Knihovna ModbusRTU

TXV 003 52.01
13. vydání
Listopad 2023
změny vyhrazeny

Historie změn

Datum	Vydání	Popis změn
Duben 2009	1	První vydání knihovny ModbusRTUlib_V10
Srpen 2009	2	Opraven rozsah *StAdr* (0000..9999) Doplněn odkaz na konstanty pro *chanCode* Přidán parametr *MBtimeOut* pro nastavení doby čekání na odpověď. nová verze knihovny ModbusRTUlib_V11
Únor 2010	3	Opraveno vypisování ErrCode přenášené z ComLib Opraveny kódy chybových hlášení v tabulce kap.2.5 nová verze knihovny ModbusRTUlib_V12
Srpen 2010	4	Přidány FB *ModbusTCPmas*, *ModbusCmdTCP* a struktura *TCmdStructTCP* Opraveno chování výstupu *Busy* u *fbModbusRTUmas* (pouze puls) Doplněny kódy chybových hlášení v tabulce kap.2.5 nová verze knihovny *ModbusRTUlib*_V13
Duben 2012	5	Se souhlasení příkladů s verzí knihovny ModbusRTUlib_V16
Říjen 2012	6	Opravy drobných chyb v textu
Září 2013	7	Přidány funkční bloky *fbModbusRtuSlave* a *fbModbusTcpSlave* a funkce *fcModbusTcpOff*, *fcModbusUdpOff* a *fcModbusTcpUdpOff* Revidovány příklady, opraven popis chování automatického režimu. Nastavení komunikačních kanálu přesunuto k popisu funkčních bloku, kterých se týkají.
Červenec 2015	8	Oprava popisu vstupu UnitID
Květen 2016	9	K verzi knihovny ModbusRTUlib_V31
Listopad 2017	10	Doplněno nastavení kanálu pro ModbusTCPmas Doplněn popis bloku fbModbusUDPmas
Srpen 2021	11	K verzi knihovny ModbusRTUlib_V39
Leden 2023	12	K verzi knihovny ModbusRTUlib_V42
Listopad 2023	13	K verzi knihovny ModbusRTUlib_V46 Přidány bloky *bModbusRtuSlaveOffset* a *fbModbusTcpSlaveOffset*

OBSAH

1 ÚVOD

Knihovny funkcí a funkčních bloků jsou nedílnou součástí instalace programovacího prostředí Mosaic. Z hlediska jejich výstavby je možné knihovny rozdělit na následující typy:

- vestavěné (built-in) knihovny
- standardně dodávané externí knihovny
- uživatelsky definované knihovny

Knihovna obsahuje deklarace funkcí, funkčních bloků, datových typů a globálních proměnných pro komunikaci Modbus master. Knihovna *ModbusRTUlib* používá některé funkce z knihoven *ComLib* a *CrcLib*.

Knihovna je dodávaná jako součást instalace prostředí Mosaic od verze v 2.0.25.0. Funkce a funkční bloky knihovny *ModbusRTUlib* jsou podporovány v centrálních jednotkách řady K (TC700 CP-7004 a všechny varianty systému Foxtrot) od verze v 5.7. a vyšší.

Katalogové číslo dokumentace ke knihovně *ModbusRTUlib* je TXV 003 52.01.

1.1 Protokol Modbus

Modbus RTU je otevřený sériový protokol zveřejněný firmou Modicon již v roce 1979. Modbus standard je definován jako aplikační vrstva v 7 úrovňovém OSI modelu, která vykonává „Client/Server“ komunikaci mezi dvěma zařízeními spojenými různými typy sběrnic, nebo komunikačními sítěmi. Na sériových linkách protokol provádí výměnu dat mezi jednou stanicí „master“ a jednou nebo více stanicemi „slave“ v jednom okamžiku. To znamená, že jedna stanice řídí komunikaci a ostatní stanice odpovídají na dotazy.

Varianta pro Ethernet se nazývá Modbus TCP. Má v sítích Ethernet vyhrazené své číslo portu 502. Princip přenosu dat a a komunikační funkce jsou stejné jako u Modbus RTU. V sítích Ethernet může komunikace probíhat současně i mezi více klienty a více servery najednou.

Knihovna *ModbusRTUlib* umožňuje realizovat komunikaci v obou režimech, tedy PLC Tecomat v roli serveru (slave) nebo klienta (master).

Na ethernetu je Modbus slave standardně realizován pomocí výchozí ovladače, který tímto protokolem zpřístupňuje zápisník PLC. Při použití knihovny je tento ovladač vypnut a jeho funkci přebírá funkční blok. Podobně je Modbus slave řešen na sériových kanálech, kde je dostupný jako jeden z režimů (viz TXV 004 03 – Sériová komunikace programovatelných automatů Tecomat - Model 32 bitů).

Výhody realizace Modbus slave pomocí knihovny jsou větší možnosti konfigurace komunikačních parametrů, více informací o stavu komunikace a možnost určení velikostí a umístnění paměťových oblastí protokolem přístupných.

Nevýhodou je větší režie při zpracování zpráv a nároky na paměť kódu a uživatelské registry.

Popisem protokolu Modbus se podrobně zabývají dokumenty na webových stránkách http://www.modbus.org/tech.php.

Názvosloví odpovídá specifikaci http://modbus.org/docs/PI_MBUS_300.pdf

1.2 Kódování dat v paměti

Modbus používá k ukládání dat v paměťovém prostoru model používaný firmou Motorola, který se nazývá „Big Endian“. Čísla uložená v paměti ve více bytech, např 16 bitové typu WORD je uloženo v paměti tak, že byte s vyššími řády leží na nižší adrese a byte s nižšími řády na vyšší adrese. A v tomto pořadí se odesílají datové části jednotlivých telegramů

Hodnota 0x1234 je uložena nejdříve 0x12, potom 0x34
Hodnota 0x5678 je uložena nejdříve 0x56, potom 0x78

adresa	data
0	0x12
1	0x34
2	0x56
3	0x78

Naproti tomu PLC Tecomat používá kódování v paměti podle firmy Intel, které se nazývá „Little Endian“. Čísla uložená v paměti ve více bytech, např 16 bitové typu WORD je uloženo v paměti tak, že byte s nižšími řády leží na nižší adrese a byte s vyššími řády na vyšší adrese.

Hodnota 0x1234 je uložena nejdříve 0x34, potom 0x12
Hodnota 0x5678 je uložena nejdříve 0x78, potom 0x56

adresa	data
0	0x34
1	0x12
2	0x78
3	0x56

1.3 Funkce protokolu Modbus

Tab.1 Seznam podporovaných funkcí protokolu MODBUS

Kód	Funkce	Popis
01	Read Coil Status	čtení výstupů (paměť 0X)
02	Read Input Status	čtení vstupů (paměť 1X)
03	Read Holding Registers	čtení registrů (paměť 4X)
04	Read Input Registers	čtení vstupních registrů (paměť 3X)
05	Force Single Coil	nastavení jednoho výstupu (paměť 0X)
06	Preset Single Register	nastavení jednoho registru (paměť 4X)
07	Read Exception Status *)	informace o stavu automatu
08	Diagnostics *)	diagnostické funkce
15	Force Multiple Coils	nastavování výstupů (paměť 0X)
16	Preset Multiple Registers	nastavování holding registrů (paměť 4X)
17	Report Slave ID *)	vrací identifikační číslo serveru
22	Mask Write	nastavení registru pomocí masky (paměť 4X)

*) Funkce aplikované pouze pro sériové linky. Tyto funkce nejsou podporovány blokem fb-ModbusRtuSlave

Hodnoty kódů jsou v knihovně definovány jako následující konstanty:

Proměnná	***Typ***	***Význam***
MDB_FNC_ReadCoilStatus	USINT	čtení výstupů (paměť 0X)
MDB_FNC_ReadInputStatus	USINT	čtení vstupů (paměť 1X)
MDB_FNC_ReadHoldingRegisters	USINT	čtení registrů (paměť 4X)
MDB_FNC_ReadInputRegisters	USINT	čtení vstupních registrů (paměť 3X)
MDB_FNC_ForceSingleCoil	USINT	nastavení jednoho výstupu (paměť 0X)
MDB_FNC_PresetSingleRegister	USINT	nastavení jednoho registru (paměť 4X)
MDB_FNC_ReadExceptionStatus	USINT	informace o stavu automatu
MDB_FNC_Diagnostics	USINT	diagnostické funkce
MDB_FNC_ForceMultipleCoils	USINT	nastavování výstupů (paměť 0X)
MDB_FNC_PresetMultipleRegisters	USINT	nastavování holding registrů (paměť 4X)
MDB_FNC_ReportSlaveID	USINT	vrací identifikační číslo serveru
MDB_FNC_MaskWrite	USINT	nastavení registru pomocí masky (paměť 4X)

1.4 Datový model Modbus

Objekty	typ objektu	přístup	typicky poskytuje	oblast
Cívky	pole jednotlivých bitů	Čtení i zápis	aplikační program	0xxxx
Diskrétní vstupy	pole jednotlivých bitů	Pouze čtení	I/O systém	1xxxx
Vstupní registry	pole 16-bitových wordů	Pouze čtení	I/O systém	3xxxx
Vnitřní registry	pole 16-bitových wordů	Čtení i zápis	aplikační program	4xxxx

Číslování objektů v systémech MODICON začíná číslem jedna, ale v telegramech protokolu se přenáší adresy uvnitř stanice od 0 (viz parametr *StAdr*). Objekty v oblastech 0xxxx a 1xxxx jsou číslovány jako jednotlivé po sobě jdoucí bity. Pokud se zadává počet přenášených objektů, jedná se tedy o počet přenášených bitů. Objekty v oblastech 3xxxx a 4xxxx jsou číslovány jako 16-ti bitová slova a počet přenášených objektů je tedy počtem po sobě jdoucích slov typu word (viz parametr *NoPoint*).

2 FUNKCE A FUNKČNÍ BLOKY PRO MODBUS MASTER

Knihovna *ModbusRTULib* obsahuje následující funkční bloky a funkce:

- ModbusCmd — funkce pro sestavení příkazu popisujícího jednu komunikaci
- ModbusRTUmas — FB provádějící komunikace podle příkazů ModbusCmd po sériové lince
- fbModbusRTUmas2 — FB provádějící komunikace podle příkazů ModbusCmd po sériové lince řízenou prodlevou
- ModbusCmdTCP — funkce pro sestavení příkazu popisujícího jednu komunikaci po ethernetu
- ModbusTCPmas — FB provádějící komunikace podle příkazů ModbusCmdTCP po ethernetu protokolem TCP
- fbModbusTCPmas2 — FB provádějící komunikace podle příkazů ModbusCmdTCP po ethernetu protokolem TCP s řízenou prodlevou
- fbModbusUDPmas — FB provádějící komunikace podle příkazů ModbusCmdTCP po ethernetu protokolem UDP s řízenou prodlevou

Režim Modbus lze realizovat pomocí této knihovny na kanálech a spojeních, které jsou nastaveny v režimu UNI.

Funkční blok *ModbusRTUmas* a *fbModbusRTUmas2* je určen pro komunikaci sériovým kanálem a parametry jsou dány v nastavovacím dialogu sériového kanálu (CH1_uni, ..., CH10_uni).

Pro připojení více zařízení s různými IP adresami protokolem Modbus TCP je určen funkční blok *ModbusTCPmas a ModbusTCPmas2,* kde IP adresa koncových slave zařízení je zadávána v každém příkazu samostatně a jejich počet není omezen. IP adresy se dynamicky mění za chodu programu. Pro vyšší rychlost odbavení příkazů je možné rozdělit příkazy na více spojení, tím že FB spustíme vícekrát paralelně pokaždé s jiným spojením (UNI ETH1_uni0, ..., ETH1_uni7). První řídící příkaz otevře TCP spojení a provede relaci. Pokud jsou příkazy na stejnou IP adresu, spojení se nezavírá. Při přechodu na jinou IP adresu se nejdříve spojení uzavře a potom se otevře spojeni s novou IP adresou. Při ztrátě spojení například při poruše přenosové trasy nebo vypnutí koncového zařízení je třeba počítat, že uzavření TCP spojení může trvat až několik sekund z důvodu timeout časů v TCP protokolu.

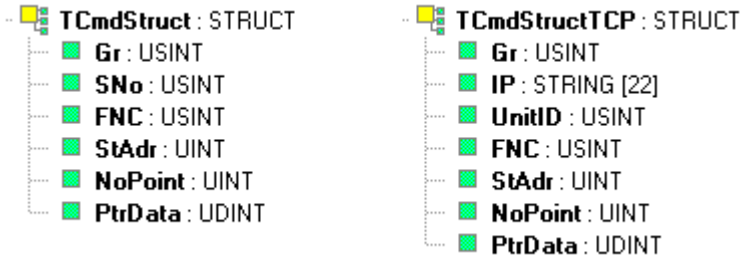
Funkční blok *ModbusUDPmas* umožňuje komunikaci s více zařízení s různými IP adresami obdobně jako *ModbusTCPmas,* ale místo TCP protokolu je použit protokol UDP.

UDP protokol není zatížen režií na navazování a ukončování spojení, tudíž komunikace s více podřízenými stanicemi bude rychlejší.

2.1 Popis řízení komunikace Modbus master

V aplikaci je typicky třeba vyměňovat data mezi více zařízeními a z různých míst uvnitř těchto zařízení. K tomu účelu je potřeba tyto výměny popsat ve formě pole příkazů.

Příkaz popisující komunikaci má následující strukturu *TCmdStruct* pro FB *ModbusRTUmas* nebo *TCmdStructTCP* pro FB *ModbusTCPmas* :



Proměnná	***Význam***
Gr	číslo skupiny příkazů pro řízení v automatickém módu nebo manuálním modu (1..255)
Sno	adresa slave stanice od 1 do 247; 0 je rezervována pro broadcast; >247 je rezervována.
IP	IP adresa (a případně port) slave zařízení (např. '192.168.1.1' nebo '192.168.1.1:512')
UnitID	Identifikátor slave zařízení (využívá se pro komunikaci se slave zařízeními na sériové lince přes Modbus TCP bránu)
FNC	kód funkce Modbus (viz tab.1)
StAdr	počáteční adresa objektů uvnitř stanice Modbus mínus jedna (0000..9999). Např: pro holding registr 3 bude StAdr = 2. (Toto je číslo, které je také přenášeno v telegramu.)
NoPoint	Počet datových objektů , které budou čteny nebo zapsány (type BOOL 1 ... 2000 nebo WORD 1 ... 125)
PtrData	ukazatel na počátek pole proměnných v PLC Tecomat přenášených tímto příkazem

Každý příkaz popisuje jednu komunikační funkci, odkud a kam se data budou přenášet a kolik se jich přenese.

Podle hodnoty nastavené v parametru *GrSel* se příkazy vykonávají v manuálním režimu jednotlivě, nebo v automatickém režimu cyklicky, podle následujících pravidel:

Příkazy s číslem *Gr* = 1 jsou v automatickém módu vždy vykonávány cyklicky.

Příkazy s číslem *Gr* = 255 jsou v automatickém módu vykonávány jen jednou v inicializaci.

Příkazy s jinými hodnotami *Gr* mohou být do cyklického režimu zařazeny podmíněně. Při *GrSel* > 1 jsou cyklicky vykonávány všechny příkazy, kde *Gr* = 1 a *Gr* = *GrSel*.

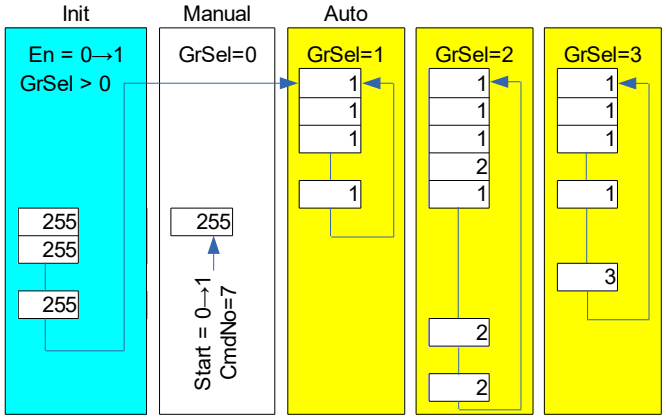
Příkaz s číslem *Gr* = 0 jsou vždy přeskočeny

Příklad funkce

Tabulka příkazů

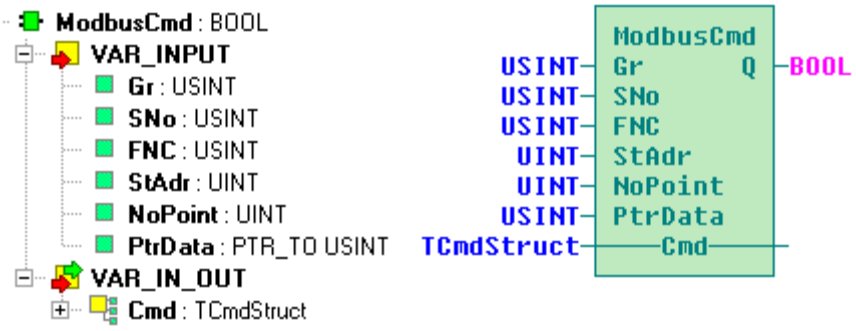
n	Gr	SNo	FNC	StAdr	NoPoint	PtrData
[1]	1					
[2]	1					
[3]	1					
[4]	2					
[5]	1					
[6]	255					
[7]	255					
[8]	3					
[9]	255					
[10]	2					
[11]	0					
[12]	2					

MaxCmd=12



2.2 Funkce ModbusCmd

Knihovna: *ModbusRTUlib*



Funkce nastaví parametry komunikace do pole příkazů, kterými se pak řídí komunikace se zařízeními Modbus slave. Tato funkce slouží pro nastavení příkazů pro funkční blok *ModbusRTUmas a fbModbusRTUmas2*. Funkce vrací *TRUE* pokud je *Gr* různé od nula

