ctrlX - CORE

- Communication with XM

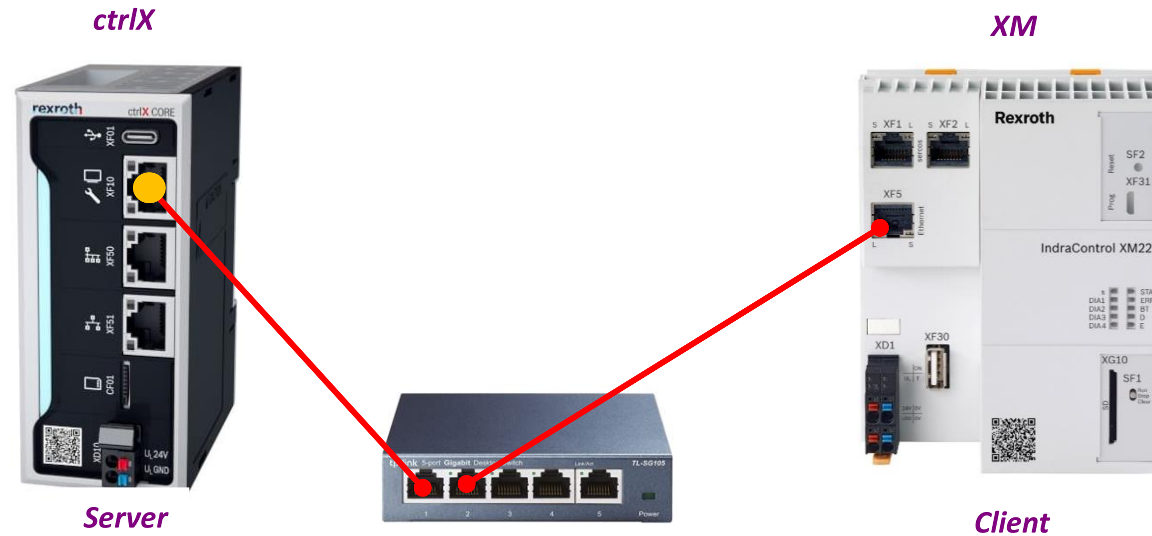
- Connection using Modbus

rexroth
A Bosch Company

Jordi Laboria (DCET/SLF4-ES)

2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

In the next tutorial we will see how a communication works using the Modbus protocol to pass data from a ctrlX to an XM and vice versa.
The system uses the Server - Client model



The Modbus operating modules that we are going to use allow the passage of up to four types of areas.

I / O Area
65536 Bit Coil
65536 Discrete Input
65536 Word Input Register
65536 Word Holding Register

Although this is relatively true, the reality is that the allocation of areas in both teams, differs slightly in the part called Coil.
So care must be taken when transferring data.

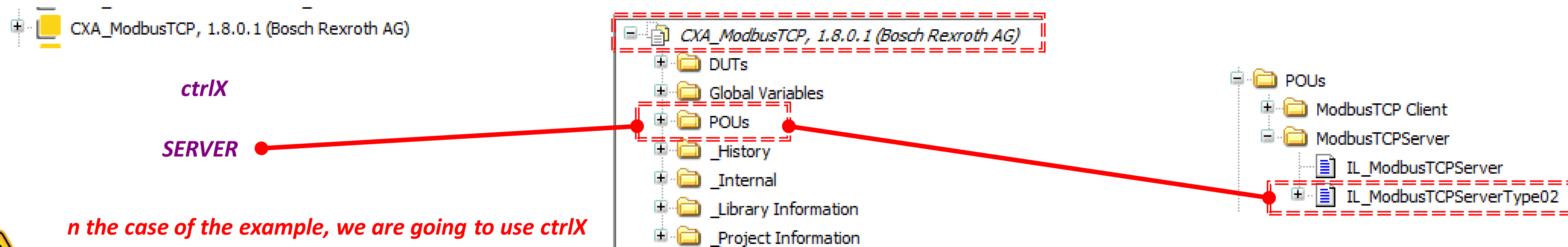
I / O Area	ctrlX
65536 Bit Coil	Byte
65536 Discrete Input	Byte
65536 Word Input Register	Word
65536 Word Holding Register	Word

I / O Area	XM
65536 Bit Coil	Bool
65536 Discrete Input	Bool
65536 Word Input Register	Word
65536 Word Holding Register	Word

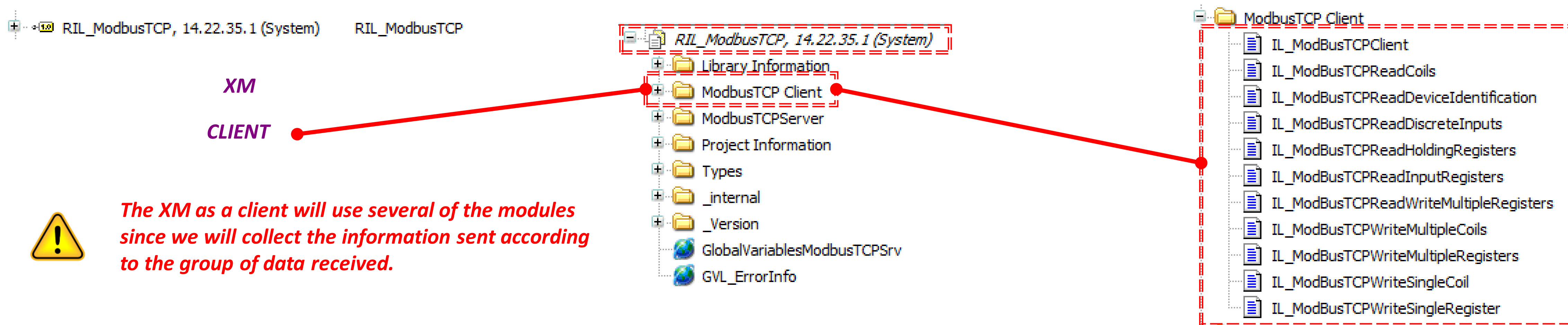
2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

ctrlX - Libraries used for Modbus communication

For this we are going to use the following libraries that must be installed on both computers



n the case of the example, we are going to use ctrlX as a server and to send the data, we will use the module "… Type2", which allows us to send (and receive in two areas) the four commented areas.



