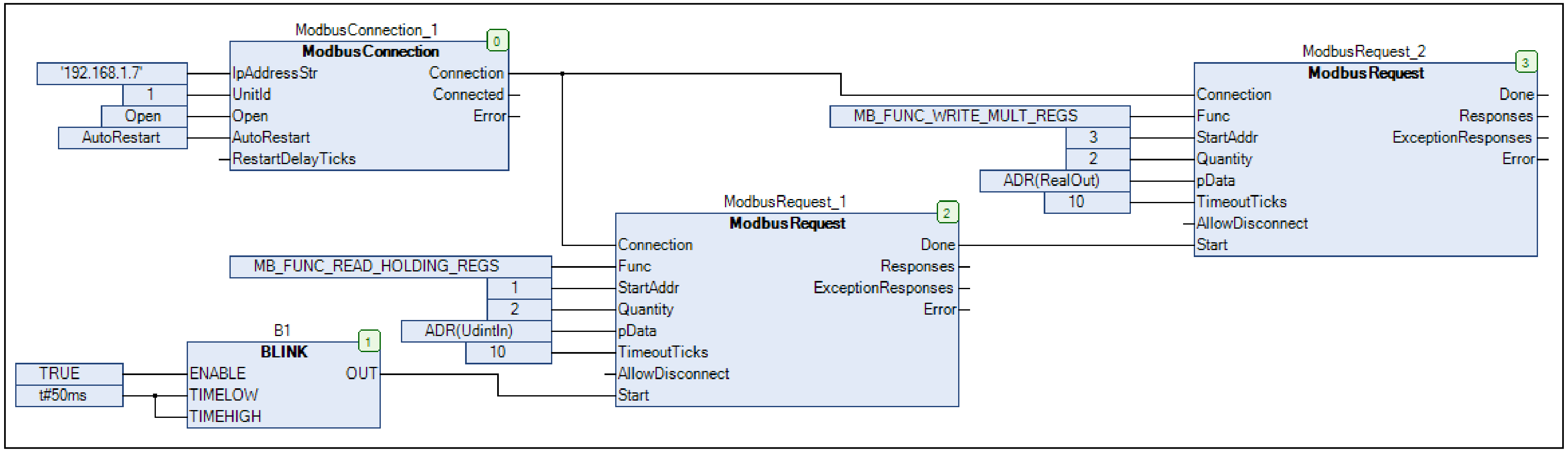
This example connects to a Modbus slave at IP address 192.168.1.7. By setting AutoRestart to TRUE the master will attempt to re-connect to the slave if the connection is lost.

Two Modbus Requests are associated with this connection:

The first request is using the **Read Holding Registers** function reading from 2 registers from address 1. The data is will be written into the 32-bit unsigned integer variable **UdintIn**. The request is made every 100ms using a Blink function.

The second request is using the **Write Multiple Registers** function writing to 2 registers from address 3. The data is read from the 32-bit real variable **RealOut**. The request is initiated when the first request function block has completed.

Note for 32-bit data types as used in this example, on an AC30 platform the most-significant 16 bits of data will be sent first (high word first).


ModbusConnection_1
0
ModbusConnection
'192.168.1.7'
1
Open
AutoRestart
IpAddressStr
UnitId
Open
AutoRestart
RestartDelayTicks
Connection
Connected
Error
ModbusRequest_2
3
ModbusRequest
MB_FUNC_WRITE_MULT_REGS
3
2
ADR(RealOut)
10
Connection
Func
StartAddr
Quantity
pData
TimeoutTicks
AllowDisconnect
Start
Done
Responses
ExceptionResponses
Error
ModbusRequest_1
2
ModbusRequest
MB_FUNC_READ_HOLDING_REGS
1
2
ADR(UdintIn)
10
Connection
Func
StartAddr
Quantity
pData
TimeoutTicks
AllowDisconnect
Start
Done
Responses
ExceptionResponses
Error
B1
1
BLINK
TRUE
t#50ms
ENABLE
TIMELOW
TIMEHIGH
OUT