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	Gr	USINT	Číslo skupiny příkazů pro automatický nebo manuální režim (1..255)
	SNo	USINT	Adresa slave zařízení 1 až 247; Nula je určena pro broadcast a větší než čísla 247 jsou rezervována pro jiné použití.
	FNC	USINT	Číslo příkazu Modbus protokolu 01,02,03,04,05,06,15,16,17
	StAdr	UINT	Počáteční adresa Modbus objektu 0..65535 (Coil, Inputs, Input registers, Holding registers)
	NoPoint	UINT	Počet objektů (BOOL nebo WORD), které budou čteny nebo zapisovány
	PtrData	PTR_TO USINT	Ukazatel na počátek pole proměnných v PLC Tecomat přenášených tímto příkazem **!! Pozor !! Je-li FCN rovno 1, 2 nebo 5 musí pointer ukazovat na typ BOOL nebo pole typu BOOL**
VAR_IN_OUT			
	Cmd	TCmdStruct	Výsledný příkaz pro řízení jedné komunikace, který bude funkcí nastaven
ModbusCmd			
	Návratová hodnota	BOOL	TRUE pokud je *Gr* různé od nula

Příklad volání funkcí v jazyku ST pro inicializaci dvou příkazů pro komunikační kanál CH2:

```
TYPE
  TDataBlock1 : STRUCT //teploty
    Temperature1 : REAL;
    Temperature2 : REAL;
  END_STRUCT;

  TDataBlock2 : STRUCT //tlaky
    Pressure1 : REAL;
    Pressure2 : REAL;
    Pressure3 : REAL;
  END_STRUCT;
END_TYPE

VAR_GLOBAL
  Temps    : TDataBlock1;
  Press    : TDataBlock2;
  Flags    : ARRAY [1..5] OF BOOL;        // pole bitových příznaků
  CmdCH2   : ARRAY [1..3] OF TCmdStruct; (* pole příkazů Modbus
                                            pro řízení kanálu CH2 *)
END_VAR

PROGRAM prgMain
  ModbusCmd(Gr:=1, FNC:=03, SNo:=1,
            StAdr:=0, NoPoint:=SIZEOF(Temps)/2,
            PtrData:=adr(Temps), Cmd:=CmdCH2[1]);
  ModbusCmd(Gr:=1, FNC:=03, SNo:=2,
            StAdr:=4, NoPoint:=SIZEOF(Press)/2,
            PtrData:=adr(Press), Cmd:=CmdCH2[2]);
  ModbusCmd(Gr:=1, FNC:=01, SNo:=1,
            StAdr:=9, NoPoint:=5,
            PtrData:=adr(Flags), Cmd:=CmdCH2[3]);
END_PROGRAM
```

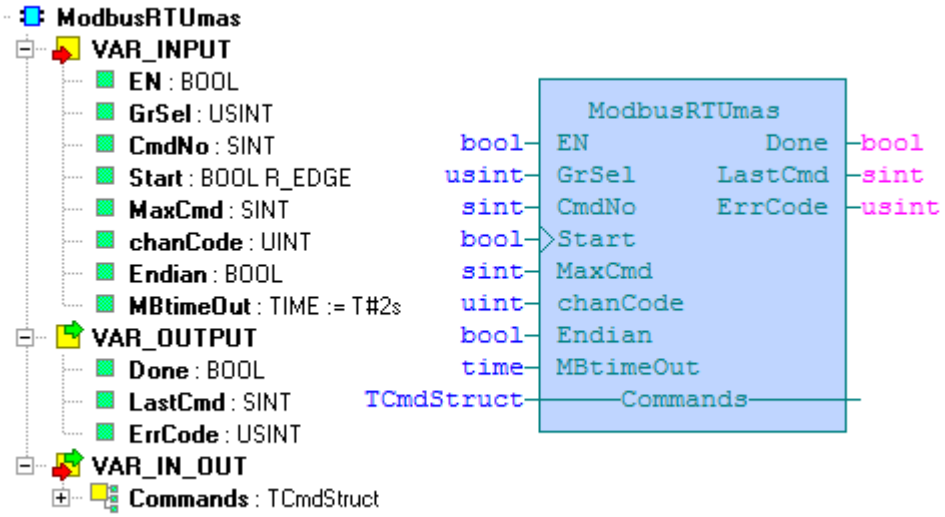
První příkaz bude funkcí *03 Read Holding Registers* vyčítat ze stanice s adresou 1, počínaje prvním objektem, čtyři wordy do proměnných *Temps*. Druhý příkaz bude funkcí *03 Read Holding Registers* vyčítat ze stanice s adresou 2, počínaje pátým objektem, šest wordů do proměnných *Press*. Třetí příkaz bude funkcí 01 *Read Coil Status* vyčítat ze stanice s adresou 1, počínaje desátým objektem, pět bitových příznaků a uloží je do pole *Flags*.

Všechny příkazy se zapíší do pole struktur *CmdCH2*.

Funkce *SIZEOF* vrací velikost proměnné v bytech, proto je výsledek dělen dvěma, aby se získala velikost ve wordech.

2.3 Funkční blok ModbusRTUmas

Knihovna: *ModbusRTUlib*



Funkční blok *ModbusRTUmas* sestavuje komunikační relace podle pole připravených řídících příkazů a prostřednictvím zvoleného komunikačního kanálu vyměňuje data mezi PLC master a připojenými zařízeními typu Modbus RTU slave.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	EN	BOOL	Povolení činnosti funkčního bloku
	GrSel	USINT	Výběr režimu komunikace. Nastavuje způsob vysílání zpráv, které jsou definovány v poli příkazů typu *TCmdStruct*. *GrSel* = 0 Manuální režim. Na náběžnou hranu vstupu Start bude odeslán příkaz, jehož pořadí v poli příkazů udává vstup *CmdNo*. *GrSel* = 1 Automatický režim. Jsou odesílány pouze příkazy s *Gr* = 1 definované v poli příkazů. Po odeslání všech příkazů s *Gr* = 1 z pole příkazů se odesílání cyklicky opakuje. *GrSel* > 1 Automatický režim. Jsou odesílány příkazy s *Gr* = 1 a *Gr* = *GrSel* definované v poli příkazů. Po odeslání všech příkazů s *Gr* = 1 a *Gr* = *GrSel* z pole příkazů se odesílání cyklicky opakuje. *GrSel* = 255 Nepoužívat. Příkazy s *Gr* = 255 definované v poli příkazů jsou určeny k inicializaci slave zařízení. Tyto příkazy se pošlou právě jednou ihned po nastavení vstupu *EN*, pokud je *GrSel* = 1,...,254 (automatický režim)

	Proměnná	Typ	Význam
	CmdNo	SINT	Číslo příkazu, který bude odeslán z pole příkazů v manuálním režimu. V automatickém režimu bude tento vstup ignorován.
	Start	BOOL R_EDGE	V manuálním režimu se na náběžnou hranu tohoto vstupu odešle příkaz daný vstupem CmdNo. V automatickém režimu bude tento vstup ignorován.
	MaxCmd	SINT	Celkový počet příkazů, které jsou prohledávány v řídicím poli příkazů
	chanCode	UINT	Komunikační kanál, kterým probíhá komunikace (CH1_uni, ..., CH10_uni)
	Endian	BOOL	Uložení wordových registrů v komunikaci; 0 - BigEndian; 1 - LittleEndian
	MBtimeOut	TIME	Modbus timeout (implicitně = 2 [sec])
VAR_OUTPUT			
	Done	BOOL	Komunikační příkaz byl úspěšně vykonán (pulz délky 1 cyklu PLC)
	LastCmd	SINT	Číslo naposledy vykonaného příkazu 0=žádný, [1..MaxRec]
	ErrCode	USINT	Chybový kód
VAR_IN_OUT			
	Commands	TCmdStruct	Pole příkazů pro řízení komunikace Modbus

Manuální mód (GrSel = 0) :

FB ModbusRTUmas zpracovává jednorázově příkaz daný číslem *CmdNo* na náběžnou hranu signálu *Start,* je-li proměnná *En* nastavena na hodnotu *true* .

Automatický mód (GrSel > 0)

Vždy po nastavení proměnné *En* na hodnotu *true* se v prvním cyklu vykonávají pouze příkazy s Gr = 255. Tyto povely slouží k případné inicializaci slave zařízení.
V následujících cyklech se vždy vykonávají příkazy, které mají Gr = 1 a Gr = GrSel a to v pořadí v jakém jsou zapsány v poli příkazů pro řízení komunikace Modbus. (Viz obr. v kap. 2.1)

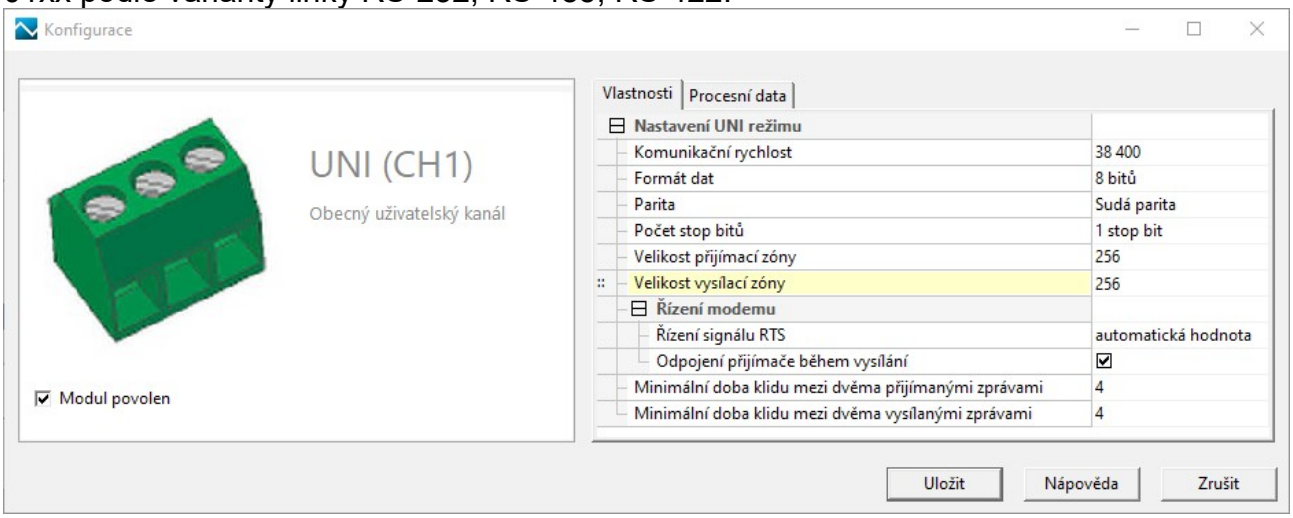
Parametry *CmdNo* a *LastCmd:* Nesou pořadové číslo příkazu (1...MaxCmd) v tabulce příkazů *Commands* nezávisle na tom je-li pole příkazů definované od nuly ***array[0..MaxCmd-1] of TCmdStruct*** nebo od jedné ***array [1..MaxCmd] of TCmdStruct***. Deklarace od jedné je vhodnější, protože se pak číslo shoduje s indexem pole příkazů.

Parametr *chanCode:* V PLC Tecomat jsou sériové komunikační kanály označovány jako CH1_UNI,... CH10_UNI. Mohou být osazeny různými rozhraními RS-232, RS-485, RS-422 apod. Jména CH1_UNI,... CH10_UNI jsou jména konstant typu UINT definovaných v knihovně ComLib.

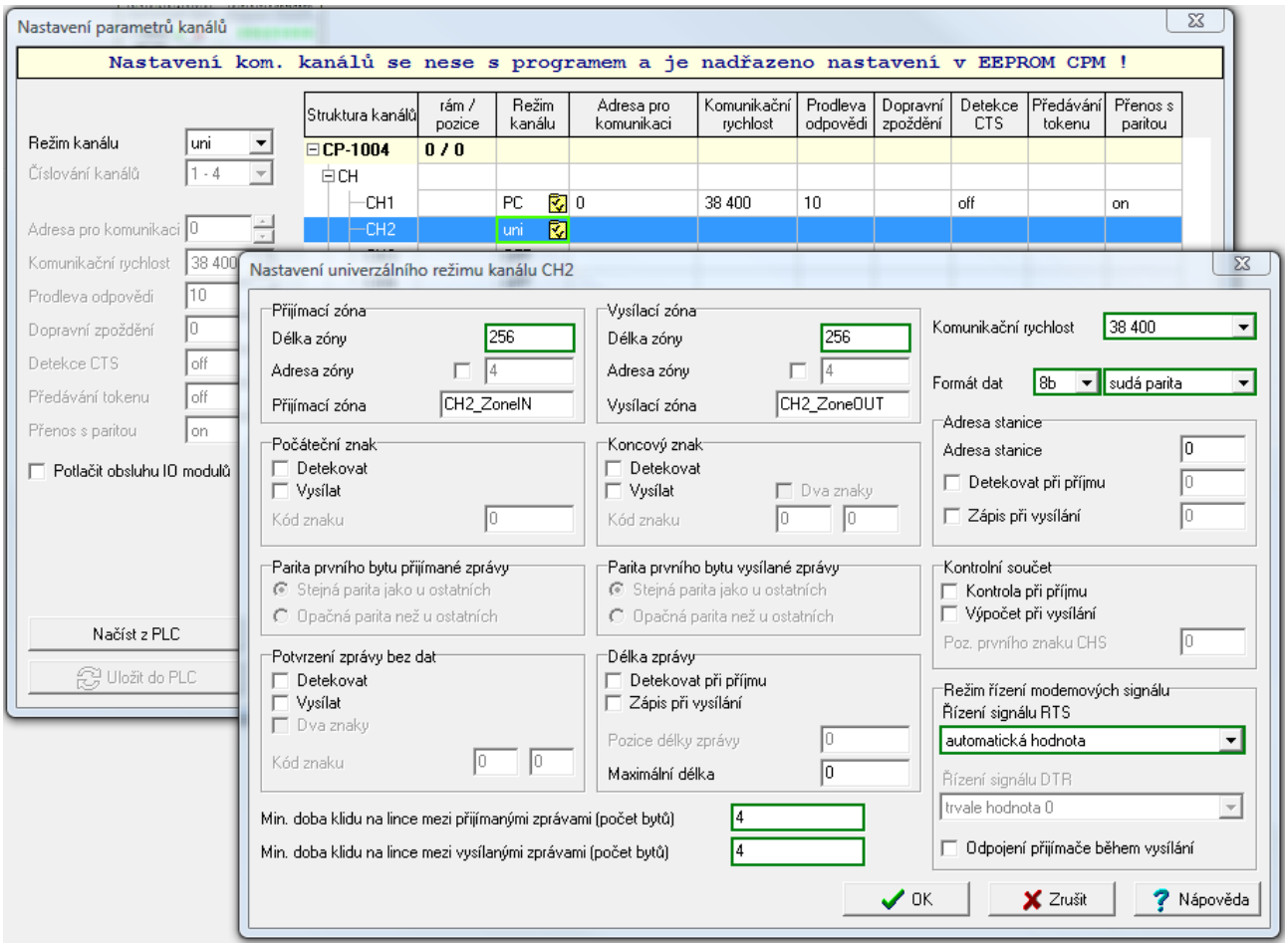
Zvolený komunikační kanál je nutné nastavit do režimu UNI. Délka přijímací a vysílací zóny musí být nastavena na **256** bytů a minimální doby klidu na lince nastavit na 4 znaky. Podle podřízených zařízení nastavit komunikační rychlost, formát dat a paritu.

Pracuje-li podřízené zařízení „bez parity" musí se nastavit v dialogu „parita trvale 1", aby byla splněna podmínka dvou stop bitů.

Pokud není použit kanál s pevných rozhraním, musí být osazen submodulem MR-01xx podle varianty linky RS-232, RS-485, RS-422.



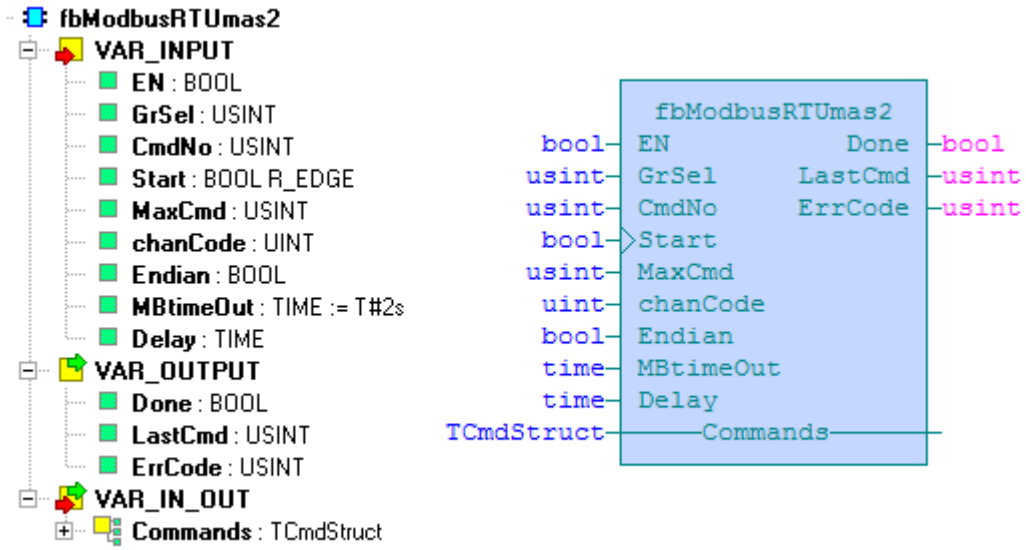
Příklad nastavení kanálu CH1 pro blok ModbusRTUmas v nástroji I/O Configurátor



Příklad nastavení kanálu CH2 pro blok ModbusRTUmas

2.4 Funkční blok fbModbusRTUmas2

Knihovna: *ModbusRTUlib*



fbModbusRTUmas2 je variantou bloku *ModbusRTUmas*, který poskytuje možnost nastavit prodlevu mezi jednotlivými dotazy na vstupu *Delay* a zvyšuje maximální počet příkazů až na 255.

Navýšení počtu příkazů si vyžádalo změnu typu vstupů *CmdNo* a *MaxCmd* z typu SINT na typ USINT.