The XM as a client will use several of the modules since we will collect the information sent according to the group of data received.

2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

rexroth
A Bosch Company

Module used in ctrlX to send data. This module also allows us to read the CoilData and HoldingRegisterData that can be modified from the Client

Variables used in ctrlX

```
VAR
    fbModbusComToXM: IL_ModbusTCPServerType02;
    arCoilData: ARRAY [0..7] OF BYTE;
    arInputData: ARRAY [0..7] OF BYTE;
    bEnable: BOOL;
    bInOperation: BOOL;
    bError: BOOL;
    stRegisterData:dutRegisterData;
    stHoldingRegisterData:dutRegisterData;
END_VAR
```

- fbModbusComToXM: *Modbus communications module*
- arCoilData: *Array in byte format that we will send to the XM (These bytes can also be written from the "Client")*
- arInputData: *Array in byte format that we will send to the XM*
- bEnable: *Module activation bit*
- bInOperation: *Data sending activated*
- bError: *Active error in the module*
- stRegisterData: *Data structure for the "Register Data"*
- stHoldingRegisterData: *Data structure for the "Holding Register Data"*

(These structure can also be written from the "Client")

```
TYPE dutRegisterData :
STRUCT
    arWord:   ARRAY[0..999]OF WORD;     // 0 - 999
    arInt:    ARRAY[0..999]OF INT;      // 1000 - 1999
    arUint:   ARRAY[0..999]OF UINT;     // 2000 - 2999
    arDint:   ARRAY[0..499]OF DINT;     // 3000 - 3999
    arUdint:  ARRAY[0..499]OF UDINT;    // 4000 - 4999
    arReal:   ARRAY[0..499]OF REAL;     // 5000 - 5999
    arString: ARRAY[0..99]OF STRING(19);// 6000 - 6999

END_STRUCT
END_TYPE
```

Communications control module

```
fbModbusComToXM( // IL_ModbusTCPServerType02
    Enable:=bEnable ,
    InOperation=> bInOperation,
    Error=> bError,
    ErrorID=> ,
    ErrorIdent=> ,
    Port:= ,
    CoilData:=ADR(arCoilData) ,
    SizeOfCoilData:=SIZEOF(arCoilData) ,
    InputData:=ADR(arInputData) ,
    SizeOfInputData:=SIZEOF(arInputData) ,
    HoldingRegisterData:=ADR(stHoldingRegisterData) ,
    SizeOfHoldingRegisterData:=SIZEOF(stHoldingRegisterData) ,
    InputRegisterData:=ADR(stRegisterData) ,
    SizeOfInputRegisterData:=SIZEOF(stRegisterData) ,
    Stats=> );
```

- fbModbusComToXM: *Module used for the Server*
- Enable: *Activate communication*
- InOperation: *Module in Operation*
- Error: *Module in Error*
- Port:

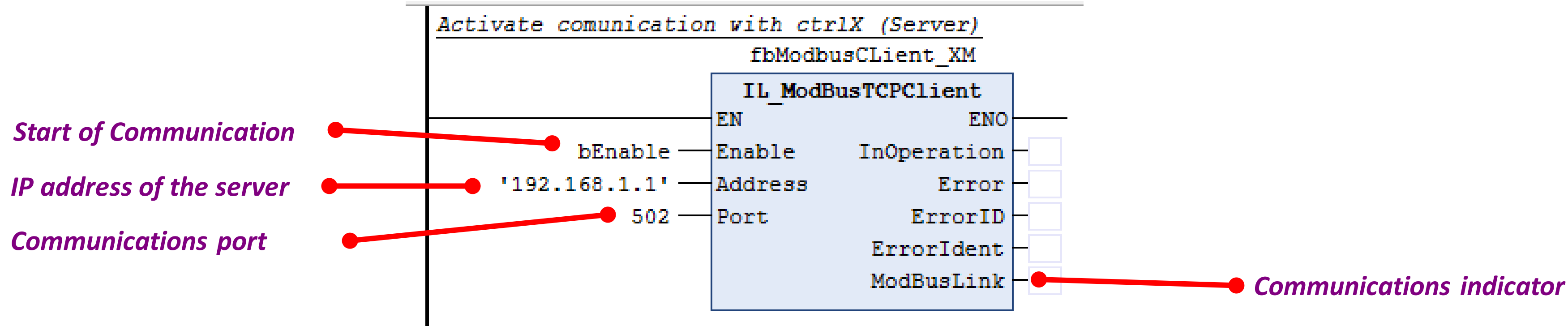
- CoilData: *Direction extraction*
- SizeOfCoilData: *Defining the size of the structure*

As can be seen, we can use structures with different types of data to use in communication.

2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

rexroth
A Bosch Company

As we have commented previously, on the client side, more modules are used to carry out the readings or, where appropriate, the writings. The first module to use will be the "IL_ModbusTCPClient" that helps us to establish communication and obtain the communications indicator.



