APPLICATION EXAMPLE

AC500 V3 MODBUS RTU

USING MODRTUTOKEN AND MODRTUREAD / MODRTUWRITE

Contents

1 Disclaimer

A. For customers domiciled outside Germany /

Für Kunden mit Sitz außerhalb Deutschlands

„Warranty, Liability:

The user shall be solely responsible for the use of this products described within this file. ABB shall be under no warranty whatsoever. ABB's liability in connection with application of the products or examples provided or the files included within this products, irrespective of the legal ground, shall be excluded. The exclusion of liability shall not apply in the case of intention or gross negligence. The present declaration shall be governed by and construed in accordance with the laws of Switzerland under exclusion of its conflict of laws rules and of the Vienna Convention on the International Sale of Goods (CISG)."

„Gewährleistung und Haftung:

Der Nutzer ist allein für die Verwendung des in diesem Dokument beschriebenen Produkte und beschriebenen Anwendungsbeispiele verantwortlich.

ABB unterliegt keiner Gewährleistung. Die Haftung von ABB im Zusammenhang mit diesem Anwendungsbeispiel oder den in dieser Datei enthaltenen Dateien - gleich aus welchem Rechtsgrund - ist ausgeschlossen. Dieser Ausschluss gilt nicht im Falle von Vorsatz oder grober Fahrlässigkeit. Diese Erklärung unterliegt Schweizer Recht unter Ausschluss der Verweisungsnormen und des UN-Kaufrechts (CISG)."

B. Nur für Kunden mit Sitz in Deutschland

„Gewährleistung und Haftung:

Die in diesem Dokument beschriebenen Anwendungsbeispiele oder enthaltenen Dateien beschreiben eine mögliche Anwendung der AC500 bzw. zeigen eine mögliche Einsatzart. Sie stellen nur Beispiele für Programmierungen dar, sind aber keine fertigen Lösungen. Eine Gewähr kann nicht übernommen werden.

Der Nutzer ist für die ordnungsgemäße, insbesondere vollständige und fehlerfreie Programmierung der Steuerungen selbst verantwortlich. Im Falle der teilweisen oder ganzen Übernahme der Programmierbeispiele können gegen ABB keine Ansprüche geltend gemacht werden.

Die Haftung von ABB, gleich aus welchem Rechtsgrund, im Zusammenhang mit den Anwendungsbeispielen oder den in dieser Datei enthaltenen Beschreibung wird ausgeschlossen. Der Haftungsausschluss gilt jedoch nicht in Fällen des Vorsatzes, der groben Fahrlässigkeit, bei Ansprüchen nach dem Produkthaftungsgesetz, im Falle der Verletzung des Lebens, des Körpers oder der Gesundheit oder bei schuldhafter Verletzung einer wesentlichen Vertragspflicht. Im Falle der Verletzung einer wesentlichen Vertragspflicht ist die Haftung jedoch auf den vertragstypischen, vorhersehbaren Schaden begrenzt, soweit nicht zugleich ein anderer der in Satz 2 dieses Unterabsatzes erwähnten Fälle gegeben ist. Eine Änderung der Beweislast zum Nachteil des Nutzers ist hiermit nicht verbunden.

Es gilt materielles deutsches Recht unter Ausschluss des UN-Kaufrechts."

2 Introduction

2.1 Scope of the document

To use the AC500 V3 or AC500-eCo V3 as a **Modbus RTU client** two possible programming methods can be used:

- Using the modbus client function block **ModRtuMast** and program all read and/or write jobs in a sequence.
  If several ModRtuMast are used, e.g. for several sever connections, the sequence must take care that only one of those ModRtuMast function blocks is active at a time.
- Using the function blocks **ModRtuToken** along with **ModRtuRead** and/or **ModRtuWrite** and/or **ModRtuReadWrite23**
  Instead of programming a sequence a simple connection via structure variables between the function blocks is necessary to ensure the sequencial functionality of the Modbus RTU.

Only one of the two above mentioned methods can be used for one COM_x interface at a time. A mix will not work.

This application example explains the use of the second method, **using the ModRtuToken, ModRtuRead and ModRtuWrite.**

The application example uses one AC500 V3 and one AC500-eCo V3 as modbus servers.
So also the configuration for an AC500 V3 as a Modbus RTU server can be seen in the project.