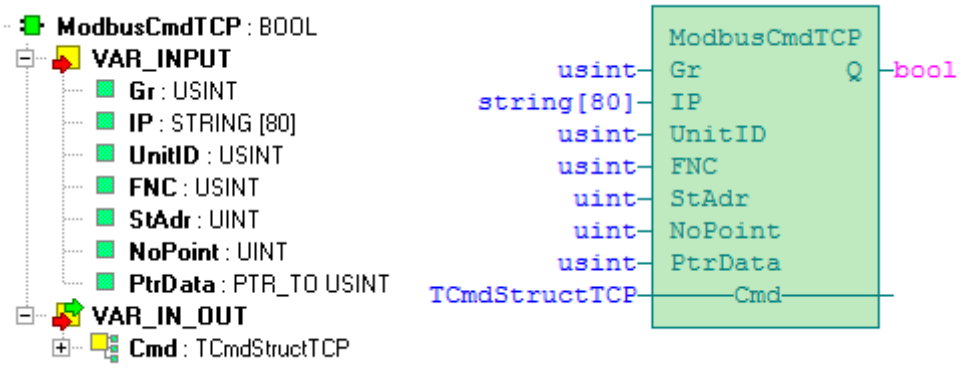
Všechny ostatní vstupy, výstupy a nastavení jsou shodného typu a významu jako u původního bloku *ModbusRTUmas*.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	EN	BOOL	Povolení činnosti funkčního bloku
	GrSel	USINT	Nastavuje způsob vysílání zpráv, které jsou definovány v poli příkazů typu *TCmdStruct*. *GrSel* = 0 Manuální režim. Na náběžnou hranu vstupu Start bude odeslán příkaz, jehož pořadí v poli příkazů udává vstup *CmdNo*. *GrSel* = 1 Automatický režim. Jsou odesílány pouze příkazy s *Gr* = 1 definované v poli příkazů. Po odeslání všech příkazů s *Gr* = 1 z pole příkazů se odesílání cyklicky opakuje. *GrSel* > 1 Automatický režim. Jsou odesílány příkazy s *Gr* = 1 a *Gr* = *GrSel* definované v poli příkazů. Po odeslání všech příkazů s *Gr* = 1 a *Gr* = *GrSel* z pole příkazů se odesílání cyklicky opakuje. *GrSel* = 255 Nepoužívat. Příkazy s *Gr* = 255 definované v poli příkazů jsou určeny k inicializaci slave zařízení. Tyto příkazy se pošlou právě jednou ihned po nastavení vstupu *EN*, pokud je *GrSel* = 1,...,254 (automatický režim)
	CmdNo	USINT	Číslo příkazu, který bude odeslán z pole příkazů v manuálním režimu. V automatickém režimu bude tento vstup ignorován.
	Start	BOOL R_EDGE	V manuálním režimu se na náběžnou hranu tohoto vstupu odešle příkaz daný vstupem CmdNo. V automatickém režimu bude tento vstup ignorován.
	MaxCmd	USINT	Celkový počet příkazů, které jsou prohledávány v řídicím poli příkazů
	chanCode	UINT	Komunikační kanál, kterým probíhá komunikace (CH1_uni, ..., CH10_uni)
	Endian	BOOL	Uložení wordových registrů v komunikaci; 0 - BigEndian; 1 - LittleEndian
	MBtimeOut	TIME	Modbus timeout (implicitně = 2 [sec])
	Delay	TIME	Prodleva mezi přijatou zprávou a novou zprávou
VAR_OUTPUT			
	Done	BOOL	Komunikační příkaz byl úspěšně vykonán (pulz délky 1 cyklu PLC)
	LastCmd	USINT	Číslo naposledy vykonaného příkazu 0=žádný, [1..MaxRec]
	ErrCode	USINT	Chybový kód
VAR_IN_OUT			
	Commands	TCmdStruct	Pole příkazů pro řízení komunikace Modbus

2.5 Funkce ModbusCmdTCP

Knihovna: *ModbusRTUlib*



Funkce nastaví parametry komunikace do pole příkazů, kterými se pak řídí komunikace se zařízeními Modbus slave. Tato funkce slouží pro nastavení příkazů pro funkční blok *ModbusTCPmas*, *fbModbusTCPmas2* a *fbModbusUDPmas*. Funkce vrací *TRUE* pokud je *Gr* různé od nula

Popis proměnných :

	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	Gr	USINT	Číslo skupiny příkazů pro automatický nebo manuální režim (1..255)
	IP	STRING[80]	IP adresa (a případně port) slave zařízení (např. '192.168.1.1' nebo '192.168.1.1:512')
	UnitID	USINT	Identifikátor slave zařízení (využívá se pro komunikaci se slave zařízeními na sériové lince přes Modbus TCP bránu)
	FNC	USINT	Číslo příkazu Modbus protokolu 01,02,03,04,05,06,15,16
	StAdr	UINT	Adresa Modbus objektu 0000..65535 (Coil,InpBits,InpRegs,HoldRegs)
	NoPoint	UINT	Počet objektů (BOOL nebo WORD), které budou čteny nebo zapisovány
	PtrData	PTR_TO USINT	Ukazatel na počátek pole proměnných v PLC Tecomat přenášených tímto příkazem **!! Pozor !! Je-li FCN rovno 1, 2 nebo 5 musí pointer ukazovat na typ BOOL nebo pole typu BOOL**
VAR_IN_OUT			
	Cmd	TCmdStructTCP	příkaz, který bude nastaven
ModbusCmdTCP			
	Návratová hodnota	BOOL	TRUE pokud je Gr různé od nula

IP adresu zapisujeme jako STRING v jednoduchých uvozovkách, čtyři čísla oddělená tečkou bez mezer. Je-li třeba posílat rámce na jiný než implicitní port (502), lze číslo portu napsat na konci stringu IP adresy za dvojtečku. Příklady zápisu: '192.168.33.2' nebo '192.168.033.002:3215'.

Příklad volání funkcí v jazyku ST pro inicializaci dvou příkazů pro TCP spojení:

```
TYPE
  TDataBlock1 : STRUCT //teploty
    Temperature1 : REAL;
    Temperature2 : REAL;
  END_STRUCT;

  TDataBlock2 : STRUCT //tlaky
    Pressure1 : REAL;
    Pressure2 : REAL;
    Pressure3 : REAL;
  END_STRUCT;
END_TYPE

VAR_GLOBAL
  Temps   : TDataBlock1;
  Press   : TDataBlock2;
  Flags   : ARRAY [1..5] OF BOOL;       // pole bitových příznaků
  CmdTCP1 : ARRAY [1..3] OF TCmdStructTCP; // pole příkazů Modbus
END_VAR

PROGRAM prgMain
  ModbusCmdTCP(Gr:=1, FNC:=03, UnitID:=0, IP:='192.168.33.2',
               StAdr:=0, NoPoint:=SIZEOF(Temps)/2,
               PtrData:=adr(Temps), Cmd:=CmdTCP1[1]);
  ModbusCmdTCP(Gr:=1, FNC:=03, UnitID:=0, IP:='192.168.33.2',
               StAdr:=4, NoPoint:=SIZEOF(Press)/2,
               PtrData:=adr(Press), Cmd:=CmdTCP1[2]);
  ModbusCmdTCP(Gr:=1, FNC:=01, UnitID:=0, IP:='192.168.33.161:3215',
               StAdr:=9, NoPoint:=5,
               PtrData:=adr(Flags), Cmd:=CmdTCP1[3]);
END_PROGRAM
```

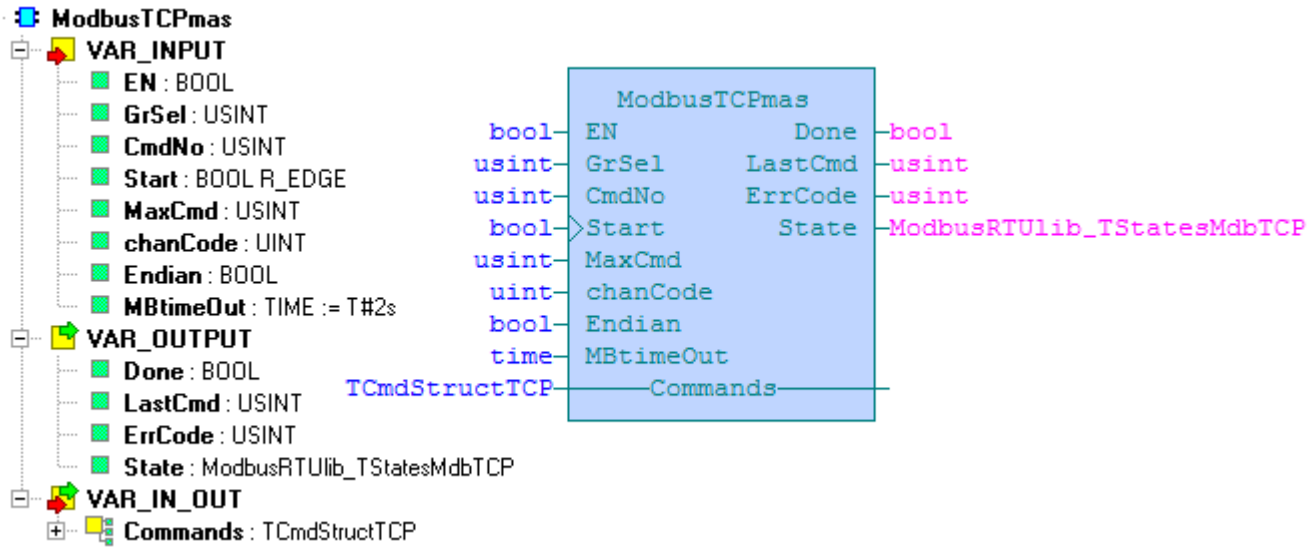
První příkaz bude funkcí *03 Read Holding Registers* vyčítat ze stanice s IP adresou 192.168.33.2, počínaje prvním objektem, čtyři wordy do proměnných *Temps*. Druhý příkaz bude funkcí *03 Read Holding Registers* vyčítat ze stanice s toutéž adresou, počínaje pátým objektem, šest wordů do proměnných *Press*. Třetí příkaz bude funkcí *01 Read Coil Status* vyčítat ze stanice s IP adresou 192.168.33.161 poslouchající na portu 3215, počínaje desátým objektem, pět bitových příznaků a uloží je do pole *Flags.*

Všechny příkazy se zapíší do pole struktur *CmdTCP1*.

Funkce *SIZEOF* vrací velikost proměnné v bytech, proto je výsledek dělen dvěma, aby se získala velikost ve wordech.

2.6 Funkční blok ModbusTCPmas

Knihovna: ModbusRTUlib



Sestavuje komunikační relace podle pole připravených řídících příkazů a prostřednictvím zvoleného TCP spojení vyměňuje data mezi PLC master a připojenými zařízeními typu Modbus slave.

Popis proměnných :

	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	EN	BOOL	Povolení činnosti funkčního bloku
	GrSel	USINT	Výběr režimu komunikace. Nastavuje způsob vysílání zpráv, které jsou definovány v poli příkazů typu *TCmdStructTCP*. *GrSel* = 0 Manuální režim. Na náběžnou hranu vstupu Start bude odeslán příkaz, jehož pořadí v poli příkazů udává vstup *CmdNo*. *GrSel* = 1 Automatický režim. Jsou odesílány pouze příkazy s *Gr* = 1 definované v poli příkazů typu *TCmdStructTCP*. Po odeslání všech příkazů s *Gr* = 1 z pole příkazů se odesílání cyklicky opakuje. *GrSel* > 1 Automatický režim. Jsou odesílány příkazy s *Gr* = 1 a *Gr* = *GrSel* definované v poli příkazů typu *TCmdStructTCP*. Po odeslání všech příkazů s *Gr* = 1 a *Gr* = *GrSel* z pole příkazů se odesílání cyklicky opakuje. *GrSel* = 255 Nepoužívat. Příkazy s Gr = 255 definované v poli příkazů typu TCmdStructTCP jsou určeny k inicializaci slave zařízení. Tyto příkazy se pošlou právě jednou ihned po nastavení vstupu EN, pokud je GrSel = 1,...,254 (automatický režim)

	Proměnná	Typ	Význam
	CmdNo	USINT	Číslo příkazu, který bude odeslán z pole příkazů v manuálním režimu. V automatickém režimu bude tento vstup ignorován.
	Start	BOOL R_EDGE	V manuálním režimu se na náběžnou hranu tohoto vstupu odešle příkaz daný vstupem CmdNo. V automatickém režimu bude tento vstup ignorován.
	MaxCmd	USINT	Celkový počet příkazů v poli příkazů
	chanCode	UINT	Číslo komunikačního kanálu (ETH1_uni0, ..., ETH1_uni7)
	Endian	BOOL	Uložení wordových registrů v komunikaci; 0 - BigEndian; 1 - LittleEndian
	MBtimeOut	TIME	Modbus timeout (implicitně=2[sec])
VAR_OUTPUT			
	Done	BOOL	Příkaz byl zpracován (pulz délky 1 cyklu PLC)
	LastCmd	USINT	Pořadové číslo naposledy vykonaného příkazu 0=žádný,[1..MaxRec]
	ErrCode	USINT	Chybový kód
	State	ModbusRTUlib_ TStatesMdbTCP	Stav komunikace
VAR_IN_OUT			
	Commands	TCmdStructTCP	Pole příkazů pro řízení komunikace

Manuální mód (GrSel = 0) :

FB *ModbusTCPmas* zpracovává jednorázově příkaz daný číslem *CmdNo* na náběžnou hranu signálu *Start,* je-li proměnná *En* nastavena na hodnotu *true* .

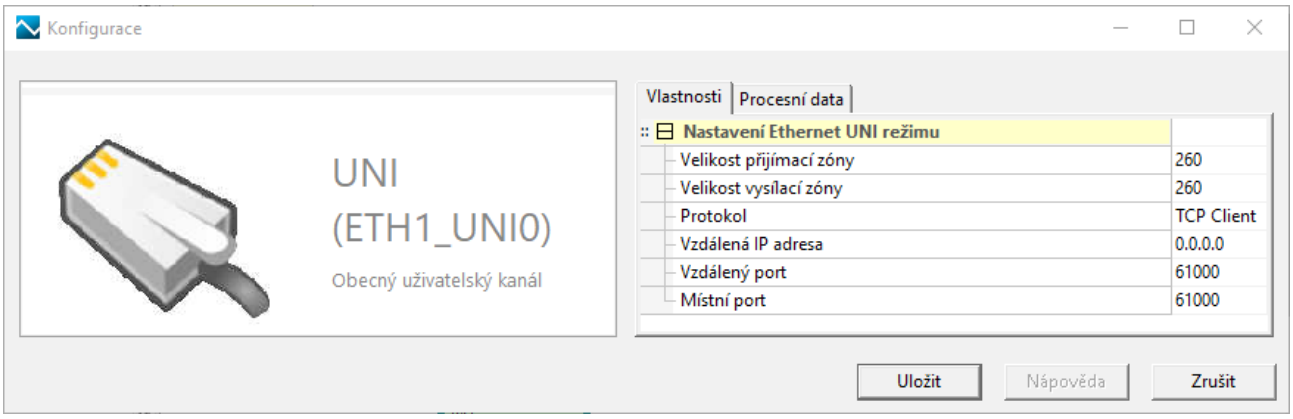
Automatický mód (GrSel > 0)

Vždy po nastavení proměnné *En* na hodnotu *true* se v prvním cyklu vykonávají pouze příkazy s Gr = 255. Tyto povely slouží k případné inicializaci slave zařízení.
V následujících cyklech se vždy vykonávají příkazy, které mají Gr = 1 a Gr = GrSel a to v pořadí v jakém jsou zapsány v poli příkazů pro řízení komunikace Modbus. (Viz obr. v kap. 2.1)

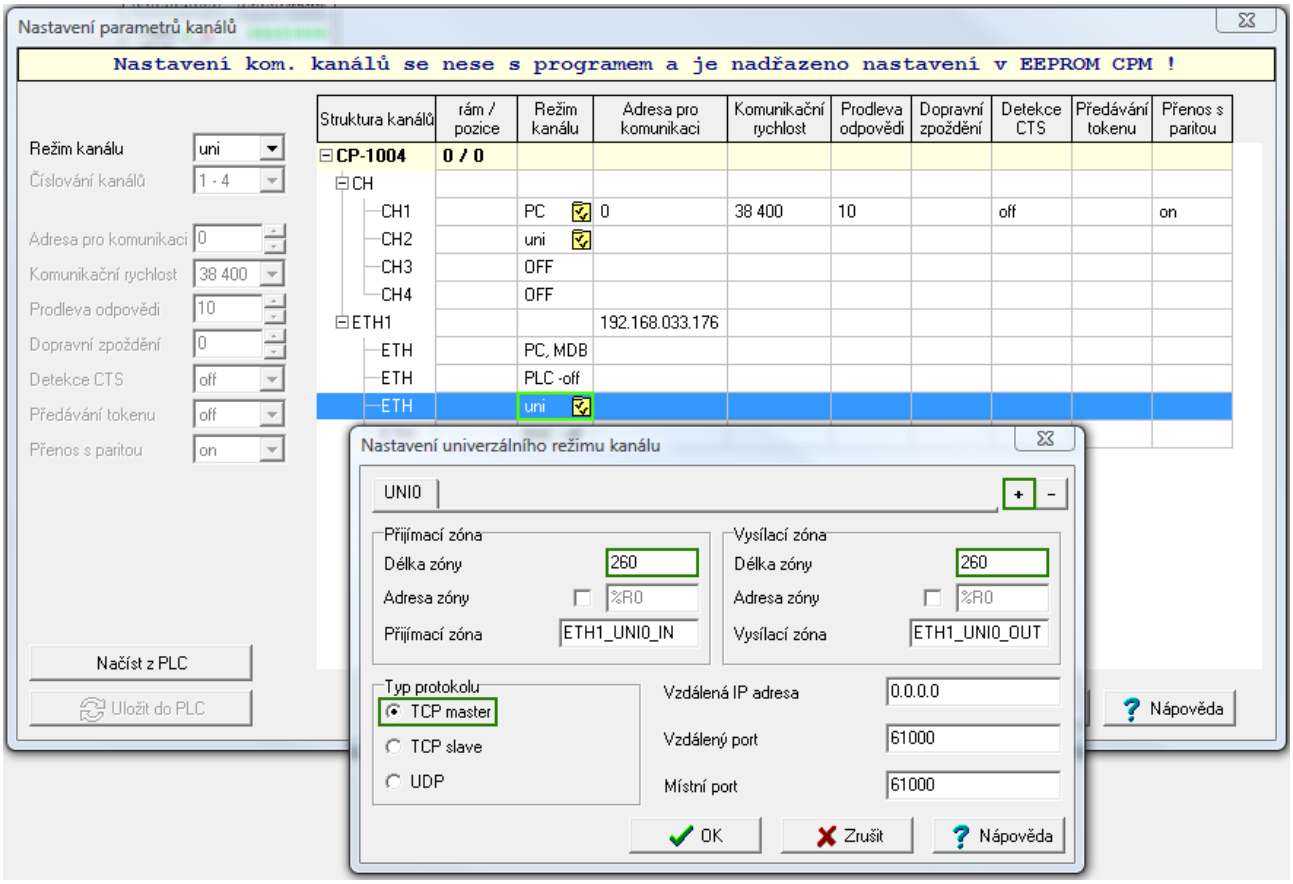
Parametry *CmdNo* a *LastCmd:* Nesou pořadové číslo příkazu (1...MaxCmd) v tabulce příkazů *Commands* nezávisle na tom je-li pole příkazů definované od nuly ***array [0..MaxCmd-1] of TCmdStructTCP*** nebo od jedné ***array [1..MaxCmd] of TCmdStructTCP***. Deklarace od jedné je vhodnější, protože se pak číslo shoduje s indexem pole příkazů.