The ctrlX communications module transmits the data to the XM, in some cases as read / write and must be "read" or "written according to the provisions of the generated structures themselves, as described below.

ctrlX

```
fbModbusComToXM( // IL_ModbusTCPServerType02
    Enable:=bEnable ,
    InOperation=> bInOperation,
    Error=> bError,
    ErrorID=> ,
    ErrorIdent=> ,
    Port:= ,
    CoilData:=ADR(arCoilData) ,
    SizeOfCoilData:=SIZEOF(arCoilData) ,
    InputData:=ADR(arInputData) ,
    SizeOfInputData:=SIZEOF(arInputData) ,
    HoldingRegisterData:=ADR(stHoldingRegisterData) ,
    SizeOfHoldingRegisterData:=SIZEOF(stHoldingRegisterData) ,
    InputRegisterData:=ADR(stRegisterData) ,
    SizeOfInputRegisterData:=SIZEOF(stRegisterData) ,
    Stats=> );
```

XM

		Annotation
fbModbusCLient	IL_ModBusTCPClient	Communication
fbModBusReadCoils	IL_ModBusTCPReadCoils	Reading Coils
fbModBusReadDiscreteInputs	IL_ModBusTCPReadDiscreteInputs	Read Discrete Inputs
fbModBusReadHoldingRegisters_Word	IL_ModBusTCPReadHoldingRegisters	HoldingRegisterData reading
fbModBusReadHoldingRegisters_int	IL_ModBusTCPReadHoldingRegisters	HoldingRegisterData reading
fbModBusReadHoldingRegisters_Uint	IL_ModBusTCPReadHoldingRegisters	HoldingRegisterData reading
fbModBusReadHoldingRegisters_Dint	IL_ModBusTCPReadHoldingRegisters	HoldingRegisterData reading
fbModBusReadHoldingRegisters_Udint	IL_ModBusTCPReadHoldingRegisters	HoldingRegisterData reading
fbModBusReadHoldingRegisters_Real	IL_ModBusTCPReadHoldingRegisters	HoldingRegisterData reading
fbModBusReadHoldingRegisters_String	IL_ModBusTCPReadHoldingRegisters	HoldingRegisterData reading
fbModBusWriteMultipleCoils	IL_ModBusTCPWriteMultipleCoils	Write CoilData
fbModbusWriteMultipleHoldingRegisters	IL_ModBusTCPWriteMultipleRegisters	Write HoldingRegistersData

The "InputRegisterData" area works exactly the same as the "HoldingRegisterData" area. However, it cannot be written.

2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

rexroth
A Bosch Company

The data sent or received as we have said can be of four different types:

COIL DATA:

Coil Data	
Options :	Read / Write
Structure :	Byte (ctrlX) / Bool (XM)
Define array size with :	SIZEOF()
Representation :	1 Array / 8 bits
Optional :	Use of structures

ctrlX
```
arCoilData: ARRAY [0..7] OF BYTE;
```
XM
```
arCoilData: ARRAY [0..7] OF BOOL;
```

INPUT DATA:

Input Data	
Options :	Read
Structure :	Byte (ctrlX) / Bool (XM)
Define array size with :	SIZEOF()
Representation :	1 Array / 8 bits
Optional :	Use of structures

ctrlX
```
arInputData: ARRAY [0..7] OF BYTE;
```
XM
```
arInputData: ARRAY [0..7] OF BOOL;
```

HOLDING REGISTER DATA:

HoldingRegisterData	
Options :	Read / Write
Structures :	Word
Define array size with :	SIZEOF()
Representation :	Word (16 bits)
Optional :	Use of structures

```
stHoldingRegisterData:dutRegisterData;
```

REGISTER DATA:

RegisterData	
Optiones :	Read
Structures :	Word
Define array size with :	SIZEOF()
Representation :	Word (16 bits)
Optional :	Use of structures

```
stRegisterData:dutRegisterData;
```

```
TYPE dutRegisterData :
STRUCT
  arWord:   ARRAY[0..999]OF WORD;     // 0 - 999
  arInt:    ARRAY[0..999]OF INT;      // 1000 - 1999
  arUint:   ARRAY[0..999]OF UINT;     // 2000 - 2999
  arDint:   ARRAY[0..499]OF DINT;     // 3000 - 3999
  arUdint:  ARRAY[0..499]OF UDINT;    // 4000 - 4999
  arReal:   ARRAY[0..499]OF REAL;     // 5000 - 5999
  arString: ARRAY[0..99]OF STRING(19);// 6000 - 6999

END_STRUCT
END_TYPE
```


2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

rexroth
A Bosch Company

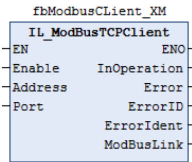
Example of sending data at the CoilData level, which is also useful for the "Discrete Inputs"

ctrlX

```
fbModbusComToXM( // IL_ModbusTCPServerType02
    Enable:=bEnable ,
    InOperation=> bInOperation,
    Error=> bError,
    ErrorID=> ,
    ErrorIdent=> ,
    Port:= ,
    CoilData:=ADR(arCoilData) ,
    SizeOfCoilData:=SIZEOF(arCoilData) ,
    InputData:=ADR(arInputData) ,
    SizeOfInputData:=SIZEOF(arInputData) ,
    HoldingRegisterData:=ADR(stHoldingRegisterData) ,
    SizeOfHoldingRegisterData:=SIZEOF(stHoldingRegisterData) ,
    InputRegisterData:=ADR(stRegisterData) ,
    SizeOfInputRegisterData:=SIZEOF(stRegisterData) ,
    Stats=> );
```

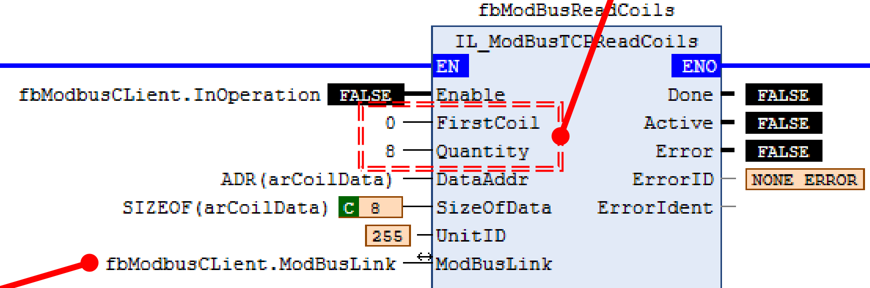
Sending / Receiving CoilData Area

Communications indicator extracted from the connection module



The maximum value of the "Quantity" parameter is 2000. If we want to access the following "CoilsData" we must modify the value of "FisrtCoil"