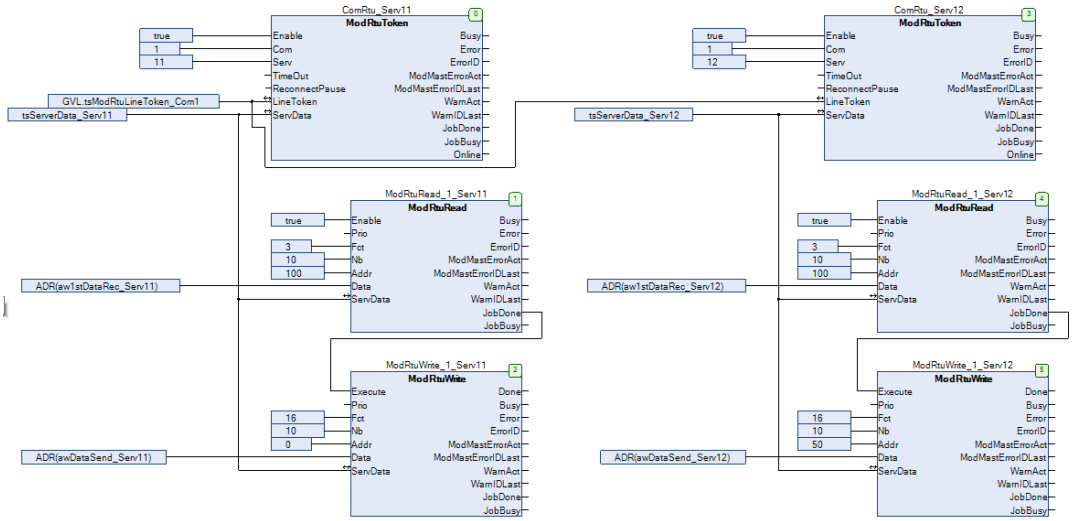
The use of **ModRtuToken** together with **DrvModbusRtu** or **DrvModbusRtuBroadcast** for the same COM interface is mentioned in chapter 4.1.1

2.2 Compatibility

The application example explained in this document has been used with the below engineering system versions. They should also work with other versions, nevertheless some small adaptations may be necessary, for future versions.

- AC500 V3 and AC500-eCo V3 PLC
- Automation Builder 2.4.1 or newer

2.3 Overview

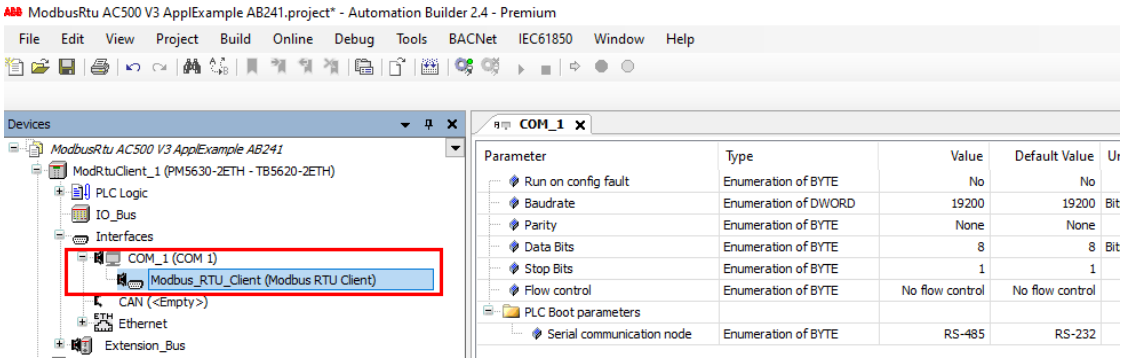

ComRtu_Serv11
ModRtuToken
true
1
11
Enable
Com
Serv
TimeOut
ReconnectPause
LineToken
ServData
Busy
Error
ErrorID
ModMastErrorAct
ModMastErrorIDLast
WarnAct
WarnIDLast
JobDone
JobBusy
Online
GVL.tsModRtuLineToken_Com1
tsServerData_Serv11
ComRtu_Serv12
ModRtuToken
true
1
12
tsServerData_Serv12
ModRtuRead_1_Serv11
ModRtuRead
true
3
10
100
Prio
Fct
Nb
Addr
Data
ADR(aw1stDataRec_Serv11)
ModRtuRead_1_Serv12
ADR(aw1stDataRec_Serv12)
ModRtuWrite_1_Serv11
ModRtuWrite
Execute
Done
16
10
0
ADR(awDataSend_Serv11)
ModRtuWrite_1_Serv12
50
ADR(awDataSend_Serv12)

3 Configuration

3.1 Modbus RTU client

In the Automation Builder the empty COM slot must be replaced by a COM_X slot.

Then the parameters for the interface must be adapted. If more than one server is used the serial communication mode must be set to “RS485”. All other interface parameters must be identical to the parameters of the connected servers.