Parametr *chanCode:* V PLC Tecomat jsou univerzální spojení na Ethernetu (sokety) označovány ETH1_UNI0,... ETH4_UNI7. Jména ETH1_UNI0,... ETH4_UNI7 jsou jména konstant typu UINT definovaných v knihovně ComLib.

Zvolené spojení musí být v režimu UNI s typem protokolu TCP master. Délka přijímací a vysílací zóny musí být nastavena na **260** bytů. Vzdálená IP adresa a port mohou být zde nastaveny na libovolnou hodnotu, protože budou doplněny vždy s každým novým příkazem.



Příklad nastavení spojení UNI0 pro blok ModbusTCPmas v nástroji I/O Configurator



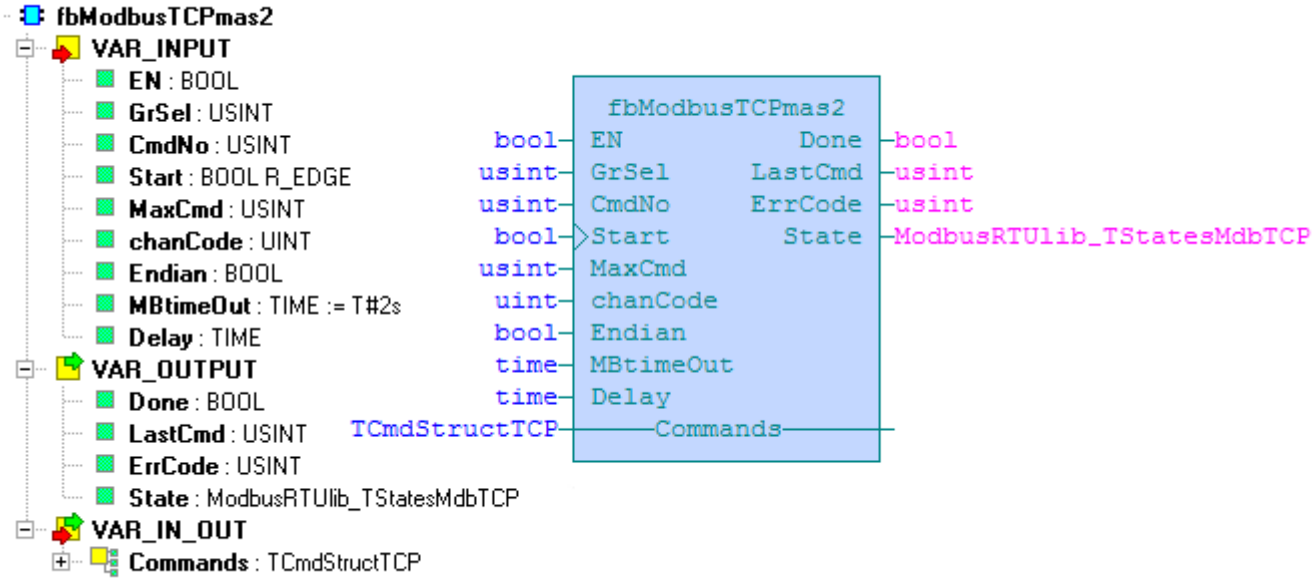
Příklad nastavení spojení UNI0 pro blok ModbusTCPmas v HW konfiguraci (další spojení mohou být přidána tlačítkem +)

ModbusRTUlib_TStatesMdbTCP – Stavy komunikace funkčního bloku *ModbusTCPmas*

Hodnota		***Význam***
0	*MdbTCP_init*	Inicializace
1	*MdbTCP_start*	Čekání na Start v manuální režimu, výběr dalšího příkazu v automatickém režimu
2	*MdbTCP_selCommand*	Výběr příkazu nastaveného automatickým nebo manuálním režimem
3	*MdbTCP_setIPadr*	Nastavení IP adresy podle vybraného příkazu
4	*MdbTCP_estabCon*	Navázání spojení s vybranou IP adresou
5	*MdbTCP_sendData*	Vysílání příkazu
6	*MdbTCP_recData*	Příjem odpovědi podřízené stanice
7	*MdbTCP_error*	Zavírání spojení po detekci chyby

2.7 Funkční blok fbModbusTCPmas2

Knihovna: ModbusRTUlib



fbModbusTCPmas2 je variantou bloku *ModbusTCPmas*, který poskytuje možnost nastavit prodlevu mezi jednotlivými dotazy na vstupu *Delay*.

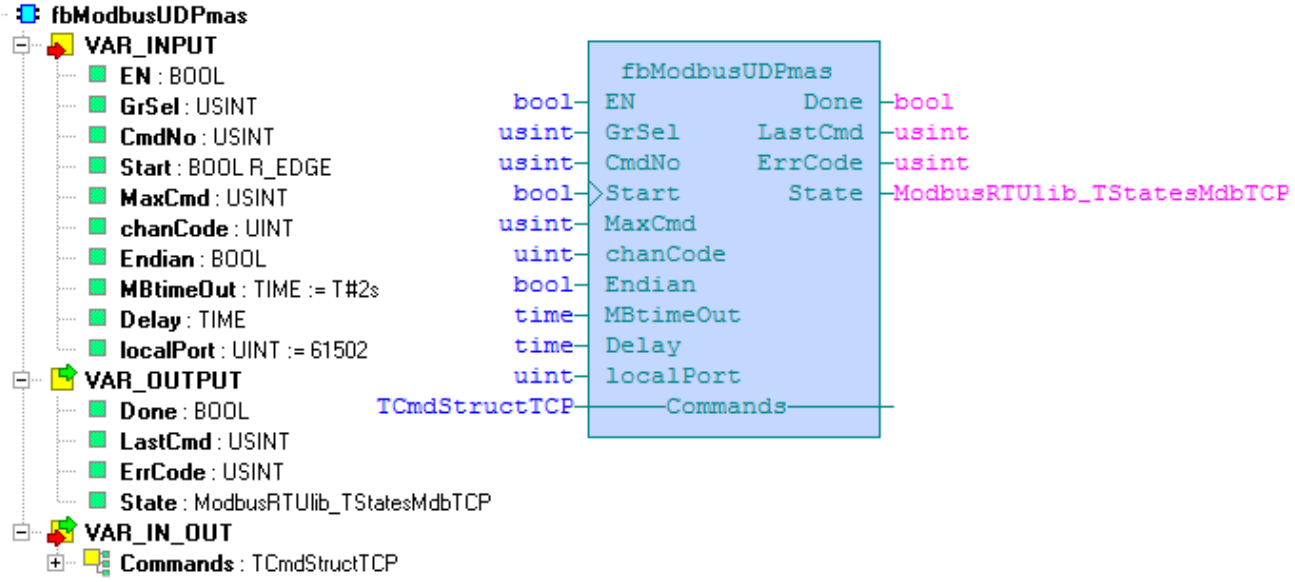
Všechny ostatní vstupy, výstupy a nastavení jsou shodného typu a významu jako u původního bloku *ModbusTCPmas*.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	EN	BOOL	Povolení činnosti funkčního bloku
	GrSel	USINT	Výběr režimu komunikace. Nastavuje způsob vysílání zpráv, které jsou definovány v poli příkazů typu *TCmdStructTCP*.
	CmdNo	USINT	Číslo příkazu, který bude odeslán z pole příkazů v manuálním režimu. V automatickém režimu bude tento vstup ignorován.
	Start	BOOL R_EDGE	V manuálním režimu se na náběžnou hranu tohoto vstupu odešle příkaz daný vstupem CmdNo. V automatickém režimu bude tento vstup ignorován.
	MaxCmd	USINT	Celkový počet příkazů v poli příkazů
	chanCode	UINT	Číslo komunikačního kanálu (ETH1_uni0, ..., ETH1_uni7)
	Endian	BOOL	Uložení wordových registrů v komunikaci; 0 - BigEndian; 1 - LittleEndian
	MBtimeOut	TIME	Modbus timeout (implicitně=2[sec])
	Delay	TIME	Prodleva mezi přijatou zprávou a novou zprávou
VAR_OUTPUT			
	Done	BOOL	Příkaz byl zpracován (pulz délky 1 cyklu PLC)
	LastCmd	USINT	Pořadové číslo naposledy vykonaného příkazu 0=žádný,[1..MaxRec]
	ErrCode	USINT	Chybový kód
	State	ModbusRTUlib_ TStatesMdbTCP	Stav komunikace
VAR_IN_OUT			
	Commands	TCmdStructTCP	Pole příkazů pro řízení komunikace

2.8 Funkční blok fbModbusUDPmas

Knihovna: ModbusRTUlib



fbModbusUDPmas sestavuje komunikační relace podle pole připravených řídících příkazů a prostřednictvím zvoleného UDP spojení vyměňuje data mezi PLC master a připojenými zařízeními typu Modbus slave.

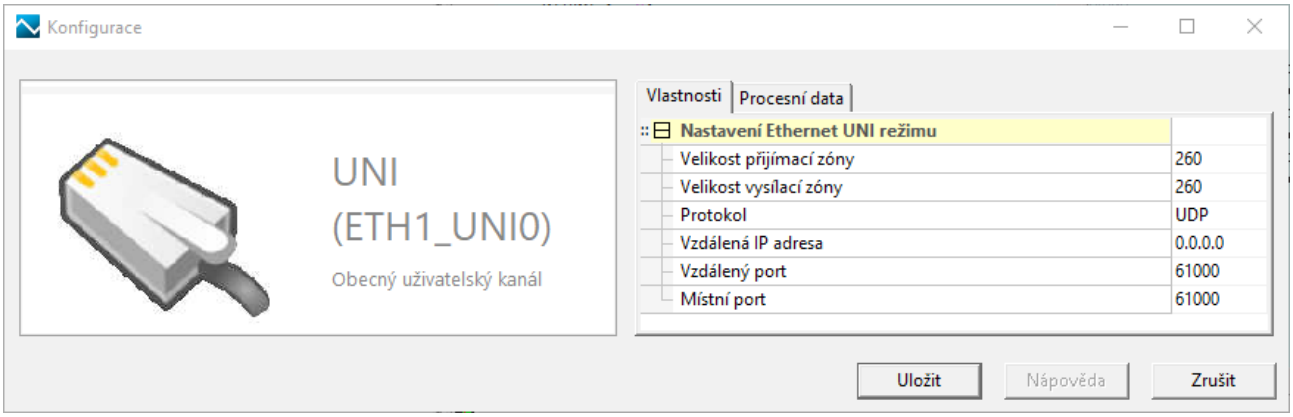
Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	EN	BOOL	Povolení činnosti funkčního bloku
	GrSel	USINT	Výběr režimu komunikace. Nastavuje způsob vysílání zpráv, které jsou definovány v poli příkazů typu *TCmdStructTCP*.
	CmdNo	USINT	Číslo příkazu, který bude odeslán z pole příkazů v manuálním režimu. V automatickém režimu bude tento vstup ignorován.
	Start	BOOL R_EDGE	V manuálním režimu se na náběžnou hranu tohoto vstupu odešle příkaz daný vstupem *CmdNo*. V automatickém režimu bude tento vstup ignorován.
	MaxCmd	USINT	Celkový počet příkazů v poli příkazů
	chanCode	UINT	Číslo komunikačního kanálu (ETH1_uni0, ..., ETH1_uni7)
	Endian	BOOL	Uložení wordových registrů v komunikaci; 0 - BigEndian; 1 - LittleEndian
	MBtimeOut	TIME	Modbus timeout (implicitně=2[sec])
	Delay	TIME	Prodleva mezi přijatou zprávou a novou zprávou
	localPort	UINT	Místní port pro odesílání zpráv, každá instance musí mít jiný

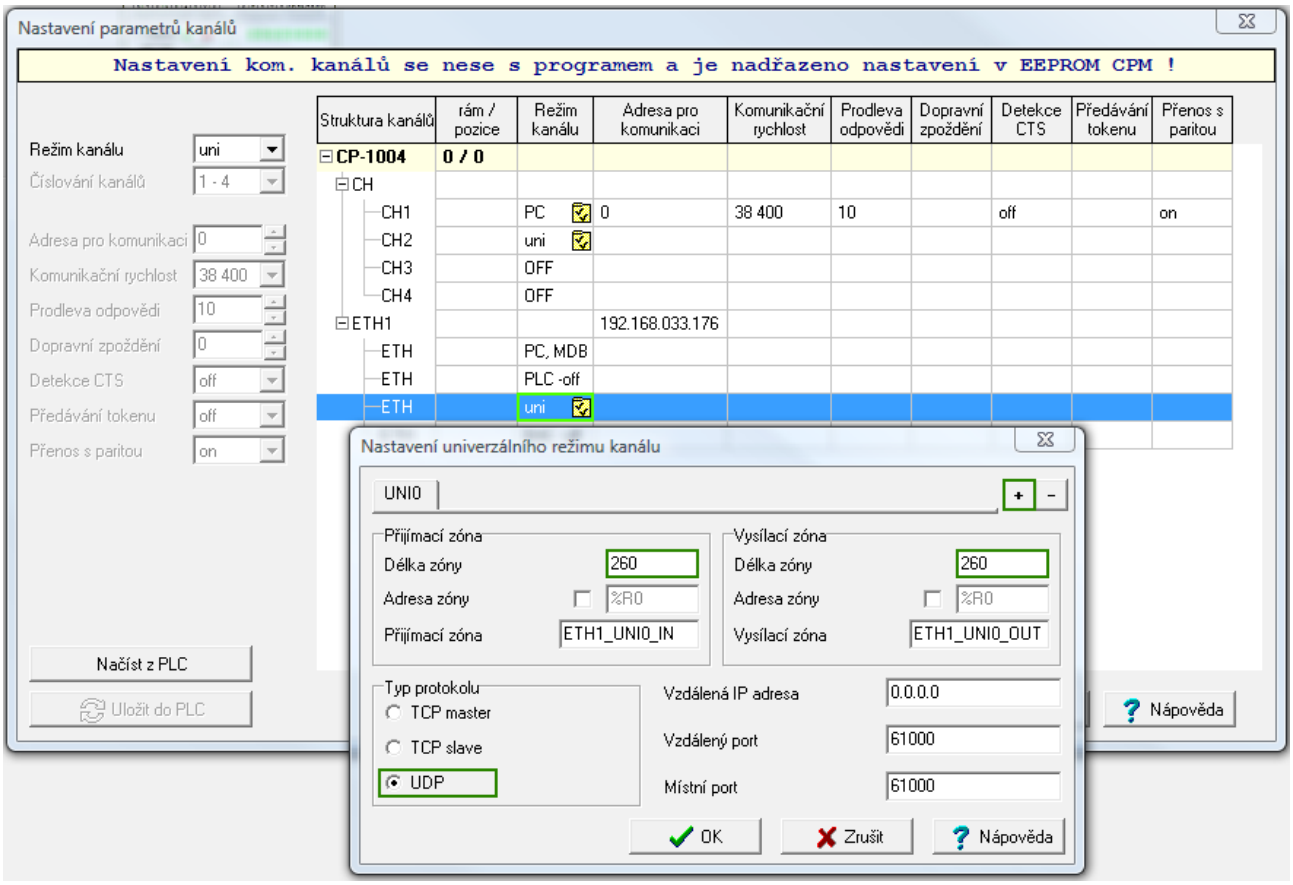
	Proměnná	Typ	Význam
VAR_OUTPUT			
	Done	BOOL	Příkaz byl zpracován (pulz délky 1 cyklu PLC)
	LastCmd	USINT	Pořadové číslo naposledy vykonaného příkazu 0=žádný,[1..MaxRec]
	ErrCode	USINT	Chybový kód
	State	ModbusRTUlib_TStatesMdbTCP	Stav komunikace
VAR_IN_OUT			
	Commands	TCmdStructTCP	Pole příkazů pro řízení komunikace

Ovládání bloku a pracovní stavy jsou shodné jako u bloku *ModbusTCPmas*. Jediným rozdílem je nastavení ethernetového spojení.

Zvolené spojení musí být v režimu UNI s typem protokolu UDP. Délka přijímací a vysílací zóny musí být nastavena na **260** bytů. Vzdálená IP adresa a port mohou být zde nastaveny na libovolnou hodnotu, protože budou doplněny vždy s každým novým příkazem. Zatím co IP adresa vzdálený port se nese v příkazu, místní port se definuje na vstupu *localPort*. Pro více instancí funkčního bloku je nutné aby hodnoty *localPort* byly unikátní.