XM



arCoilData	ARRAY [0..7] OF BYTE
arCoilData[0]	BYTE
arCoilData[1]	BYTE
arCoilData[2]	BYTE
arCoilData[3]	BYTE
arCoilData[4]	BYTE
arCoilData[5]	BYTE
arCoilData[6]	BYTE
arCoilData[7]	BYTE

If we generate the array in Byte format, we will only see the first of the bits

arCoilData	ARRAY [0..7] OF BOOL
arCoilData[0]	BOOL
arCoilData[1]	BOOL
arCoilData[2]	BOOL
arCoilData[3]	BOOL
arCoilData[4]	BOOL
arCoilData[5]	BOOL
arCoilData[6]	BOOL
arCoilData[7]	BOOL

As already mentioned, both areas are different in the two teams. ctrlX sends the data in byte format so the first transmit byte sends or receives the first seven bits of the XM area

2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

rexroth
A Bosch Company

Example of sending data at the HoldingRegisterData level, which also serves for the RegisterData

ctrlX

```
fbModbusComToXM( // IL_ModbusTCPServerType02
    Enable:=bEnable ,
    InOperation=> bInOperation,
    Error=> bError,
    ErrorID=> ,
    ErrorIdent=> ,
    Port:= ,
    CoilData:=ADR(arCoilData) ,
    SizeOfCoilData:=SIZEOF(arCoilData) ,
    InputData:=ADR(arInputData) ,
    SizeOfInputData:=SIZEOF(arInputData) ,
    HoldingRegisterData:=ADR(stHoldingRegisterData) ,
    SizeOfHoldingRegisterData:=SIZEOF(stHoldingRegisterData) ,
    InputRegisterData:=ADR(stRegisterData) ,
    SizeOfInputRegisterData:=SIZEOF(stRegisterData) ,
    Stats=> );
```

Sending / Receiving HoldingRegisterData Area

XM

fbModbusCLient	IL_ModBusTCPClient
fbModBusReadCoils	IL_ModBusTCPReadCoils
fbModBusReadDiscreteInputs	IL_ModBusTCPReadDiscreteInputs
fbModBusReadHoldingRegisters_Word	IL_ModBusTCPReadHoldingRegisters
fbModBusReadHoldingRegisters_int	IL_ModBusTCPReadHoldingRegisters
fbModBusReadHoldingRegisters_Uint	IL_ModBusTCPReadHoldingRegisters
fbModBusReadHoldingRegisters_Dint	IL_ModBusTCPReadHoldingRegisters
fbModBusReadHoldingRegisters_Udint	IL_ModBusTCPReadHoldingRegisters
fbModBusReadHoldingRegisters_Real	IL_ModBusTCPReadHoldingRegisters
fbModBusReadHoldingRegisters_String	IL_ModBusTCPReadHoldingRegisters
fbModBusWriteMultipleCoils	IL_ModBusTCPWriteMultipleCoils
fbModbusWriteMultipleHoldingRegisters	IL_ModBusTCPWriteMultipleRegisters

The data must be extracted in individual groups, if we want to access the different parts of the generated structure, therefore the same type of module is used, but each one for a different group.

ctrlX

stHoldingRegisterData	dutRegisterData
arWord	ARRAY [0..999] OF WORD
arInt	ARRAY [0..999] OF INT
arUint	ARRAY [0..999] OF UINT
arDint	ARRAY [0..499] OF DINT
arUdint	ARRAY [0..499] OF UDINT
arReal	ARRAY [0..499] OF REAL
arString	ARRAY [0..99] OF STRING(19)

XM

stHoldingRegisterData	
arWord	fbModBusReadHoldingRegisters_Word
arInt	fbModBusReadHoldingRegisters_int
arUint	fbModBusReadHoldingRegisters_Uint
arDint	fbModBusReadHoldingRegisters_Dint
arUdint	fbModBusReadHoldingRegisters_Udint
arReal	fbModBusReadHoldingRegisters_Real
arString	fbModBusReadHoldingRegisters_String

2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

rexroth
A Bosch Company

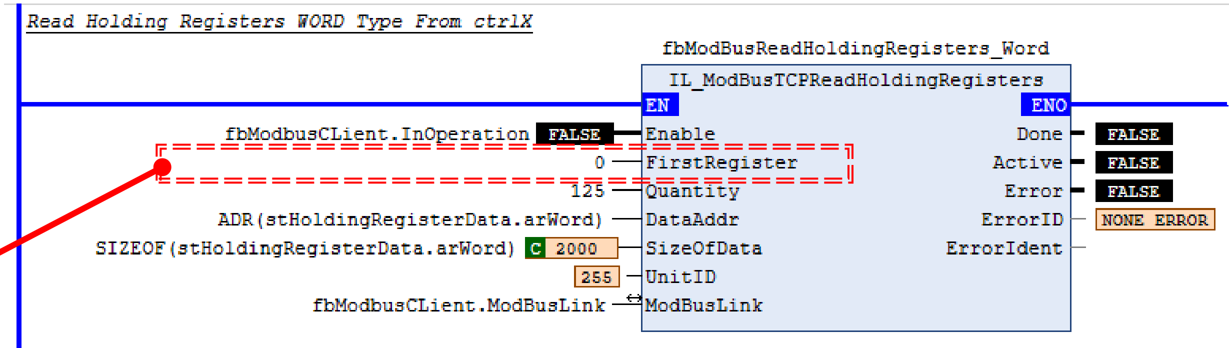
ctrlX - Example of control of HoldingRegisterData - Use of structures to send complex data

The communication structure of the HoldingRegisters area is in Word format, so we must calculate at which point each read / write sector is initialized.