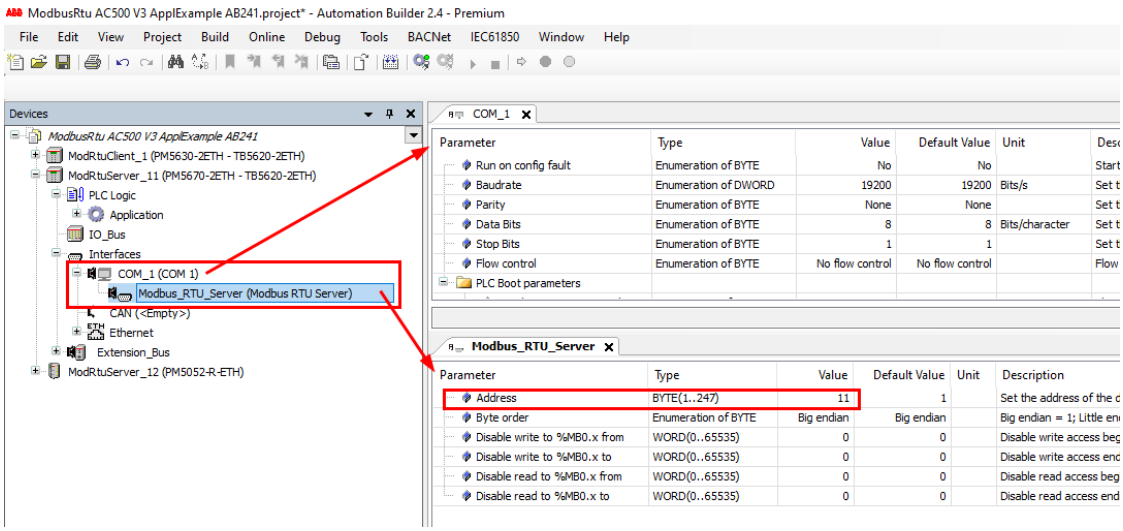
Beneath the COM_X interface a “Modbus_RTU_Client” node must be added, which has no parameters, as the node address of the client is always “0”.

3.2 Modbus RTU server

In the Automation Builder the empty COM slot must be replaced by a COM_X slot.

Then the parameters for the interface must be adapted. If more than one server is used the serial communication mode must be set to “RS485”. All other interface parameters must be identical to the parameters of the connected client.

Beneath the COM_X interface a “Modbus_RTU_Server” node must be added, which has some parameters. Mainly the node address needs to be set there.



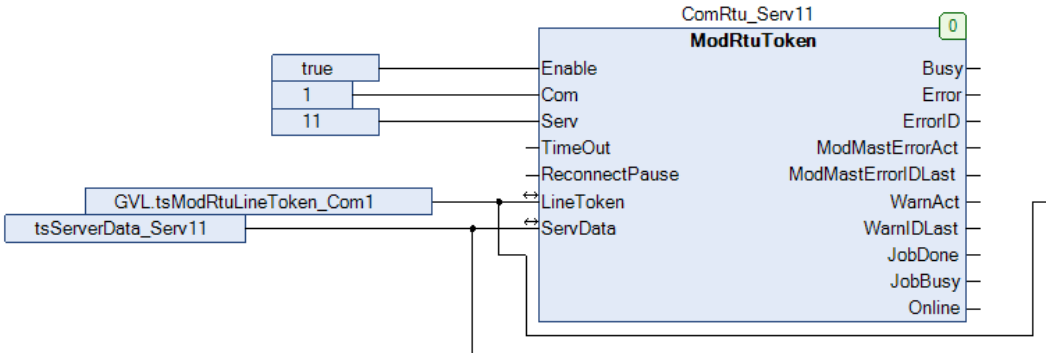
4 System Technology

The described function blocks can be found in the library “**AC500_ModbusRtu.library**”

The use of the function blocks ModRtuToken, ModRtuRead and ModRtuWrite and ModRtuReadWrite23 is based on the following principles.

	Note: **Either use ModRtuToken or ModRtuMast** function blocks for one COM interface. The use of both function blocks for the same COM interface in parallel will not work. This is due to the fact, that the ModRtuToken uses internally a ModRtuMast instance that will occupy the COM interface all the time.