Příklad nastavení spojení UNI0 pro blok *fbModbusUDPmas* v nástroji I/O Configurator



Příklad nastavení spojení UNI0 pro blok fbModbusUDPmas
(další spojení mohou být přidána tlačítkem +)

3 FUNKCE A FUNKČNÍ BLOKY PRO MODBUS SLAVE

Knihovna *ModbusRTULib* obsahuje následující funkční bloky a funkce:

fbModbusRTUslave	Funkční blok realizující funkci Modbus serveru na sériové lince
fbModbusRTUslaveOffset	Funkční blok realizující funkci Modbus serveru na sériové lince
fbModbusTCPslave	Funkční blok realizující funkci Modbus serveru na Ethernetu
fbModbusTCPslaveOffset	Funkční blok realizující funkci Modbus serveru na Ethernetu
fcModbusUdpOff	Funkce pro vypnutí integrovaného driveru na UDP port 502
fcModbusTcpOff	Funkce pro vypnutí integrovaného driveru na TCP port 502
fcModbusTcpUdpOff	Funkce pro vypnutí integrovaného driveru na TCP i UDP

Režim Modbus lze realizovat pomocí této knihovny na kanálech a spojeních, které jsou nastaveny v režimu UNI.

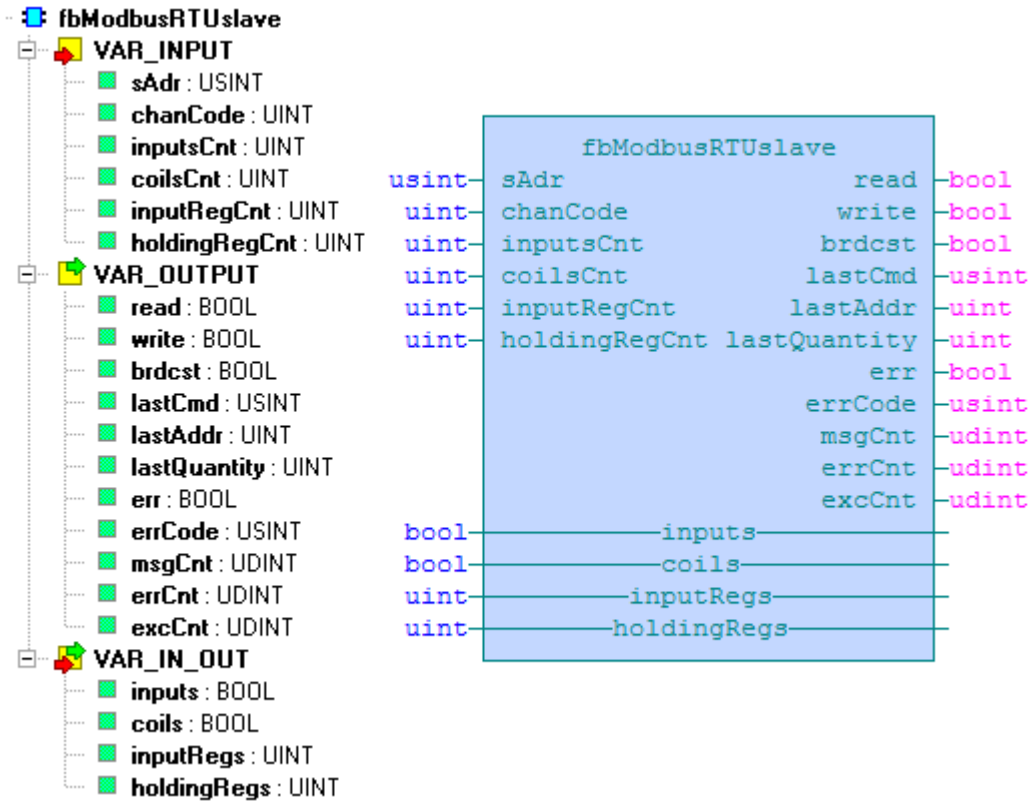
Funkční blok *fbModbusRTUslave* je určen pro komunikaci sériovým kanálem a parametry jsou dány v nastavovacím dialogu sériového kanálu (CH1_uni, ..., CH10_uni).

Komunikace po Ethernetu řeší blok *fbModbusTCPslave*. Pokud je potřeba paralelní připojení více klientů je možné založit více instancí bloku z nichž každá bude pracovat na svém spojení (UNI ETH1_uni0, ..., ETH1_uni7).

Při použití bloku *fbModbusTCPslave* je automaticky deaktivován integrovaný TCP driver jako při zavolání funkce *fcModbusTcpOff*.

3.1 Funkční blok fbModbusRTUslave

Knihovna: *ModbusRTUlib*



Funkční blok *fbModbusRTUslave* interpretuje příkazy Modbus RTU na sériovém kanálu specifikovaných vstupem *chanCode* cílené na adresu danou vstupem *sAdr* a na příkazy s broadcastovou adresou 0. Podporované příkazy viz. Kap. 1.3 Tab. 1.

Datové bloky jsou definovány vstupy *inputs* (diskrétní vstupy), *coils* (cívky), *inputRegs* (vstupní registry) a *holdingRegs* (vnitřní registry). Počty objektů v zónách jsou dány vstupy *inputsCnt, coilsCnt, inputRegCnt* a *holdingRegCnt.*

Je důležité, aby vstupy *inputs* a *coils* odkazovali na proměnné typu *BOOL* nebo *ARRAY OF BOOL.* Vstupy *inputRegs* a *holdingRegs* mohou odkazovat na jakýkoli typ kromě *BOOL*.

Počty objektů v zónách musí být rovny nebo menší než je velikost proměnných, na které odkazují.

Pokud tyto podmínky nejsou dodrženy může dojít k zápisu do jiné části paměti.

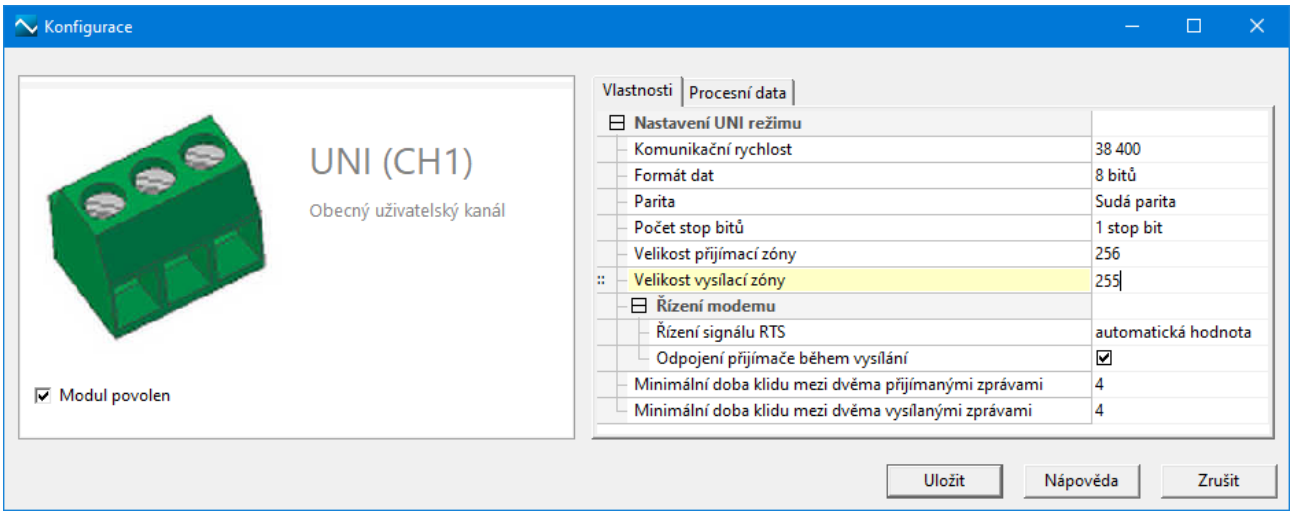
Zóny se mohou vzájemně překrývat. Překrytí zón umožňuje ke stejné paměti přistupovat jak po 16 bitových slovech tak i po bitech.

Popis proměnných :

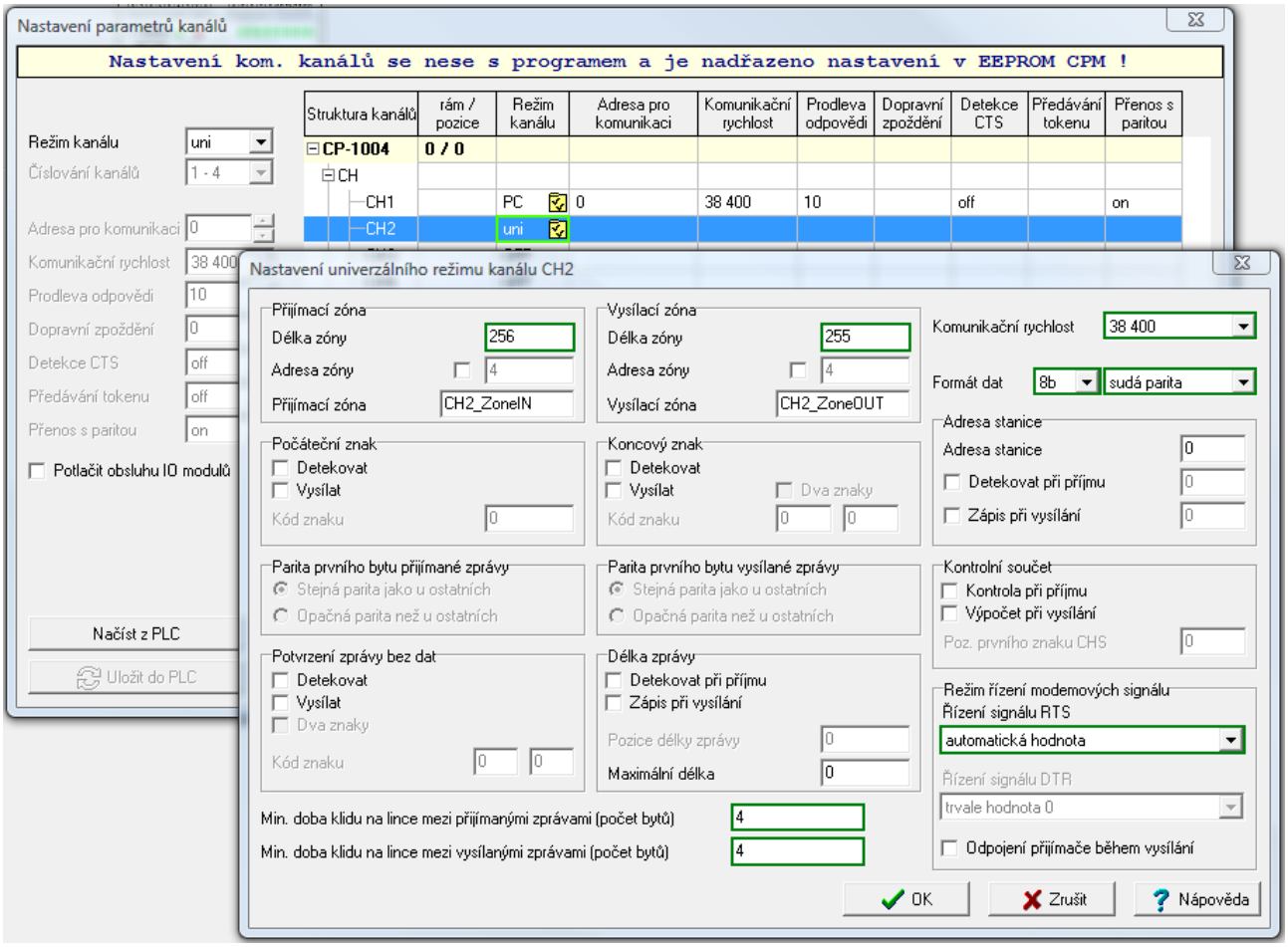
	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	sAdr	USINT	Adresa 1-247
	chanCode	UINT	kód kanálu (CH1_uni, ..., CH10_uni)
	inputsCnt	UINT	Počet diskrétních vstupů (počet BOOLů)
	coilsCnt	UINT	Počet cívek/diskrétních výstupů (počet BOOLů)
	inputRegCnt	UINT	Počet vstupních registrů (počet WORDů)
	holdingRegCnt	UINT	Počet vnitřních registrů (počet WORDů)
VAR_OUTPUT			
	read	BOOL	Byla čtena data
	write	BOOL	Byla zapsána data
	brdcst	BOOL	Byl přijat broadcast
	lastCmd	USINT	Poslední přijatý příkaz
	lastAddr	UINT	Poslední čtená/zapisovaná adresa
	lastQuantity	UINT	Poslední počet čtených/zapisovaných objektů
	err	BOOL	Chyba
	errCode	USINT	Chybový kód
	msgCnt	UDINT	Počet zpracovaných zpráv
	errCnt	UDINT	Počet chyb (crc, parita,..)
	excCnt	UDINT	Počet výjimek
VAR_IN_OUT			
	inputs	BOOL	První diskrétní vstup v poli (musí být BOOL)
	coils	BOOL	První cívka/diskrétní výstup v poli (musí být BOOL)
	inputRegs	UINT	První vstupní registr v poli
	holdingRegs	UINT	První vnitřní registr v poli

Zvolený komunikační kanál je nutné nastavit do režimu UNI. Komunikační rychlost, formát dat a parita musí být nastavena podle Modbus mastera, minimální doba klidu na lince mezi přijímanými a vysílanými zprávami by měla být 4 byty Minimální délka přijímací zóny je **256** bytů. Minimální délka vysílací zóny je **255** bytů.

Pracuje-li podřízené zařízení „bez parity" musí se nastavit v dialogu „parita trvale 1", aby byla splněna podmínka dvou stop bitů. Pokud není použit kanál s pevných rozhraním, musí být osazen submodulem MR-01xx podle varianty linky RS-232, RS-485, RS-422.



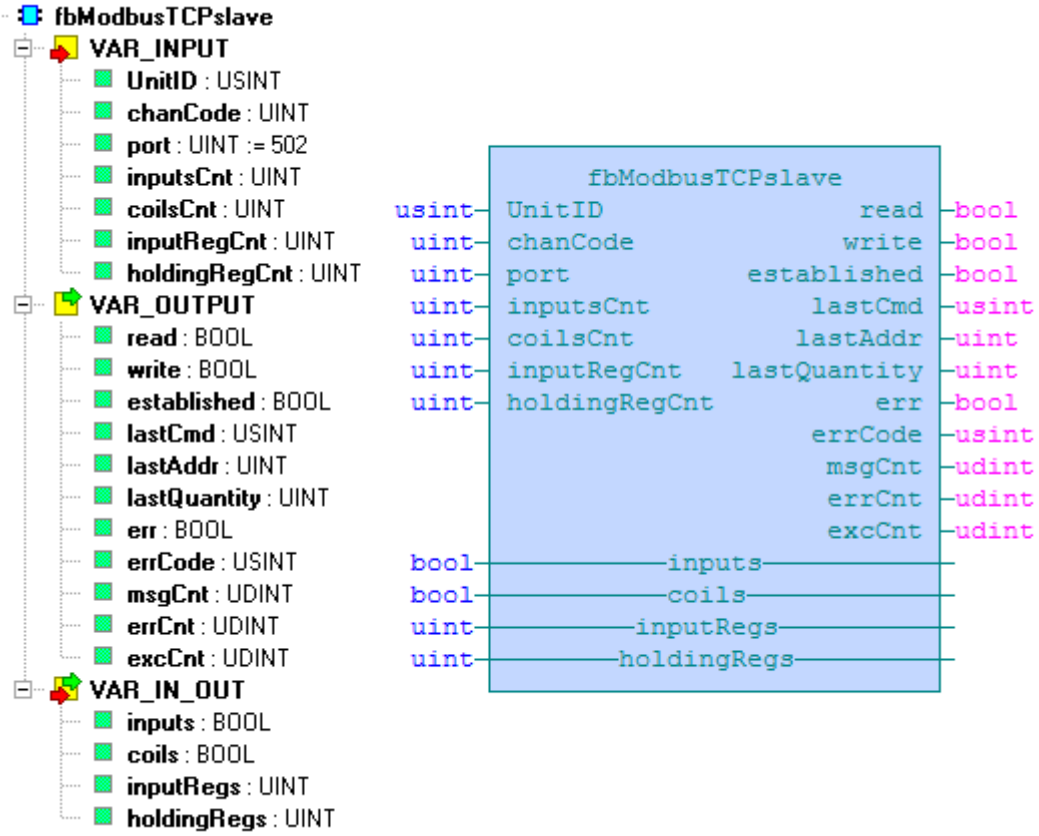
Příklad nastavení kanálu CH1 pro blok fbModbusRTUslave v nástroji I/O Configurátor



Příklad nastavení kanálu CH2 pro blok fbModbusRTUslave

3.2 Funkční blok fbModbusTCPslave

Knihovna: *ModbusRTUlib*



Funkční blok *fbModbusTCPslave* interpretuje příkazy ModbusTCP na TCP spojení specifikovaných vstupem *chanCode* s identifikátorem zařízení daným vstupem *UnitID*. Pokud je UnitID nastaveno na 0, tak jsou (od verze knihovny 3.9) přijímány všechny zprávy bez ohledu na identifikátor. Podporované příkazy viz. Kap. 1.3 Tab. 1.

Datové bloky jsou definovány vstupy *inputs* (diskrétní vstupy), *coils* (cívky), *inputRegs* (vstupní registry) a *holdingRegs* (vnitřní registry). Počty objektů v zónách jsou dány vstupy *inputsCnt, coilsCnt, inputRegCnt* a *holdingRegCnt.*

Je důležité, aby vstupy *inputs* a *coils* odkazovali na proměnné typu *BOOL* nebo *ARRAY OF BOOL.* Vstupy *inputRegs* a *holdingRegs* mohou odkazovat na jakýkoli typ kromě *BOOL*.

Počty objektů v zónách musí být rovny nebo menší než je velikost proměnných, na které odkazují.

Pokud tyto podmínky nejsou dodrženy může dojít k zápisu do jiné části paměti.