The first 4 areas of the structure are in Word format, so the actual occupation corresponds 1 to 1. The following areas are in double word format, so the assigned value * 2 should be multiplied, in this way we can adjust the values and knowing what area we can work on.

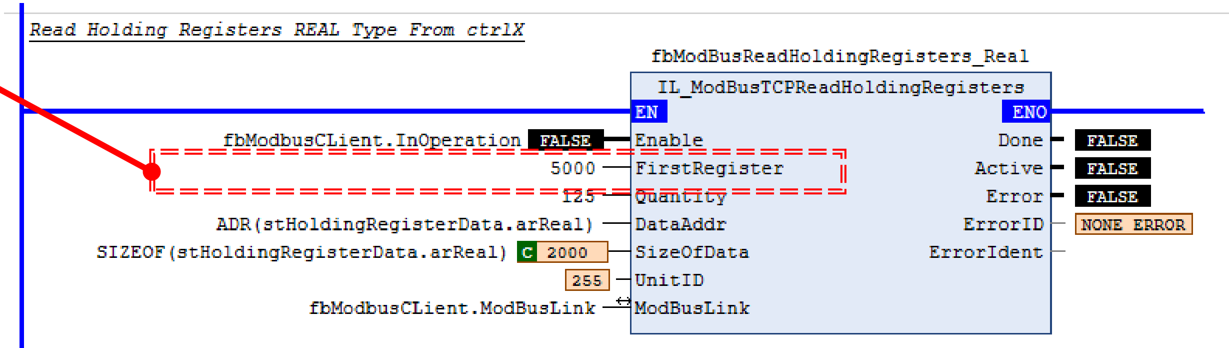
From the example and with the structure shown, each area has a starting point

stHoldingRegisterData	dutRegisterData	
arWord	ARRAY [0..999] OF WORD	0
arInt	ARRAY [0..999] OF INT	1000
arUint	ARRAY [0..999] OF UINT	2000
arDint	ARRAY [0..499] OF DINT	3000
arUdint	ARRAY [0..499] OF UDINT	4000
arReal	ARRAY [0..499] OF REAL	5000
arString	ARRAY [0..99] OF STRING(19)	6000



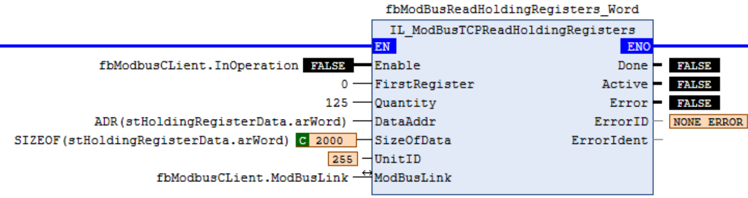
The modules only allow to send 125 elements at a time by means of the parameter "Quantity", if we have an area of 1000 words, for example, we must take into account that access should be done in eight blocks, modifying in this case the value assigned in " FirstRegister "

	Start	End
1	0	124
2	125	249
3	250	374
4	375	499
5	500	624
6	625	749
7	750	874
8	875	999

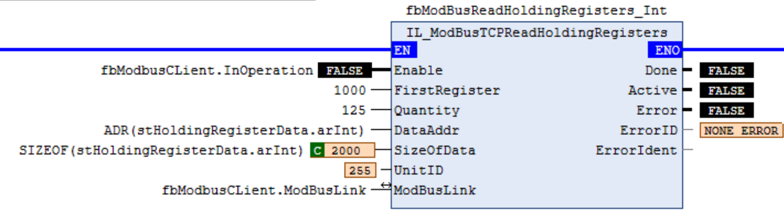


2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

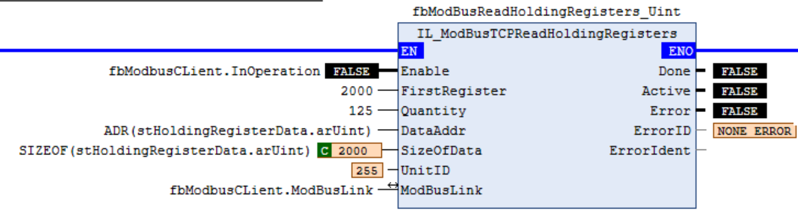
Modules used in the example in reading only the first 125 variables of each group



Word: area 0 - 999



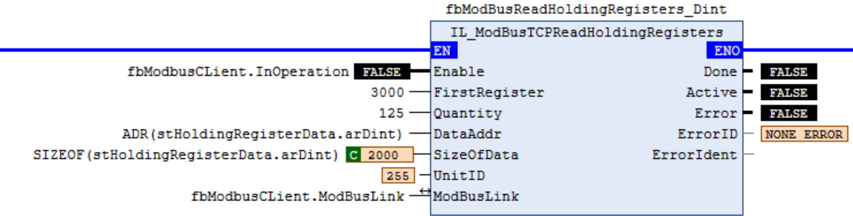
Int: area 1000 - 1999



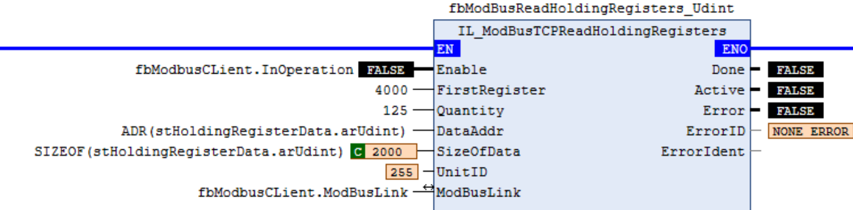
Uint: area 2000 - 2999

stHoldingRegisterData	dutRegisterData
arWord	ARRAY [0..999] OF WORD
arInt	ARRAY [0..999] OF INT
arUint	ARRAY [0..999] OF UINT
arDint	ARRAY [0..499] OF DINT
arUdint	ARRAY [0..499] OF UDINT
arReal	ARRAY [0..499] OF REAL
arString	ARRAY [0..99] OF STRING(19)