4.1 General



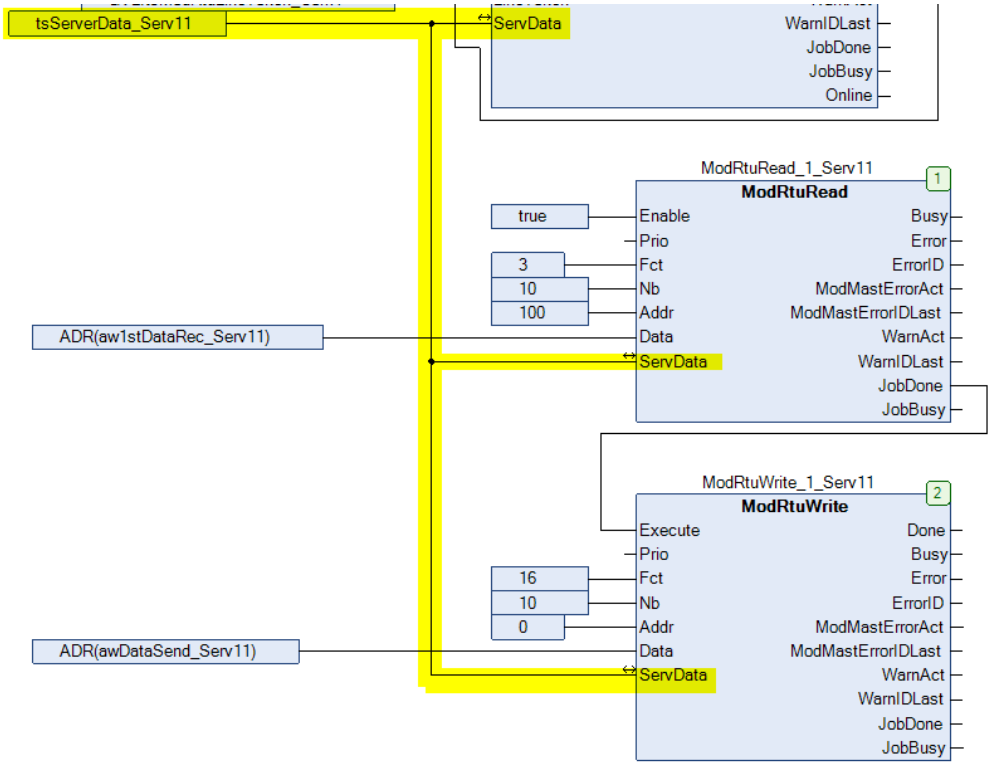
ModRtuToken uses internally the ModRtuMast functionblock.

All ModRtuToken function blocks that are configured for the same COM interface need to be connected together via one single **structured variable of type “ModRtuTokenType”**, which needs to be connected to the input “**LineToken**” of each ModRtuToken block.
All these blocks must be called within the **same PLC task**.

The LineToken variable is best to be declared in a global variable list. Then it can be used in several programs, e.g. if for each server an own program is used.

Via this variable the Line Token is hand over from one ModRtuToken block to the others.

If a ModRtuToken block has the Line Token it executes the jobs towards the server with address set at input Serv. The parameters and data of these jobs are handed over from the ModRtuRead and/or ModRtuWrite function blocks which are connected via the variable at input “ServData”.



The connection to the related (same server) ModRtuRead and ModRtuWrite function blocks has to be made via this one **structured variable of type "ModRtuGenDevDataType"**, which needs to be connected to the input "**ServData**" of each block.

4.1.1 Generic devices and drives on the same Modbus RTU line

The generic function blocks ModRtuToken, ModRtuRead and ModRtuWrite can of course also be used to establish the connection to drives / frequency converters etc.

But together with ABB ACS or DCS drives it´s recommended to use the "AC500_Drives" library.
The function blocks DrvModbusRtu and DrvModbusRtuBroadcast of the AC500_Drives library have the same input LineToken and therefore can be used in combination with the ModRtuToken function block, as long as they use the same variable for all blocks of one and the same COM_X interface.

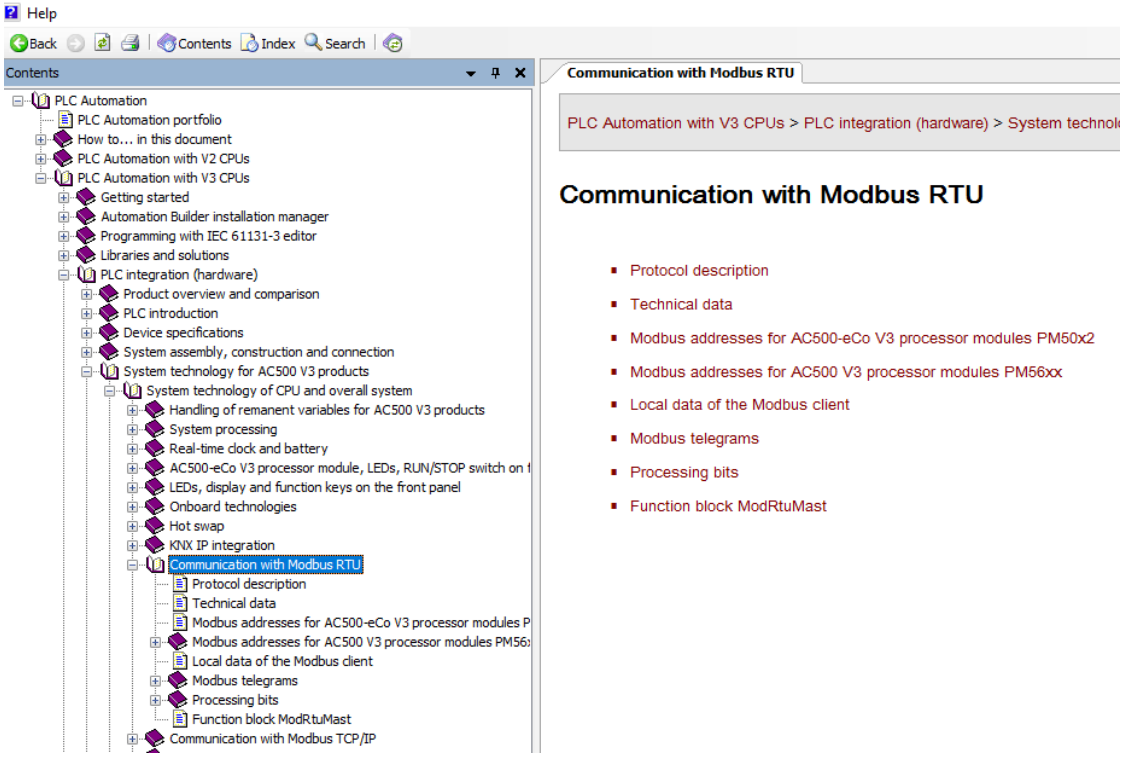
See also:

- examples and documentation of the drives library, which is installed with Automation Builder here:
  C:\Users\Public\Documents\AutomationBuilder\Examples\PS5605-Drives\Modbus_RTU
- Automation Builder Help in chapter:
  PLC Automation with V3 CPUs > Libraries and solutions > ACS/DCS drives libraries - System technology > Introduction > Overview of the drives library for V3 PLC > Modbus RTU

4.2 Documentation

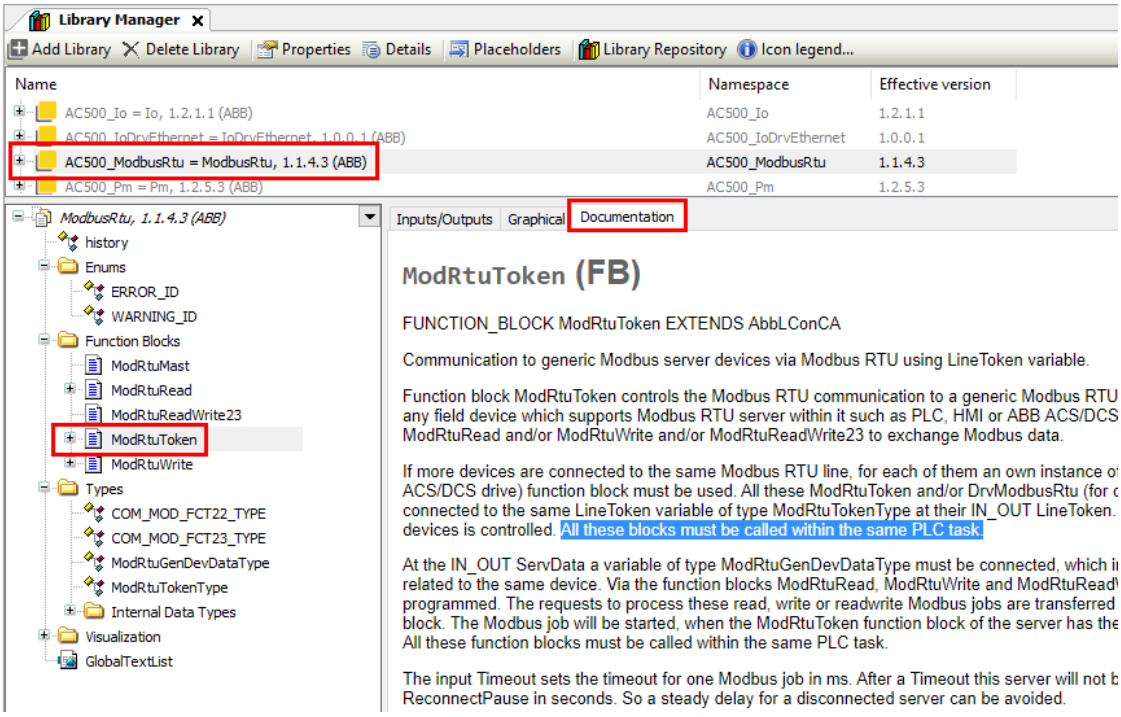
4.2.1 System technology about Modbus RTU Communication

See Automation Builder Help in chapter:
PLC Automation with V3 CPUs > PLC integration (hardware) > System technology for AC500 V3 products > System technology of CPU and overall system > Communication with Modbus RTU



4.2.2 Function blocks of library AC500_ModbusRtu

The detailed documentation can be found in Automation Builder, opening the library AC500_ModbusRtu. Then select the function block e.g. “ModRtuToken” and goto the tab “Documentation”.