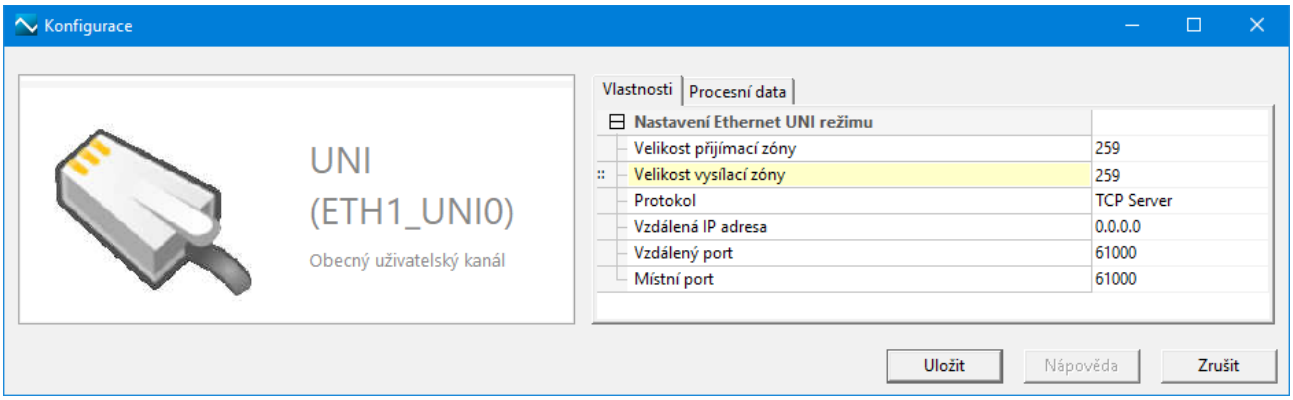
Zóny se mohou vzájemně překrývat. Překrytí zón umožňuje ke stejné paměti přistupovat jak po 16 bitových slovech tak i po bitech.

Standardně má PLC Tecomat na ethernetovém rozhraní aktivní dvě spojení TCP a UDP v režimu MDB, které zpracovávají Modbus příkazy. Voláním tohoto bloku se TCP spojení deaktivují, aby nedocházelo ke kolizi výchozího ovladače a funkčního bloku. Pro úplnou deaktivaci režimu MDB (TCP i UDP) lze použít funkci *fcModbusTcpUdpOff*.

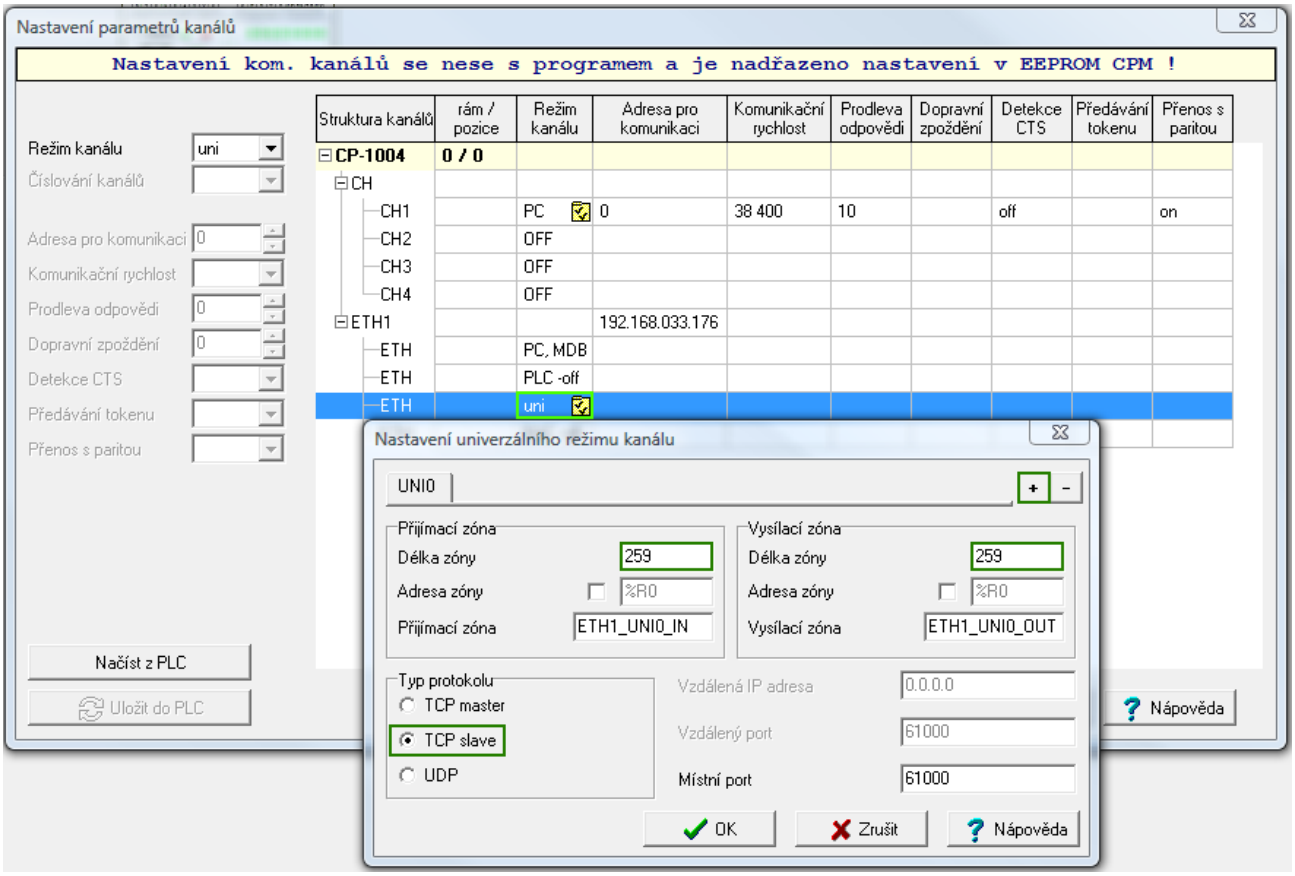
Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	UnitID	USINT	Identifikátor slave zařízení (pokud je nula přijímá všechny dotazy bez ohledu na identifikátor)
	chanCode	UINT	kód kanálu (CH1_uni, ..., CH10_uni)
	port	UINT	Číslo místního portu (502)
	inputsCnt	UINT	Počet diskrétních vstupů (počet BOOLů)
	coilsCnt	UINT	Počet diskrétních výstupů (počet BOOLů)
	inputRegCnt	UINT	Počet vstupních registrů (počet WORDů)
	holdingRegCnt	UINT	Počet registrů (počet WORDů)
VAR_OUTPUT			
	read	BOOL	Byla čtena data
	write	BOOL	Byla zapsána data
	established	BOOL	TCP spojení navázáno
	lastCmd	USINT	Poslední přijatý příkaz
	lastAddr	UINT	Poslední čtená/zapisovaná adresa
	lastQuantity	UINT	Poslední počet čtených/zapisovaných objektů
	err	BOOL	Chyba
	errCode	USINT	Chybový kód
	msgCnt	UDINT	Počet zpracovaných zpráv
	errCnt	UDINT	Počet chyb
	excCnt	UDINT	Počet výjimek
VAR_IN_OUT			
	inputs	BOOL	První diskrétní vstup v poli (musí být BOOL)
	coils	BOOL	První diskrétní výstup v poli (musí být BOOL)
	inputRegs	UINT	První vstupní registr v poli
	holdingRegs	UINT	První registr v poli

Zvolené spojení v režimu UNI musí být nastaveno na typ protokolu TCP slave. Minimální délka přijímací a vysílací zóny je **259** bytů. Místní port může být nastaven na libovolnou hodnotu, protože bude doplněny dle vstupu *port* funkčního bloku.



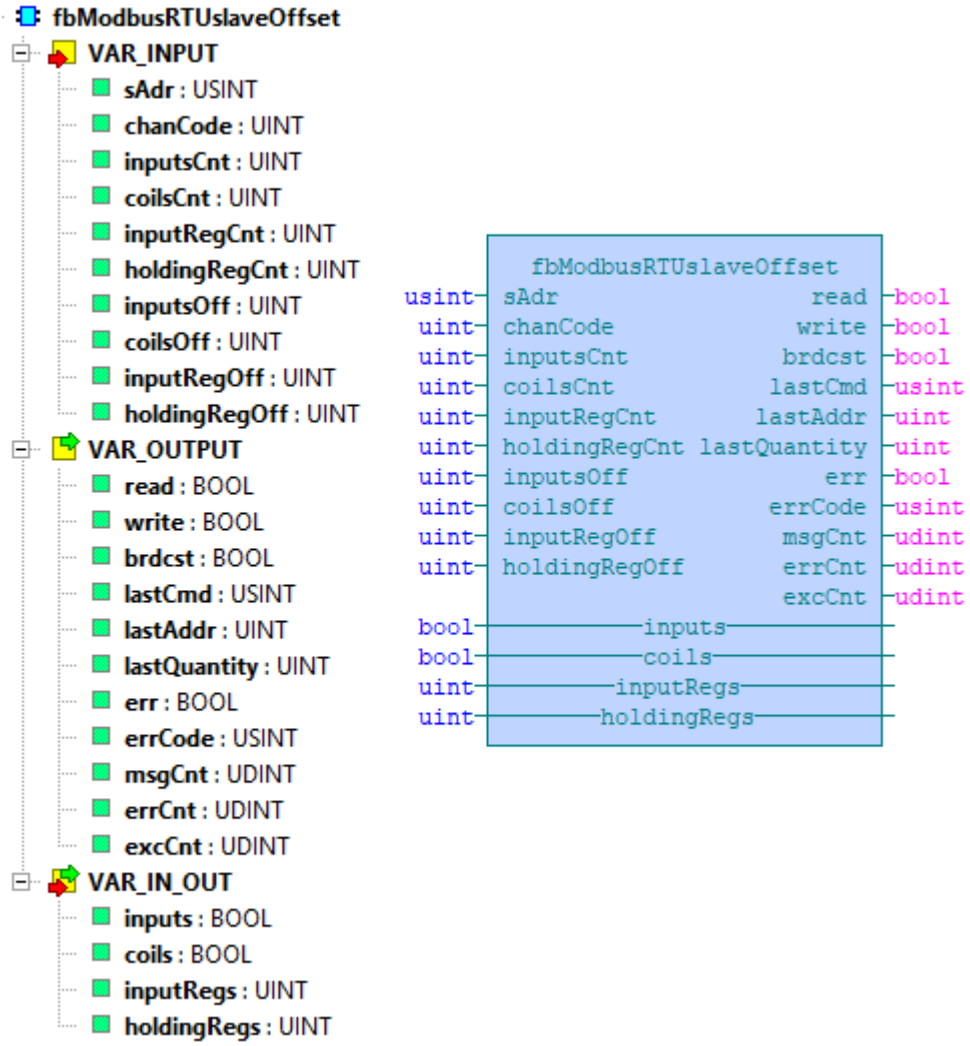
Příklad nastavení spojení UNI0 pro blok ModbusTCPslave v nástroji I/O Configurátor



Příklad nastavení spojení UNI0 pro blok ModbusTCPslave
(další spojení mohou být přidána tlačítkem +)

3.3 *Funkční blok fbModbusRTUslaveOffset*

Knihovna: *ModbusRTUlib*



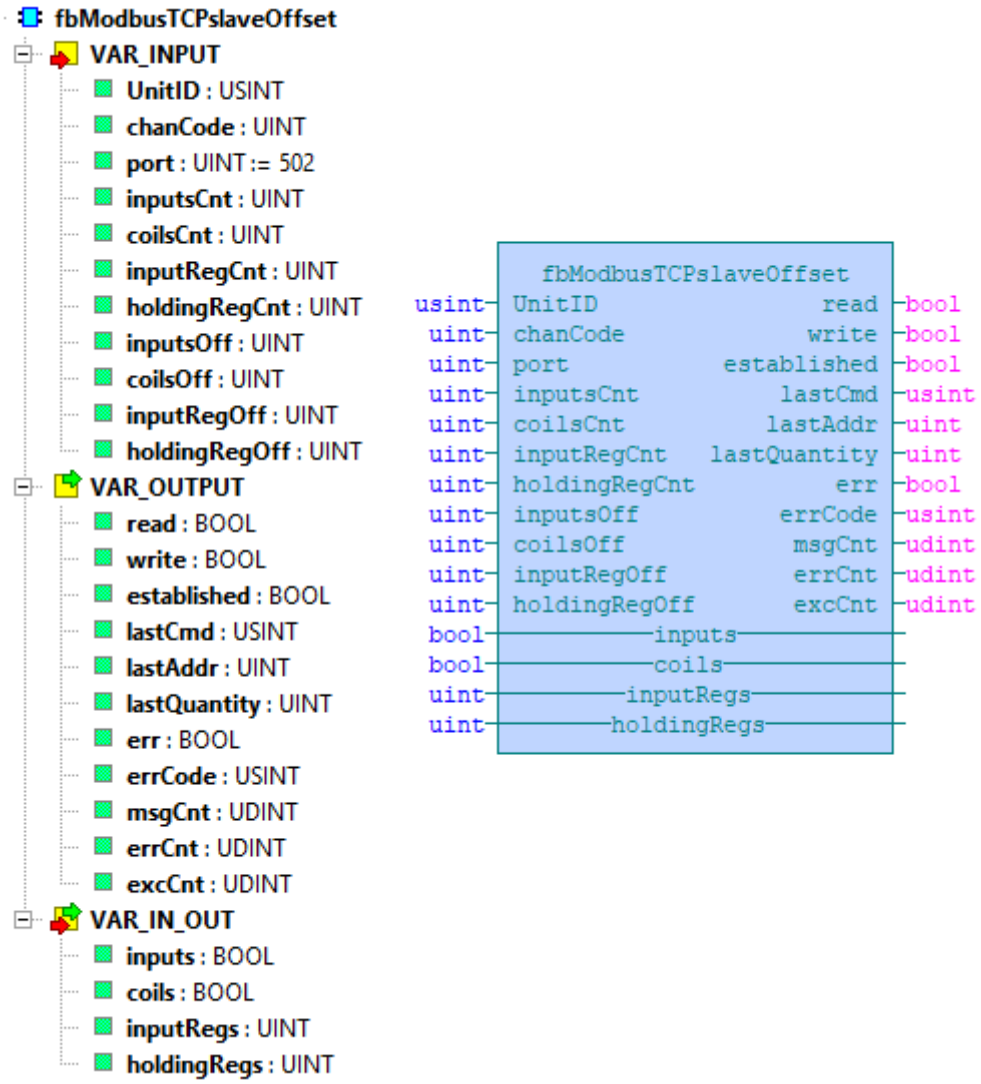
Funkční blok *fbModbusRTUslaveOffset* je variantou bloku *fbModbusRTUslave*, která umožňuje nastavit počáteční adresu vstupů, vystupů a registrů na vstupech *inputsOff*, *coilsOff*, *inputRegOff* a *holdingRegOff*.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	sAdr	USINT	Adresa 1-247
	chanCode	UINT	kód kanálu (CH1_uni, ..., CH10_uni)
	inputsCnt	UINT	Počet diskrétních vstupů (počet BOOLů)
	coilsCnt	UINT	Počet cívek/diskrétních výstupů (počet BOOLů)
	inputRegCnt	UINT	Počet vstupních registrů (počet WORDů)
	holdingRegCnt	UINT	Počet vnitřních registrů (počet WORDů)
	inputsOff	UINT	Adresa prvního diskrétního vstupu
	coilsOff	UINT	Adresa první cívky/diskrétního výstupu
	inputRegOff	UINT	Adresa prvního vstupního registru
	holdingRegOff	UINT	Adresa prvního vnitřních registru
VAR_OUTPUT			
	read	BOOL	Byla čtena data
	write	BOOL	Byla zapsána data
	brdcst	BOOL	Byl přijat broadcast
	lastCmd	USINT	Poslední přijatý příkaz
	lastAddr	UINT	Poslední čtená/zapisovaná adresa
	lastQuantity	UINT	Poslední počet čtených/zapisovaných objektů
	err	BOOL	Chyba
	errCode	USINT	Chybový kód
	msgCnt	UDINT	Počet zpracovaných zpráv
	errCnt	UDINT	Počet chyb (crc, parita,..)
	excCnt	UDINT	Počet výjimek
VAR_IN_OUT			
	inputs	BOOL	První diskrétní vstup v poli (musí být BOOL)
	coils	BOOL	První cívka/diskrétní výstup v poli (musí být BOOL)
	inputRegs	UINT	První vstupní registr v poli
	holdingRegs	UINT	První vnitřní registr v poli

3.4 Funkční blok fbModbusTCPslaveOffset

Knihovna: *ModbusRTUlib*



Funkční blok *fbModbusTCPslaveOffset* je variantou bloku *fbModbusTCPslave*, která umožňuje nastavit počáteční adresu vstupů, vystupů a registrů na vstupech *inputsOff*, *coilsOff*, *inputRegOff* a *holdingRegOff*.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	UnitID	USINT	Identifikátor slave zařízení
	chanCode	UINT	kód kanálu (CH1_uni, ..., CH10_uni)
	port	UINT	Číslo místního portu (502)
	inputsCnt	UINT	Počet diskrétních vstupů (počet BOOLů)
	coilsCnt	UINT	Počet diskrétních výstupů (počet BOOLů)
	inputRegCnt	UINT	Počet vstupních registrů (počet WORDů)
	holdingRegCnt	UINT	Počet registrů (počet WORDů)
	inputsOff	UINT	Adresa prvního diskrétního vstupu
	coilsOff	UINT	Adresa první cívky/diskrétního výstupu
	inputRegOff	UINT	Adresa prvního vstupního registru
	holdingRegOff	UINT	Adresa prvního vnitřních registru
VAR_OUTPUT			
	read	BOOL	Byla čtena data
	write	BOOL	Byla zapsána data
	established	BOOL	TCP spojení navázáno
	lastCmd	USINT	Poslední přijatý příkaz
	lastAddr	UINT	Poslední čtená/zapisovaná adresa
	lastQuantity	UINT	Poslední počet čtených/zapisovaných objektů
	err	BOOL	Chyba
	errCode	USINT	Chybový kód
	msgCnt	UDINT	Počet zpracovaných zpráv
	errCnt	UDINT	Počet chyb
	excCnt	UDINT	Počet výjimek
VAR_IN_OUT			
	inputs	BOOL	První diskrétní vstup v poli (musí být BOOL)
	coils	BOOL	První diskrétní výstup v poli (musí být BOOL)
	inputRegs	UINT	První vstupní registr v poli
	holdingRegs	UINT	První registr v poli

3.5 Funkce fcModbusUdpOff

Knihovna: ModbusRTUlib

Přítomnost volání funkce v kódu uživatelském programu vypíná Modbus UDP ovladač. Pro znovu zapnutí ovladače je nutné nahrát kód bez volání funkce a restartovat centrálu.