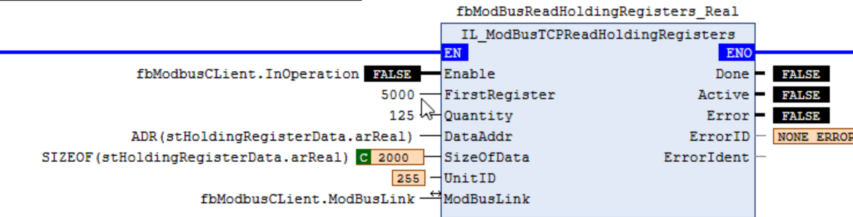
Dint: area 3000 - 3999



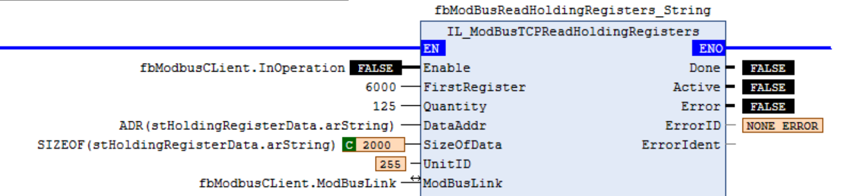
Dint: area 4000 - 4999



Real: area 5000 - 5999



Real: area 6000 - 6099



2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

ctrlX - Modules for writing CoilsData and HoldingRegisterData

The writing modules for multiple Coils or multiple registers allow us to modify the CoilsData and HoldingRegisterData values from the client side.

The operating structure for writing data is exactly the same as in the case of reading and we must modify the parameters "FirstRegister" and "Quantity" to be able to access the different areas

stHoldingRegisterData	dutRegisterData
arWord	ARRAY [0..999] OF WORD
arInt	ARRAY [0..999] OF INT
arUint	ARRAY [0..999] OF UINT
arDint	ARRAY [0..499] OF DINT
arUdint	ARRAY [0..499] OF UDINT
arReal	ARRAY [0..499] OF REAL
arString	ARRAY [0..99] OF STRING(19)

fbModBusWriteMultipleCoils
IL_ModBusTCPWriteMultipleCoils

Input	Value	Output	Value
EN		ENO	
Execute	bEnableWriteCoils FALSE	Done	FALSE
FirstCoil	0	Active	FALSE
Quantity	8	Error	FALSE
DataAddr	ADR(arCoilData)	ErrorID	NONE_ERROR
SizeOfData	SIZEOF(arCoilData) C 8	ErrorIdent	
UnitID	255		
ModBusLink	fbModbusCLient.ModBusLink		

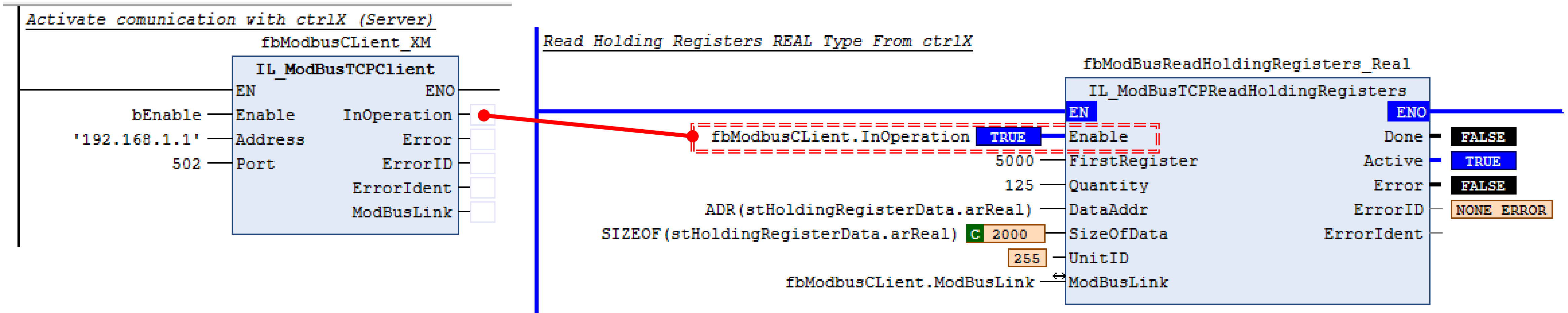
fbModbusWriteMultipleHoldingRegisters
IL_ModBusTCPWriteMultipleRegisters

Input	Value	Output	Value
EN		ENO	
Execute	bEnableWriteHoldingRegisters FALSE	Done	FALSE
FirstRegister	0	Active	FALSE
Quantity	125	Error	FALSE
DataAddr	ADR(stHoldingRegisterData.arReal)	ErrorID	NONE_ERROR
SizeOfData	SIZEOF(stHoldingRegisterData.arReal) C 2000	ErrorIdent	
UnitID	255		
ModBusLink	fbModbusCLient.ModBusLink		

2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

rexroth
A Bosch Company

In the previous examples the read modules are activated directly with the "InOperation" bit. However, it must be in mind that if we want to write data, this signal should be deactivated because if it does not continue writing the value sent by the Server



Notes:

- It is recommended to use a sequence of steps to perform a controlled sending of the data.

- Failure to do this may cause errors

- The communication must be associated to a standard Task and never to a Sercos task.

- The sequence used in the XM part manages the control of the reads and the error in the initial communication.

- This programming is done my way and it is obvious that the end goal can be achieved in many ways.

- This is just a small example of Modbus communication and some program changes may be necessary to get the best results.