Description of inputs and outputs can also be found there.

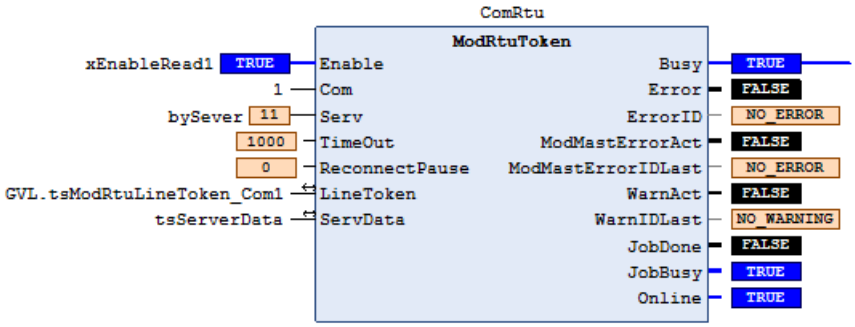
The function block must be used in same PLC task than other function blocks using the same variable on IN_OUT LineToken and IN_OUT ServData

InOut:

Scope	Name	Type	Initial	Comment
Input	Com	BYTE	1	2: Modbus communication port - e.g: 1 = COM1 - changes are valid only after rising edge of Enable input (FALSE -> TRUE). Valid values are 1 to 3 depending on PLC type and configuration.
	Serv	BYTE	1	3: Modbus RTU server address - if changed while a modbus job is running, this job will be finished with previous Serv address; 0 = Broadcast. valid range is 0..247.
	TimeOut	WORD	1000	4: Timeout [ms] for ModRtuMast function block - TimeOut value should be at least 50ms.
	ReconnectPause	WORD	0	5: Pause in seconds before next retry to connect after a timeout was detected. Timeout is detected with ModMastErrorID = 16#120
Inout	LineToken	ModRtuTokenType		6: Reference variable to connect to other Modbus RTU function blocks ModRtuRead, ModRtuWrite and ModRtuReadWrite23.
	ServData	ModRtuGenDevDataType		7: Modbus RTU server reference variable to connect to all function blocks of this server device.
Output	ErrorID	ERROR_ID		4: Error codes
	ModMastErrorAct	BOOL	FALSE	5: Active error in ModMast. Operation is running with error . This output is TRUE for at least one cycle or until Enable is set to FALSE. The output ModMastErrorIDLast gives more details about the last error.
	ModMastErrorIDLast	ERROR_ID		6: Error code of last active error in ModRtuMast, For error code description, refer to error codes in ERROR_ID.
	WarnAct	BOOL	FALSE	7: Operation is running with warning . This output is TRUE for at least one cycle or until Enable is set to FALSE. The output WarnIDLast gives more details about the warning.
	WarnIDLast	WARNING_ID		8: Warning code of last active warning
	JobDone	BOOL	FALSE	9: Modbus job finished
	JobBusy	BOOL	FALSE	10: Modbus job active
	Online	BOOL	FALSE	11: Connection established without error, all read jobs without errors and all Modbus Jobs finished within time set at input Timeout.

4.3 Some rule of thumb

4.3.1 ModRtuToken



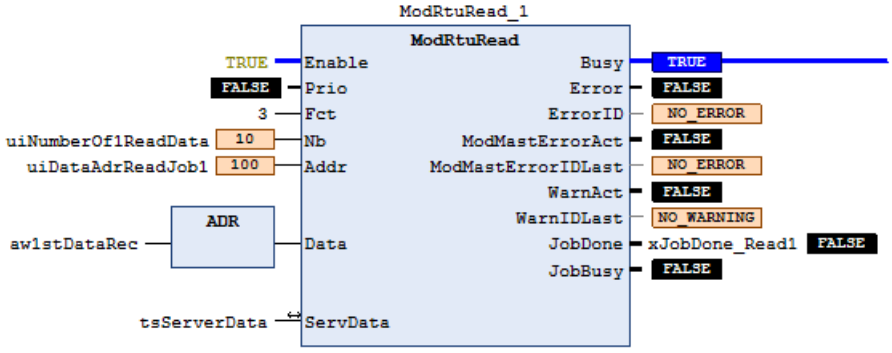
4.3.1.1 Inputs:

- Enable – can be set to fix TRUE, if Com and Serv input is fix.
  (Com and Server input are only read in at rising edge of Enable)
- Com – Communication interface number, mostly 1 . With AC500-eCo option boards can also be 2 or 3.
- Server = server node address
- TimeOut can be left to default (1000 = 1s). Waiting time for server to reply. After that a timeout error is set.
- ReconnectionPause. Can be used to skip servers, which are not connected, for specific time before a retry to connect is made again. Default is 0. Good rule would be to set to e.g. 3 for 3 seconds.
- LineToken: Connect the LineToken variable. Best to declare it in a global varlist.
- ServData: Connect the ServData variable, which can be local and needs to be connected to the ModRtuRead and/or ModRtuWrite and/or ModRtuReadWrite23 blocks.