Popis proměnných :

	Proměnná	Typ	Význam
fcModbusUdpOff			
	Návratová hodnota	BOOL	*False*

3.6 Funkce fcModbusTcpOff

Knihovna: ModbusRTUlib

Přítomnost volání funkce v kódu uživatelském programu vypíná Modbus TCP ovladač. Pro znovu zapnutí ovladače je nutné nahrát kód bez volání funkce a restartovat centrálu. Tato funkce ruší nastavení provedené pomocí *fcModbusUdpOff*. Pro vypnutí obou ovladačů použijte *fcModbusTcpUdpOff*.

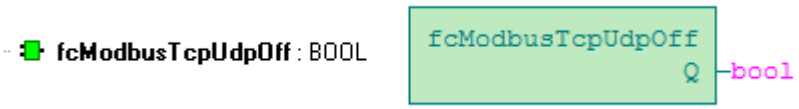
Tato funkce je automaticky volána blokem *fbModbusTCPslave*.

Popis proměnných :

	Proměnná	Typ	Význam
fcModbusTcpOff			
	Návratová hodnota	BOOL	*False*

3.7 Funkce fcModbusTcpUdpOff

Knihovna: ModbusRTUlib



Přítomnost volání funkce v kódu uživatelském programu vypíná Modbus TCP a UDP ovladač. Pro znovu zapnutí ovladače je nutné nahrát kód bez volání funkce a restartovat centrálu.

Popis proměnných :

	Proměnná	***Typ***	***Význam***
fcModbusTcpUdpOff			
	Návratová hodnota	BOOL	*False*

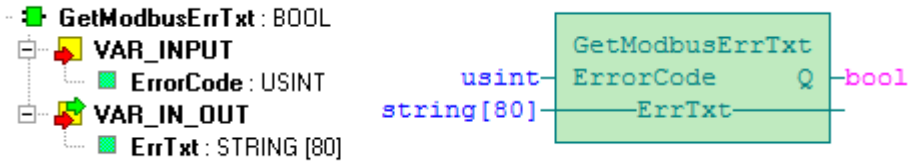
4 CHYBOVÁ HLÁŠENÍ

Funkční bloky při neúspěšné komunikaci vydávají chybové kódy umožňující blíže určit příčinu chyby. Knihovna obsahuje funkci *GetModbusErrTxt* pro převod chybových kódů na texty. Tato funkce je univerzální pro Modbus slave i master.

- *GetModbusErrTxt* funkce pro převod chybového kódu na text

4.1 Funkce GetModbusErrTxt

Knihovna: ModbusRTUlib



Převádí kód chyby na textový popis v anglickém jazyce. Funkce vrací *true* je-li chybový *ErrorCode* nenulový.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	ErrorCode	USINT	Chybový kód
VAR_IN_OUT			
	ErrTxt	STRING[80]	Popis chyby
GetModbusErrTxt			
	Návratová hodnota	BOOL	*True* je-li *ErrorCode* nenulový

Příklad volání funkce GetModbusErrTxt

```
PROGRAM prgMain
  VAR
    LastError  : STRING;
    MdbError   : STRING;
    MdbData    : ARRAY [0..1] OF UINT;
    Command    : TCmdStruct;
    ModbusMas1 : ModbusRTUmas;
  END_VAR

  ModbusCmd(Gr := 1, SNo := 1, FNC := 03, StAdr := 0, NoPoint := 2,
            PtrData := ADR(MdbData), Cmd := Command);

  ModbusMas1(EN := 1, GrSel := 1, MaxCmd := 1, chanCode := CH1_uni,
             Commands := Command);

  IF GetModbusErrTxt(ErrorCode := ModbusMas1.ErrCode, ErrTxt := MdbError) THEN
    LastError := MdbError;
  END_IF;

END_PROGRAM
```

4.2 Kódy chybových hlášení

0 ... No error	Bez chyby
1 ... Channel is not in uni mode	Kanál není v uni módu
2 ... Sending data are too long	Posílaná data jsou příliž dlouhá
3 ... Received data are too long	Přijímaná data jsou příliž dlouhá
4 ... Wrong channel code	Chybný kód kanálu
5 ... Previous message is not sent yet	Předchozí zpráva není ještě odeslána
6 … Length of data to sent is zero	Nulová délka vysílaných dat
16 ... Invalid start delimiter	Neplatný startovací znak
17 ... Parity error	Chyba parity
18 ... Maximum message length exceeded	Překročena max. délka zprávy
19 ... Invalid second byte of acknowledgment	Neplatný druhý byte potvrzení
20 ... Invalid second byte of end delimiter	Neplatný druhý byte koncového znaku
24 ... Check sum error	Chyba kontrolního součtu
25 ... Invalid end delimiter	Neplatný koncový znak
49 ... Invalid length of sent data	Neplatná délka posílané zprávy
50 ... Length of data to sent is zero	Nulová délka vysílaných dat
64 ... Timeout not held	Nedodržena přestávka
129 ... Response with other slave address	Odpověď s jinou podřízenou adresou
130 ... Response with other FNC	Odpověď s jinou FNC
131 ... Checksum error in reception	Chyba kontrolního součtu během příjmu
132 ... Unknown FNC in transmit command	Neznámý FNC ve vysílaném příkazu
133 ... Response with Unknown FNC	Odpověď s neznámým FNC
135 ... Response Timeout Error	Uplynul čas čekání na odpověď
137 ... Exception: ILLEGAL FUNCTION	Výjimka: Nepovolená funkce
138 ... Exception: ILLEGAL DATA ADDRESS	Výjimka: Nepovolená adresa dat
139 ... Exception: ILLEGAL DATA VALUE	Výjimka: Nepovolená hodnota dat
140 ... Exception: SLAVE DEVICE FAILURE	Výjimka: Neschopné podřízené zařízení
141 ... Exception: ACKNOWLEDGE	Výjimka: Potvrzení
142 ... Exception: SLAVE DEVICE BUSY	Výjimka: Podřízené zařízení zaměstnáno
144 ... Exception: MEMORY PARITY ERROR	Výjimka: Chyba parity paměti
146 ... Exception: GATEWAY PATH UNAVAILABLE	Výjimka: Nepoužitelná cesta bránou
147 ... GATEWAY TARGET DEVICE FAILED TO RESPOND	Výjimka: Cílové zařízení nepřístupné touto bránou
148 ... Ivalid parameter chanCode	Neplatný parametr chanCode
149 … Can not establish a TCP connection	Nelze navázat TCP spojení
150 … Invalid IP address of slave device	Neplatná IP adresa slave zařízení

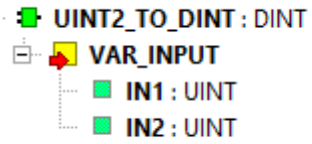
Chyby 1..64 jsou obecné chyby komunikace přebírané od funkcí z knihovny ComLib. Chyby 137..147 jsou hlášeny v odpovědi připojeného zařízení slave. Jejich podrobný popis je uveden v manuálu PI_MBUS_300.pdf

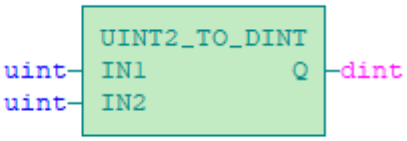
5 POMOCNÉ FUNKCE

Následující funkce usnadňují převod datových typů, které se přenáší ve více Modbus registrech. Zatímco u proměnných, které se přenášejí v rámci jednoho Modbus registru, je pořadí řešeno přímo komunikačním blokem, u větších datových typů je nutné pořadí vyřešit až v uživatelském programu.

5.1 Funkce UINT2_TO_DINT

Knihovna: ModbusRTUlib





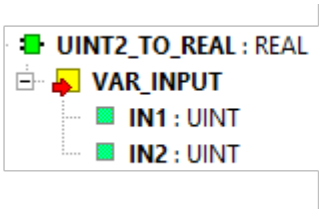
Převede 2 modbusové registry obsahující 32 bitovou celočíselnou hodnotu na DINT.

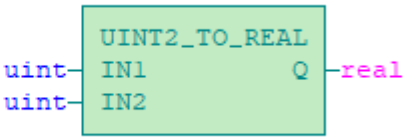
Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	IN1	UINT	více významný registr
	IN2	UINT	méně významný registr
UINT2_TO_DINT			
	Návratová hodnota	DINT	32 bitové číslo se znaménkem

5.2 Funkce UINT2_TO_REAL

Knihovna: ModbusRTUlib





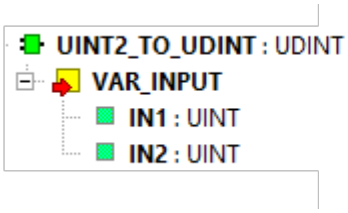
Převede 2 modbusové registry obsahující 32 bitové číslo s plovoucí čárkou dle IEEE 754 na REAL.

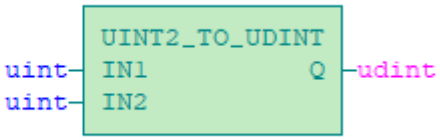
Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	IN1	UINT	více významný registr
	IN2	UINT	méně významný registr
UINT2_TO_REAL			
	Návratová hodnota	REAL	32 bitové číslo s plovoucí čárkou

5.3 Funkce UINT2_TO_UDINT

Knihovna: ModbusRTUlib





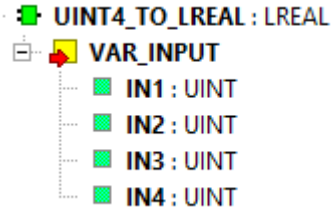
Převede 2 modbusové registry obsahující 32 bitovou celočíselnou hodnotu na UDINT.

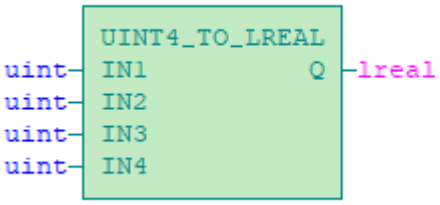
Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	IN1	UINT	více významný registr
	IN2	UINT	méně významný registr
UINT2_TO_UDINT			
	Návratová hodnota	UDINT	32 bitové číslo bez znaménka

5.4 Funkce UINT4_TO_LREAL

Knihovna: ModbusRTUlib





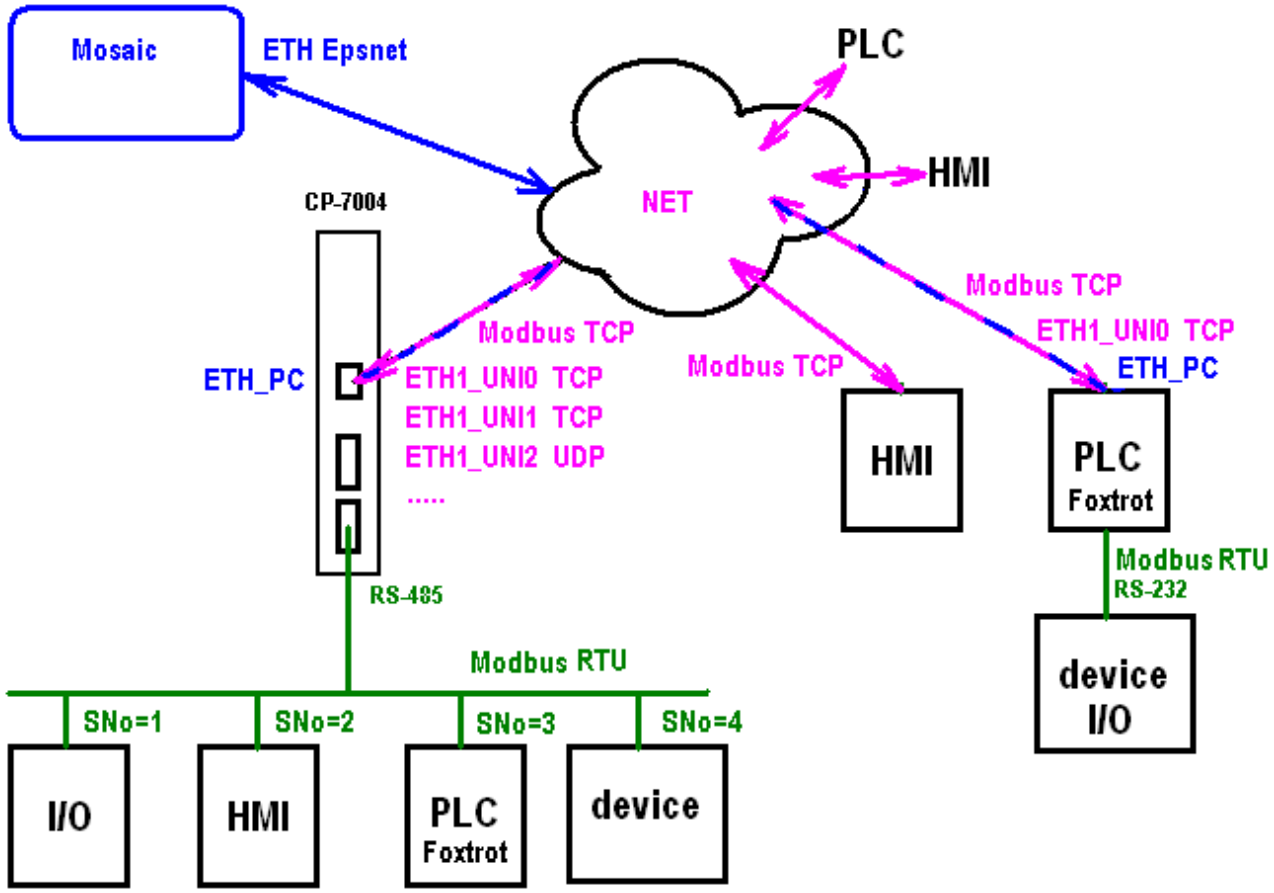
Převede 4 modbusové registry obsahující 64 bitové číslo s plovoucí čárkou dle IEEE 754 na LREAL

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	IN1	UINT	nejvíce významný registr
	IN2	UINT	více významný registr
	IN3	UINT	méně významný registr
	IN4	UINT	nejméně významný registr
UINT4_TO_LREAL			
	Návratová hodnota	LREAL	64 bitové číslo s plovoucí čárkou

6 PŘÍKLADY PROGRAMU KOMUNIKACE MODBUS MASTER

6.1 Topologie sítě zařízení Modbus



Komunikace po sériových linkách může probíhat jako Master-Slave současně na více kanálech. V síti Ethernet je možné zařízení libovolně kombinovat. Master Modbus může směřovat až do více zařízení s různými IP adresami najednou. Přitom může probíhat zároveň i jiná komunikace, například s vývojovým prostředím Mosaic, nebo s jiným PLC.

6.2 *Příklad 1 – komunikace sériovým kanálem*

Příklad komunikace sériovým kanálem na dvě stanice slave. Kanál CH2 musí být osazen submodulem MR-0114 s rozhraním RS-485. Volání funkcí *ModbusCmd* v jazyku ST pro inicializaci dvou příkazů pro komunikační kanál CH2 nastavený v režimu UNI.

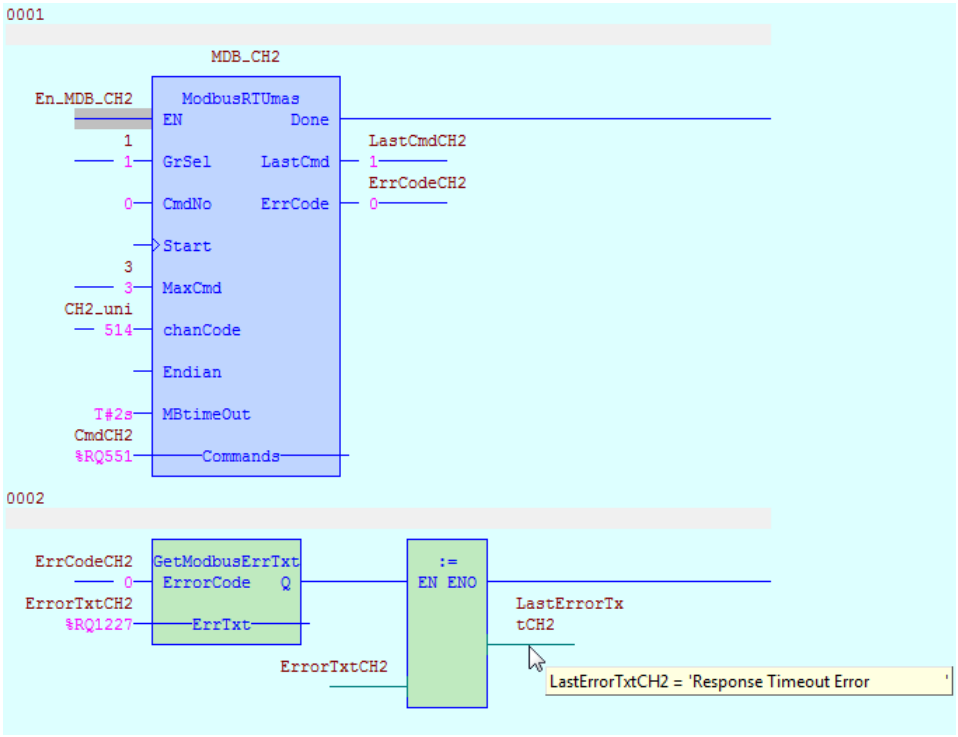
```
TYPE
  TDataBlock1 : STRUCT //teploty
    Temperature1 : REAL;
    Temperature2 : REAL;
  END_STRUCT;

  TDataBlock2 : STRUCT //tlaky
    Pressure1 : REAL;
    Pressure2 : REAL;
    Pressure3 : REAL;
  END_STRUCT;
END_TYPE

VAR_GLOBAL
  Temps   : TDataBlock1;
  Press   : TDataBlock2;
  Flags   : ARRAY [1..5] OF BOOL;        // pole bitových příznaků
  CmdCH2  : ARRAY [1..3] OF TCmdStruct; (* pole příkazů Modbus
                                           pro řízení kanálu CH2 *)
END_VAR

PROGRAM prgMain
  ModbusCmd(Gr:=1, FNC:=03, SNo:=1,
            StAdr:=0, NoPoint:=SIZEOF(Temps)/2,
            PtrData:=adr(Temps), Cmd:=CmdCH2[1]);
  ModbusCmd(Gr:=1, FNC:=03, SNo:=2,
            StAdr:=4, NoPoint:=SIZEOF(Press)/2,
            PtrData:=adr(Press), Cmd:=CmdCH2[2]);
  ModbusCmd(Gr:=1, FNC:=01, SNo:=1,
            StAdr:=9, NoPoint:=5,
            PtrData:=adr(Flags), Cmd:=CmdCH2[3]);
END_PROGRAM
```

Volání FB *ModbusRTUmas* je napsáno v grafickém jazyku FBD. Používá komunikační kanál CH2. Zobrazení je přepnuto v ladícím režimu. FB právě vykonal příkazu číslo 1. V obvodu 0002 je funkce *GetModbusErrTxt*, která převádí chybový kód z proměnné *ErrCodeCH2* do textového řetězce a ukládá jej do proměnné *ErrorTxtCH2*. Pokud nastane chyba je tato překopírována do proměnné LastErrorTxtCH2. V příkladu je vidět, že poslední chyba byla vypršení času při čekání na odpověď podřízené stanice.