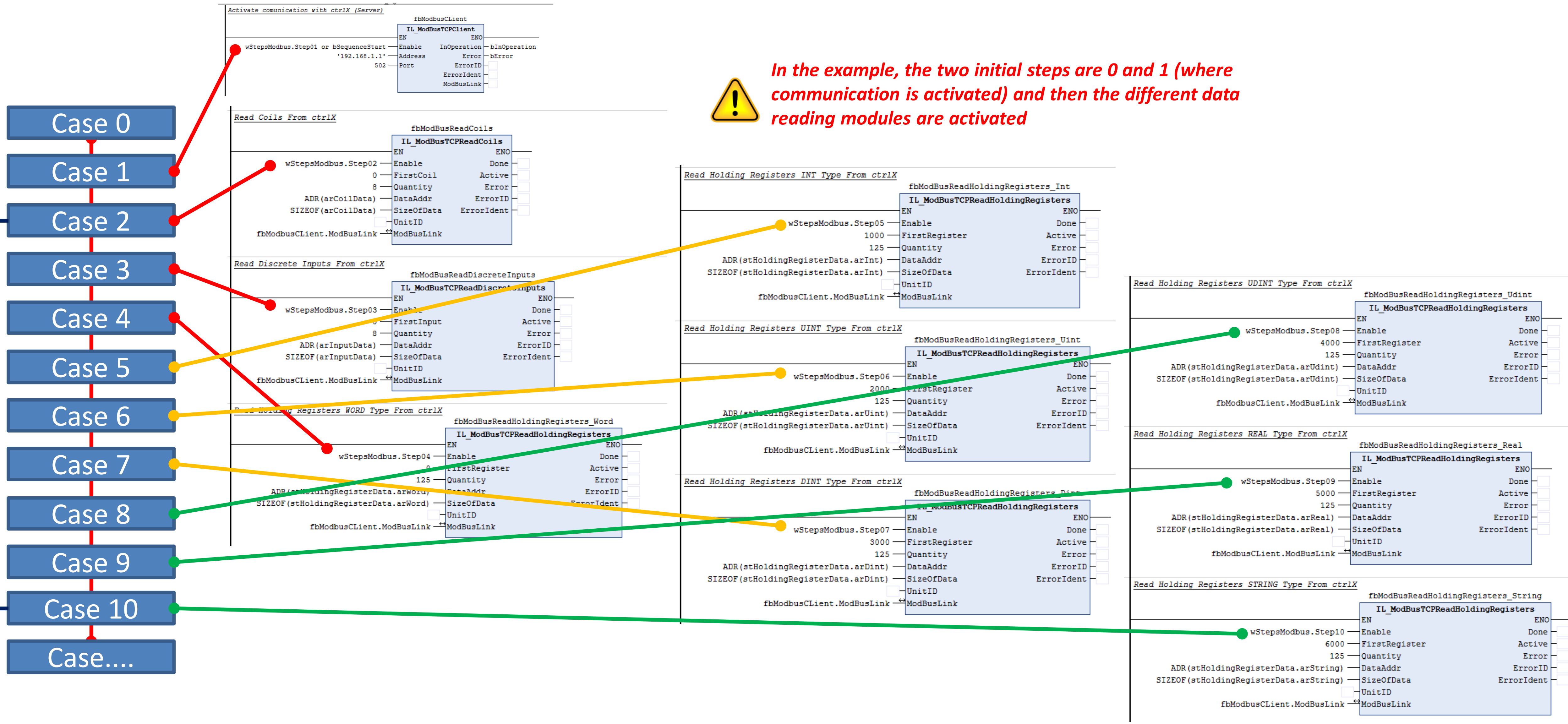
- As we have some areas for reading and writing and others for reading only, we can use them separately and in this way manage the data sent and the data received.

- In some modules the error F_Related_Table may appear, additional 1: "1817", additional 2: "13", this error does not appear in the lists but it can be generated by a too high value in "Quantity"

2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

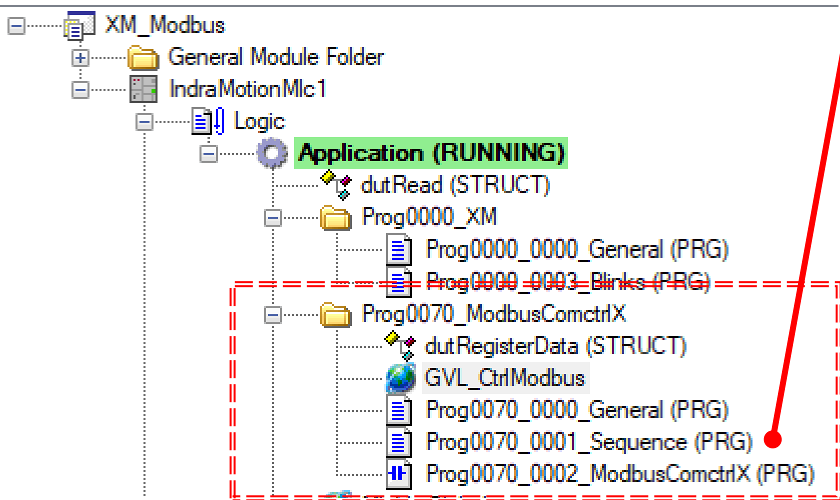
rexroth
A Bosch Company

ctrlX - *Example with Case sequence (With only reading modules)*



2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

Program modules used in the example



Modbus communication control modules (Example)