4.3.1.2 Outputs:

- ModMastErrorAct is indicating that actually an error is active.
- ModMastErrorIDLast is the last occurred ErrorID of the internally called ModRtuMast function block. It will only be cleared if the Enable input is set to FALSE.
  So it might indicate an earlier occurred error, which is not active any more.
- ModMastWarnAct is indicating that actually a warning is active.
- WarnIDLast is the last occurred WarningID. It will only be cleared if the Enable input is set to FALSE.
  So it might indicate an earlier occurred warning, which is not active any more.
- JobDone is indicating for one cycle, that the actual modbus job is finished.
- JobBusy is indicating that a modbus job is active
- Online will be set if at least one modbus job was finished without errors and no further error have been occurred.

4.3.2 ModRtuRead



4.3.2.1 Inputs

- Enable can be set to TRUE. Then the read job is executed continuosly.

  If the Enable input is set to TRUE a new job request will be created each time the previous job was finished. At that moment the inputs Fct, Nb and Addr are read in and kept internally until the next job is started. The next job starts, when this function blocks gets the internal Modbus Token.

- Prio is not yet implemented. Can be left open.

- Fct – set the modbus function code, 3 or 4 for reading words. Be aware that the BITADR operator is not working with this functionblock in case the function code 1 or 2 is used to read bits.

- Nb – Number of data.
  In case of Fct = 3 or 4, the number of words with maximum 125.
  In case of Fct = 1 or 2, number of Bits with maximum 2000.

- Addr – address of the data in the server

- Data – address of the data in this PLC. Keep this data address stable until the JobDone output indicates, that the data is written to it.

4.3.2.2 Outputs

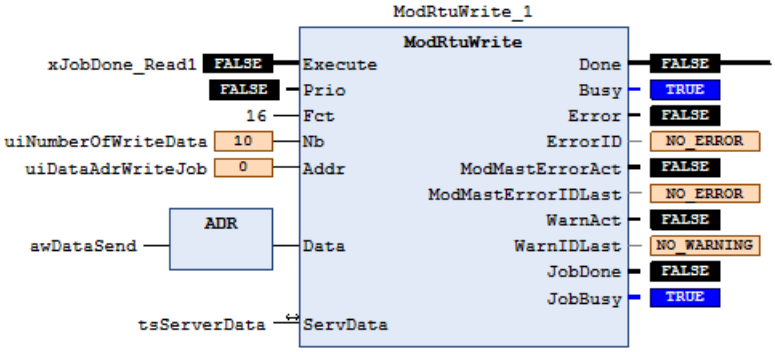
- Busy is set, if the function block has requested a job or is executing the job.

- ModMastErroAct, modMastErrorIDLast, WarnAct, WarnIDLast – see ModRtuToken.

- JobDone is indicating for one cycle, that the actual job is finished. This output can be used to e.g. start a ModRtuWrite function block at input Execute.

- JobBusy is indicating that the modbus job of this function block is actually executed.

4.3.3 ModRtuWrite



4.3.3.1 Inputs

- Execute needs a rising edge to request and start a new modbus write job.

  With the rising edge the inputs Fct, Nb and Addr are read in and kept internally until the internal Modbus Token is assigned to this function block and the write job is started. The data at the address of Data must be kept stable until the write job is finished, which is indicated via the JobDone output.

- Prio is not yet implemented. Can be left open.

- Fct – set the modbus function code.
  5 = write one bit.
  6 = write one word.
  15 = write n bits.
  16 = write n words, recommended as most servers support this function code.
  22 = mask write
  23 = read /write in one job – special structure is needed at the address of DATA.
  Be aware that the BITADR operator is not working with this function block, if function code 5 or 15 is used to write bits.

- Nb – Number of data.
  In case of Fct = 16 the number of words with maximum 125.
  In case of Fct = 15 number of bits with maximum 2000.

- Addr – address of the data in the server

- Data – address of the data in this PLC. Keep this data address stable until the JobDone output indicates, that the data is written to it.

4.3.3.2 Outputs

- Busy is set, if the function block has requested a job or is executing the job.