6.3 Příklad 2 – jednoduchá komunikace Modbus TCP

Příklad jednoduché komunikace Modbus TCP zapsaný v jazyku ST.

Program periodicky čte 6 wordů z koncového zařízení připojeného přes Ethernet. Nepoužité vstupní a výstupní parametry volaného FB nejsou uvedeny, takže zůstávají v implicitních hodnotách. Řídící pole příkazů má jenom jeden příkaz. Je-li třeba komunikaci naředit v čase, můžeme proměnnou *En* ovládat vhodným periodickým signálem nebo použít manuální režim.

```
PROGRAM prgMain
  VAR
    LastError   : STRING;
    MdbError    : STRING;
    MdbData     : ARRAY [0..5] OF UINT;
    Command     : ARRAY [1..1] OF TCmdStructTCP;
    ModbusMas : ModbusTCPmas;
  END_VAR

  ModbusCmdTCP(Gr := 1, IP := '192.168.134.41', UnitID := 0, FNC := 03,
               StAdr := 0, NoPoint := 6, PtrData := ADR(MdbData),
               Cmd := Command[1]);

  ModbusMas(EN := 1, GrSel := 1,
            MaxCmd := sizeof(Command)/sizeof(TCmdStructTCP),
            chanCode := ETH1_uni0, Commands := Command[1]);

  IF GetModbusErrTxt(ErrorCode := ModbusMas.ErrCode, ErrTxt := MdbError) THEN
    LastError := MdbError;
  END_IF;

END_PROGRAM
```

6.4 Příklad 3 – komplexní komunikace Modbus TCP v jazyku ST

Příklad komplexní komunikace Modbus TCP na 80 podřízených zařízení zapsaný v jazyku ST. V příkladu se cyklickou čtou informace s 80 zařízení s IP adresami 192.168.2.101 až 192.168.2.181. V případě, že je nastavena proměnná *WriteControl* jsou do zařízení zapsány bity obsažené v poli *Control*.

Nastavení příkazů je podmíněné funkcí *ProgramIsChanged* z knihovny SysLib verze 3.3. Tím je zajištěno, že se inicializace provede po každé změně programu a při všech typech restartů.

Zápis je realizován přepnutím *GrSel* na 2 a vyčkáním, až bude zápis proveden bez chyb do všech zařízení. Pokud se zápis podaří (poslední vykonaný příkaz je roven zapamatovanému číslu při startu zápisu *StartCommand* a počet chyb je roven zapamatované hodnotě *StartErrCnt*) je vynulována proměnná WriteControl a GrSel nastaven zpět na 1.

```
TYPE
  TDataBlock1 : STRUCT
    Temperature1 : REAL;
    Temperature2 : REAL;
  END_STRUCT;
END_TYPE

VAR_GLOBAL
  Temps    : ARRAY [1..80] OF TDataBlock1;
  Control  : ARRAY [1..80] OF BOOL;
  CmdTCP1  : ARRAY [1..160] OF TCmdStructTCP;
END_VAR

PROGRAM prgMain
  VAR
    i              : UINT;
    MDB_TCP        : ModbusTCPmas;
    GrSel          : USINT := 1;
    WriteControl   : BOOL;
    WriteControlR  : R_TRIG;
    DoneF          : F_TRIG;
    StartCommand   : USINT;
    StartErrCnt    : UDINT;
    ErrCnt         : UDINT;
    ErrorTxt       : STRING;
    LastErrorTxt   : STRING;
  END_VAR

  IF ProgramIsChanged() THEN
    FOR i := 1 TO 80 DO
      ModbusCmdTCP(Gr:=1, FNC:=03, UnitID:=0,
        IP:='192.168.2.'+UINT_TO_STRING(i+100),
        StAdr:=0, NoPoint:=SIZEOF(TDataBlock1)/2,
        PtrData:=adr(Temps[i]), Cmd:=CmdTCP1[i*2-1]);
    END_FOR;
    FOR i := 1 TO 80 DO
      ModbusCmdTCP(Gr:=2, FNC:=05, UnitID:=0,
        IP:='192.168.2.'+UINT_TO_STRING(i+100),
        StAdr:=0, NoPoint:=1,
        PtrData:=adr(Control[i]), Cmd:=CmdTCP1[i*2]);
    END_FOR;
  END_IF;
```

```
  WriteControlR(CLK := WriteControl);
  IF WriteControlR.Q THEN
    GrSel := 2;
    StartCommand := MDB_TCP.LastCmd;
    StartErrCnt  := ErrCnt;
  END_IF;

  MDB_TCP(EN := true, GrSel := GrSel,
          MaxCmd := sizeof(CmdTCP1)/sizeof(TCmdStructTCP),
          chanCode := ETH1_uni0, Commands := CmdTCP1[1]);

  IF GetModbusErrTxt(ErrorCode := MDB_TCP.ErrCode, ErrTxt := ErrorTxt) THEN
    LastErrorTxt := ErrorTxt;
    ErrCnt := ErrCnt + 1;
  END_IF;

  DoneF(CLK := MDB_TCP.Done);

  IF DoneF.Q AND StartCommand = MDB_TCP.LastCmd THEN
    IF StartErrCnt = ErrCnt THEN
      GrSel := 1;
      WriteControl := false;
    ELSE
      StartErrCnt := ErrCnt;
    END_IF;
  END_IF;

END_PROGRAM
```

6.5 *Příklad 4 – PLC jako podřízená stanice na sériové lince*

Příklad realizace sériové modbusové komunikace na kanálu CH1 pomocí funkčního bloku *fbModbusRTUslave*, kdy se PLC chová jako podřízená stanice s adresou 1. Komunikační zóny jsou definovány tak, že 64 diskrétních vstupů míří do stejné oblasti jako cívky a 128 vnitřních registrů je definováno přes stejný počet registrů vstupních.

```
VAR_GLOBAL CONSTANT
  MODBUS_BITS       : UINT := 64;
  MODBUS_WORDS      : UINT := 128;
END_VAR

VAR_GLOBAL
  ModbusInputs     : ARRAY [1..MODBUS_BITS]  OF BOOL;
  ModbusCoils      AT ModbusInputs;
  ModbusInputReg   : ARRAY [1..MODBUS_WORDS] OF UINT;
  ModbusHoldingReg AT ModbusInputReg;
END_VAR

PROGRAM prgMain
  VAR
    ModbusSlave1 : fbModbusRTUslave;
  END_VAR

  ModbusSlave1( sAdr := 1,
                chanCode := CH1_uni,
                inputsCnt := MODBUS_BITS,
                coilsCnt := MODBUS_BITS,
                inputRegCnt := MODBUS_WORDS,
                holdingRegCnt := MODBUS_WORDS,
                inputs := ModbusInputs[1],
                coils := ModbusCoils[1],
                inputRegs := ModbusInputReg[1],
                holdingRegs := ModbusHoldingReg[1]);


END_PROGRAM
```

6.6 *Příklad 5 – PLC jako podřízená stanice na Ethernetu*

Příklad použití dvou instancí bloku *fbModbusTCPslave*, umožňující paralelní připojení dvou Modbus klientů. Příklad je možné rozšířit o další spojení, při dodržení dvou základních pravidel. Každá instance musí používat vlastní univerzální spojení a všechny instance musí odkazovat na stejné zóny.

```
VAR_GLOBAL CONSTANT
  MODBUS_INPUTS_COUNT       : UINT := 64;
  MODBUS_COILS_COUNT        : UINT := 64;
  MODBUS_INPUT_REG_COUNT    : UINT := 32;
  MODBUS_HOLDING_REG_COUNT  : UINT := 128;
END_VAR

VAR_GLOBAL
  ModbusInputs     : ARRAY [1..MODBUS_INPUTS_COUNT]      OF BOOL;
  ModbusCoils      : ARRAY [1..MODBUS_COILS_COUNT]       OF BOOL;
  ModbusInputReg   : ARRAY [1..MODBUS_INPUT_REG_COUNT]   OF UINT;
  ModbusHoldingReg : ARRAY [1..MODBUS_HOLDING_REG_COUNT] OF UINT;
END_VAR

PROGRAM prgMain
  VAR
    ModbusSlave1 : fbModbusTCPslave;
    ModbusSlave2 : fbModbusTCPslave;
  END_VAR

  ModbusSlave1( UnitID := 0,
                chanCode := ETH1_uni0,
                inputsCnt := MODBUS_INPUTS_COUNT,
                coilsCnt := MODBUS_COILS_COUNT,
                inputRegCnt := MODBUS_INPUT_REG_COUNT,
                holdingRegCnt := MODBUS_HOLDING_REG_COUNT,
                inputs := ModbusInputs[1],
                coils := ModbusCoils[1],
                inputRegs := ModbusInputReg[1],
                holdingRegs := ModbusHoldingReg[1]);

  ModbusSlave2( UnitID := 0,
                chanCode := ETH1_uni1,
                inputsCnt := MODBUS_INPUTS_COUNT,
                coilsCnt := MODBUS_COILS_COUNT,
                inputRegCnt := MODBUS_INPUT_REG_COUNT,
                holdingRegCnt := MODBUS_HOLDING_REG_COUNT,
                inputs := ModbusInputs[1],
                coils := ModbusCoils[1],
                inputRegs := ModbusInputReg[1],
                holdingRegs := ModbusHoldingReg[1]);

END_PROGRAM
```

6.7 Příklad 6 – PLC jako podřízená stanice na Ethernetu pro Foxtrot řady 2xxx

Příklad použití *fbModbusTCPslave* umožňující paralelní připojení více Modbus klientů. Počet klientů je dán konstantou *MODBUS_CONNECTION_COUNT*. Spojení se vytváří automaticky s využitím funkce *OpenUniSocket*.

```
VAR_GLOBAL CONSTANT
  MODBUS_INPUTS_COUNT      : UINT := 64;
  MODBUS_COILS_COUNT       : UINT := 64;
  MODBUS_INPUT_REG_COUNT   : UINT := 32;
  MODBUS_HOLDING_REG_COUNT : UINT := 128;
  MODBUS_CONNECTION_COUNT  : UINT := 3;
END_VAR

VAR_GLOBAL
  ModbusInputs     : ARRAY[1..MODBUS_INPUTS_COUNT] OF BOOL;
  ModbusCoils      : ARRAY[1..MODBUS_COILS_COUNT] OF BOOL;
  ModbusInputReg   : ARRAY[1..MODBUS_INPUT_REG_COUNT] OF UINT;
  ModbusHoldingReg : ARRAY[1..MODBUS_HOLDING_REG_COUNT] OF UINT;
END_VAR

PROGRAM prgExample6
  VAR
    ModbusSlave : ARRAY [1..MODBUS_CONNECTION_COUNT] OF fbModbusTCPslave;
  END_VAR
  VAR_TEMP
    idx : UINT;
  END_VAR

  FOR idx := 1 TO MODBUS_CONNECTION_COUNT DO
    IF ModbusSlave[idx].chanCode = 0 THEN
      ModbusSlave[idx].chanCode := OpenUniSocket(protocol := UNI_TCP_SERVER);
    END_IF;
    ModbusSlave[idx](
      UnitID := 0,
      inputsCnt := MODBUS_INPUTS_COUNT,
      coilsCnt := MODBUS_COILS_COUNT,
      inputRegCnt := MODBUS_INPUT_REG_COUNT,
      holdingRegCnt := MODBUS_HOLDING_REG_COUNT,
      inputs := ModbusInputs[1], coils := ModbusCoils[1],
      inputRegs := ModbusInputReg[1],
      holdingRegs := ModbusHoldingReg[1]);

  END_FOR;

END_PROGRAM
```

6.8 Příklad 7 – Jednoduchá komunikace Modbus TCP pro Foxtrot řady 2xxx

Příklad použití *fbModbusTCPmas2.* Spojení se vytváří automaticky s využitím funkce *OpenUniSocket*. Příklad také demonstruje použití převodních funkcí *UINT2_TO_REAL*, *UINT4_TO_LREAL*, *UINT2_TO_DINT* a *UINT2_TO_UDINT*.

```
PROGRAM prgModbusTest1
  VAR
    MDB_TCP      : fbModbusTCPmas2;
    ErrCnt       : UDINT;
    LastErrDT    : DT;
    LastErrorTxt : STRING;
    CmdTCP       : ARRAY[1..2] OF TCmdStructTCP;
    //modbus data
    Data         : ARRAY[1..10] OF UINT;
    Inputs       : ARRAY[1..8] OF BOOL;
    Temperature  : REAL;
    Energy       : LREAL;
    Counter1     : DINT;
    Counter2     : UDINT;
  END_VAR
  VAR_TEMP
    ErrorTxt     : STRING;
  END_VAR

  //inicializace prikazu
  IF ProgramIsChanged() THEN
    ModbusCmdTCP(Gr := 1,
      FNC := MDB_FNC_ReadHoldingRegisters,
      UnitID := 0, IP := '10.1.1.78',
      StAdr := 0,
      NoPoint := SIZEOF(Data) / 2,
      PtrData := adr(Data[1]),
      Cmd := CmdTCP[1]);
    ModbusCmdTCP(Gr := 1,
      FNC := MDB_FNC_ReadInputStatus,
      UnitID := 0, IP := '10.1.1.78',
      StAdr := 0, NoPoint := SIZEOF(Inputs) * 8,
      PtrData := adr(Inputs[1]),
      Cmd := CmdTCP[2]);
  END_IF;
  //zalozeni socketu (Foxtrot 2)
  IF MDB_TCP.chanCode = 0 THEN
    MDB_TCP.chanCode := OpenUniSocket(protocol := UNI_TCP_CLIENT);
  END_IF;
  //obsluha komunikace
  MDB_TCP(EN := true, GrSel := 1,
    MaxCmd := sizeof(CmdTCP) / sizeof(TCmdStructTCP),
    Commands := CmdTCP[1]);
  //konverze dat
  IF MDB_TCP.Done THEN
    IF MDB_TCP.LastCmd = 1 THEN
      Temperature := UINT2_TO_REAL(IN1 := Data[1], IN2 := Data[2]);
      Energy      := UINT4_TO_LREAL(IN1 := Data[3], IN2 := Data[4],
                                    IN3 := Data[5], IN4 := Data[6]);
      Counter1    := UINT2_TO_DINT(IN1 := Data[7], IN2 := Data[8]);
      Counter2    := UINT2_TO_UDINT(IN1 := Data[9], IN2 := Data[10]);
    END_IF;
  END_IF;
```

```
  //vyhodnoceni chyb
  IF GetModbusErrTxt(ErrorCode := MDB_TCP.ErrCode, ErrTxt := ErrorTxt) THEN
    LastErrorTxt := ErrorTxt;
    ErrCnt := ErrCnt + 1;
    LastErrDT := GetDateTime();
  END_IF;
END_PROGRAM
```

6.9 Příklad 8 – Náhrada režimu MDB pro Foxtrot řady 2xxx

Příklad použití *fbModbusRTUslave*, který představuje alternativu k režimu MDB. Na rozdíl od MDB režimu však omezuje čtení a zápis do uživatelsky definovaných zón místo, aby zpřístupňoval celé části zápisníku.

```
PROGRAM prgExample8
  VAR CONSTANT
    INPUTS_COUNT      : UINT := 64;
    COILS_COUNT       : UINT := 64;
    INPUT_REG_COUNT   : UINT := 32;
    HOLDING_REG_COUNT : UINT := 128;
  END_VAR
  VAR
    ModbusRTUslave : fbModbusRTUslave := (
      sAdr := 1,  chanCode := CH1_uni,
      inputsCnt   := INPUTS_COUNT,
      coilsCnt := COILS_COUNT,
      inputRegCnt := INPUT_REG_COUNT,
      holdingRegCnt := HOLDING_REG_COUNT);
    inputs : ARRAY [1..INPUTS_COUNT] OF BOOL;
    coils  : ARRAY [1..COILS_COUNT]  OF BOOL;
    inputRegs : ARRAY [1..INPUT_REG_COUNT] OF UINT;
    holdingRegs : ARRAY [1..HOLDING_REG_COUNT] OF UINT;
  END_VAR

  ModbusRTUslave(
    inputs := inputs[1],
    coils := coils[1],
    inputRegs := inputRegs[1],
    holdingRegs := holdingRegs[1]);

END_PROGRAM
```

TXV 003 52.01