All steps are executed in the same way

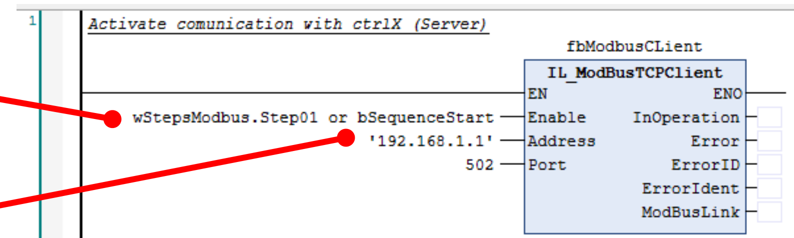
```
1  CASE iSecComModbus OF
2  0: // Initialization. Enable Communication Module
3  wStepsModbus:=wStepsModbusMask;
4  bSequenceStart:=FALSE;
5  iSecComModbus:=1;
6  1:  // Initialization. Enable Communication Module
7  wStepsModbus:=wStepsModbusMask;
8  IF NOT fbModbusCLient.InOperation AND NOT fbModbusCLient.Error THEN
9      wStepsModbus.Step01:=TRUE; // Enable
10 END_IF
11
12 IF fbModbusCLient.Enable AND fbModbusCLient.Error THEN
13     wStepsModbus.Step01:=FALSE; // Enable
14     iErrorStep:=1;
15     strErrorMessage:='Communication Not Start';
16 END_IF
17
18 IF fbModbusCLient.Enable AND fbModbusCLient.InOperation THEN
19     bSequenceStart:=TRUE;
20     iSecComModbus:=2;
21     iErrorStep:=0;
22     strErrorMessage:='';
23 END_IF
24
25 2: // Read Coils
26 wStepsModbus:=wStepsModbusMask;
27 wStepsModbus.Step02:=TRUE;
28 // Module error check
29 IF fbModBusReadCoils.Enable AND fbModBusReadCoils.Error THEN
30     iErrorStep:=2;
31     strErrorMessage:='Module Read Coils Fault';
32 END_IF
33 // If the module runs correctly, jump to the next step
34 IF fbModBusReadCoils.Enable AND fbModBusReadCoils.Done THEN
35     bSequenceStart:=TRUE;
36     iSecComModbus:=3;
37     iErrorStep:=0;
38     strErrorMessage:='';
39 END_IF
40 // Control of general communication for restart in case of failure
41 IF fbModbusCLient.Error THEN
42     iSecComModbus:=0;
43 END_IF
44 3: // Read Discrete Inputs
```

Reading module failure check

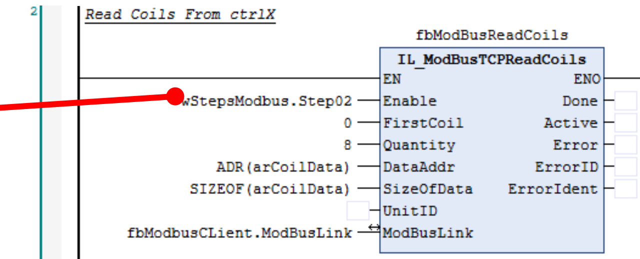
Control to jump to the next step.

Communication operation check. If there is an error, we jump to the boot step

Activate communication with ctrlX (Step 1)



Reading Coils from ctrlX (Step 2)



2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

rexroth
A Bosch Company

Graph of the Enable states. As can be seen there is "something" strange, since the fourth bit, corresponding to the values of type "Int" is practically always active.

Problem in the execution state of the enable of the "Int" type variables module. This even generates a "Delay" in the reception of signals.

2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

rexroth
A Bosch Company

ctrlX- Graph of the enable bits of the reading modules (Ok)

In the following graph you can see the operation of the sequence correctly

Configuration
Add variable

fbModBusReadCoils.Enable
0 | 0 | Δ0
fbModBusReadDiscreteInputs.Enable
1 | 1 | Δ0
GVL_CtrlModbus.fbModBusReadHoldingRegisters_Word.Enable
0 | 0 | Δ0
GVL_CtrlModbus.fbModBusReadHoldingRegisters_int.Enable
0 | 0 | Δ0
GVL_CtrlModbus.fbModBusReadHoldingRegisters_Uint.Enable
0 | 0 | Δ0
GVL_CtrlModbus.fbModBusReadHoldingRegisters_dint.Enable
0 | 0 | Δ0
GVL_CtrlModbus.fbModBusReadHoldingRegisters_Udint.Enable
0 | 0 | Δ0
GVL_CtrlModbus.fbModBusReadHoldingRegisters_Real.Enable
0 | 0 | Δ0
GVL_CtrlModbus.fbModBusReadHoldingRegisters_String.Enable
0 | 0 | Δ0

Fixed the sequence control, the steps are executed correctly

2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

rexroth
A Bosch Company

ctrlX - Video example with Case sequence (Send Values to XM)

```
fbModbusComToXM( // IL_ModbusTCPServerType02
    Enable FALSE :=bEnable FALSE ,
    InOperation FALSE => bInOperation FALSE,
    Error FALSE => bError FALSE,
    ErrorID=> ,
    ErrorIdent=> ,
    Port:= ,
    CoilData 16#0000007F4010A848 :=ADR(arCoilData) ,
    SizeOfCoilData 8 :=SIZEOF(arCoilData) ,
    InputData 16#0000007F4010A850 :=ADR(arInputData) ,
    SizeOfInputData 8 :=SIZEOF(arInputData) ,
    HoldingRegisterData 16#0000007F4010DF08 :=ADR(stHoldingRegisterData) ,
    SizeOfHoldingRegisterData 14000 :=SIZEOF(stHoldingRegisterData) ,
    InputRegisterData 16#0000007F4010A8 :=ADR(stRegisterData) ,
    SizeOfInputRegisterData 14000 :=SIZEOF(stRegisterData) ,
    Stats=> );

RETURN
```


2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

ctrlX - *Video example with Case sequence (Send Values to ctrlX)*

In general and if we structure the sending and receiving data, we can use the "RegisterData" as data from the ctrlX and use the HoldingRegisterData, which can also be written, as values to send to the crtlX

2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

Thanks you for your attention

2021/11/04| DCET / SFL4-ES | Jordi Laboria | Bosch Rexroth AG2021 All rights reserved, also regarding any disposal, explotacion, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.