- ModMastErroAct, modMastErrorIDLast, WarnAct, WarnIDLast – see ModRtuToken.

- JobDone is indicating for one cycle, that the actual job is finished.

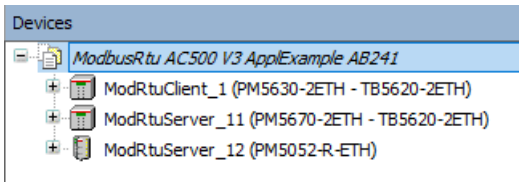
- JobBusy is indicating that the modbus job of this function block is actually executed.

4.3.4 ModRtuReadWrite23

For detailed description of this function block, which can be used to read and write data in one modbus job using function code 23, please see the documentation in the library directly.

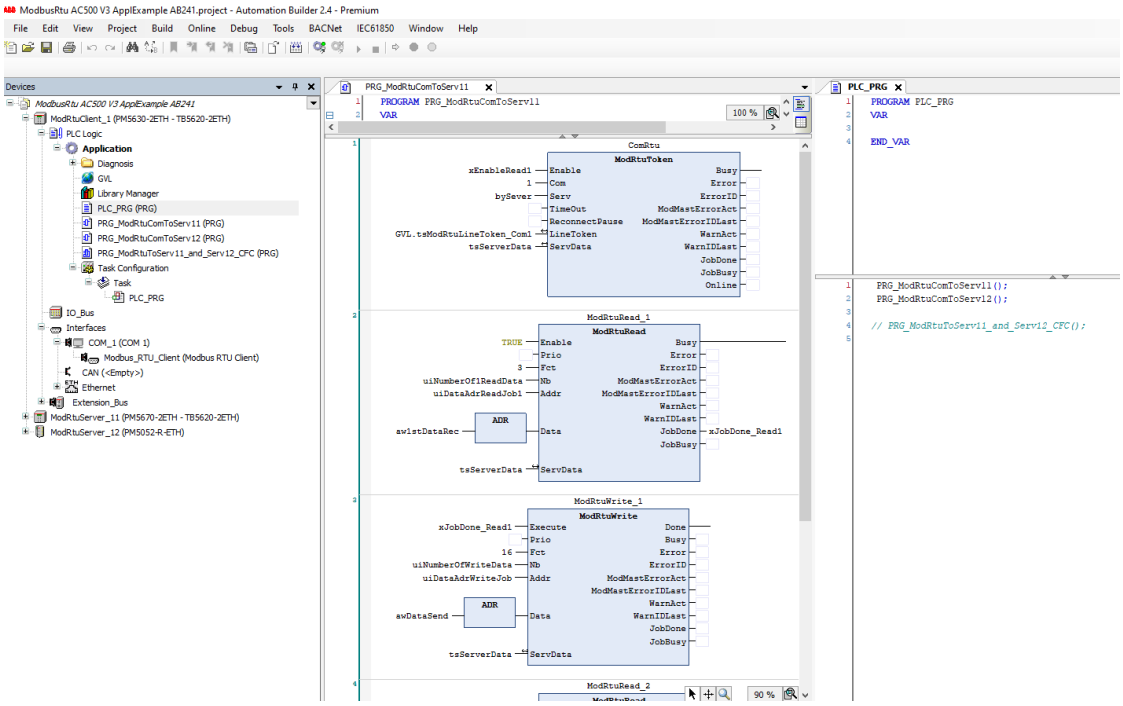
5 Application example project

The application examples includes the Modbus client (PM5630), one Modbus server with address 11 (PM5670) and one server with address 12 (PM5052).



The main program PLC_PRG runs in a 250ms task, which makes it easier to see the set outputs. Of course, it can also be run in a 10ms task to achieve faster communication.

Two program parts are provided. Only one should be called from the PLC_PRG:



- **PRG_ModRtuComToServ11** and **PRG_ModRtuComToServ12**
  Both programs are written in FUP and are identical, except the server and data addresses, which can be adapted in the declaration part.
  They contain two read and one write job blocks.
  This is an example, how a generic program can be reused, if several identical servers needs to be connected.

- **PRG_ModRtuToServ11_and_Serv12_CFC**

  This program is written in CFC to show the structure of the used function blocks.
  It has only one read and one write block for each of the two servers.

6 Template Boxes

6.1 Caution

CAUTION!

Generally, the user in all applications is fully and alone responsible for checking all functions carefully, especially for safe and reliable operation.

6.2 Note

Note: The Function Blocks contained in the example can only be executed in RUN mode of the PLC, but not in simulation mode.

ABB AG
Eppelheimer Straße 82
69123 Heidelberg, Germany
Phone: +49 62 21 701 1444
Fax: +49 62 21 701 1382
E-Mail: plc.support@de.abb.com
www.abb.com/plc