Komunikace AMREG s řídicími systémy AMiT (MODBUS RTU)

Abstrakt

Parametrizace regulátorů AMREG komunikujících jako master / slave v síti MODBUS RTU s řídicími systémy.

Autor: Petr Latina, Zbyněk Říha
Dokument: ap0055_cz_01.pdf

Příloha

Obsah souboru: ap0055_cz_01.zip

master_rtu_amr_p1_cz_01.dso	Příklad parametrizace AMREG – master.
slave_rtu_rs_p2_cz_01.dso	Příklad parametrizace řídicího systému – slave.
slave_rtu_amr_p3_cz_01.dso	Příklad parametrizace AMREG – slave (objekt ModbusSlave).
master_rtu_rs_p4_cz_01.dso	Příklad parametrizace řídicího systému – master (pro p3)
slave_rtu_amr_p5_cz_01.dso	Příklad parametrizace AMREG – slave (objekt SerialBusN).
master_rtu_rs_p6_cz_01.dso	Příklad parametrizace řídicího systému – master (pro p5).

Obsah

Historie revizí

Verze	Datum	Autor změny	Změny
001	26. 06. 2014	Latina Petr Říha Zbyněk	Nový dokument.

Související dokumentace

1. Nápověda k vývojovému prostředí DetStudio
   soubor: Ovladani_cs.chm
2. Nápověda k obrazovkám vývojového prostředí DetStudio
   soubor: Tridet_cs.chm
3. Nápověda k části PseDet vývojového prostředí DetStudio
   soubor: Psedet_cs.chm
4. Nápověda k části EsiDet vývojového prostředí DetStudio
   soubor: Esidet_cs.chm
5. Aplikační poznámka AP0008 – Komunikace v síti MODBUS
   soubor: ap0008_cz_xx.pdf
6. Aplikační poznámka AP0016 – Zásady používání RS485
   soubor: ap0016_cz_xx.pdf
7. www.modbus.org – protokol MODBUS RTU

1 Definice použitých pojmů

Master
Zařízení (v případě sítě MODBUS RTU pouze jedno), které aktivně zasílá dotazy / pokyny jednotlivým periferiím v síti. V roli mastera je tedy nadřazený systém.

Slave
Zařízení, která do sítě aktivně nezasílají žádné dotazy ani pokyny. Na přijaté dotazy / pokyny pouze odpovídají (v případě, že jsou mu adresovány). V roli slave jsou ovládaná / sledovaná zařízení.

Datový bod
Hodnota přenášená prostřednictvím protokolu MODBUS RTU. V rámci této aplikační poznámky se pojmem datový bod rozumí uchovávací (holding) registry, vstupní (input) registry, cívky (coils) a diskrétní vstupy (discrete inputs). Každý druh datových bodů má vlastní, nezávislou řadu adres začínajících od 0 (viz popis protokolu MODBUS RTU).

Registr
16 bitová hodnota, přenášená prostřednictvím sítě MODBUS RTU.

2 AMREG – master

K datu vytvoření této aplikační poznámky nelze jako master v síti MODBUS RTU definovat regulátor **AMR-DI2RDO2, AMR-UI2RDO2** a programovatelné nástěnné ovladače řady **AMR-OP3x** a **AMR-OP4x**.

V této aplikační poznámce bude jako master v síti MODBUS RTU použit regulátor **AMR-OP84**.

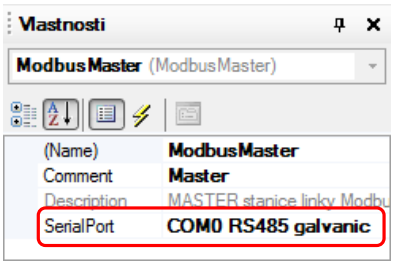
2.1 SW parametrizace

2.1.1 Komunikační objekty

Pro parametrizaci regulátoru AMREG komunikujícího jako master v síti MODBUS RTU je nutné vložit do projektu objekty:

- **`ModbusMaster`** (definuje v regulátoru AMREG protokol MODBUS RTU),
- **`ModbusDevice`** (definuje jedno zařízení slave, se kterými má regulátor AMREG komunikovat).

V panelu „Vlastnosti“ objektu **`ModbusMaster`** je nutné nastavit pouze položku **`SerialPort`**, viz Obr. 1.

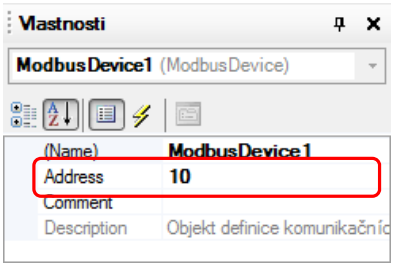


Obr. 1 – Panel „Vlastnosti“ objektu **`ModbusMaster`**

Význam položky, označené na Obr. 1:

SerialPort
Výběr COM portu regulátoru pro komunikaci protokolem MODBUS RTU.

Pro každé zařízení slave, se kterým má regulátor AMREG komunikovat je nutné do projektu vložit jeden objekt **`ModbusDevice`**. V panelu „Vlastnosti“ každého objektu **`ModbusDevice`** je nutné nadefinovat pouze položku **`Address`**, viz Obr. 2. Položku **`Address`** musí mít každý objekt nadefinovánu jako jedinečnou.



Obr. 2 – Panel vlastnosti objektu **`ModbusDevice`**

Význam položky označené na Obr. 2:

Address
Adresa slave zařízení se kterým bude regulátor AMREG komunikovat.

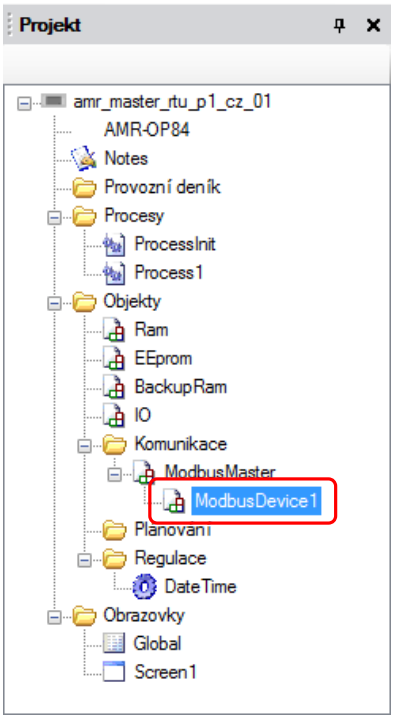
Aby fungovala komunikace v síti MODBUS RTU správně, musí mít každá stanice v síti nastavenu stejnou komunikační rychlost, stejnou paritu a každý slave musí mít nastavenu jedinečnou adresu. Regulátorům AMREG lze adresu, komunikační rychlost a paritu v síti MODBUS RTU nastavit DIP přepínačem (pokud jej regulátor má). Pokud regulátor AMREG DIP přepínač nemá, lze parametry nastavit následovně:

- Pomocí SW AMRConfig (součást instalace prostředí DetStudio).
- Pomocí skriptu EsiDetu, např. v procesu **`ProcessInit`**.
- Prostřednictvím displeje (pouze u vybraných ovladačů typu AMREG).

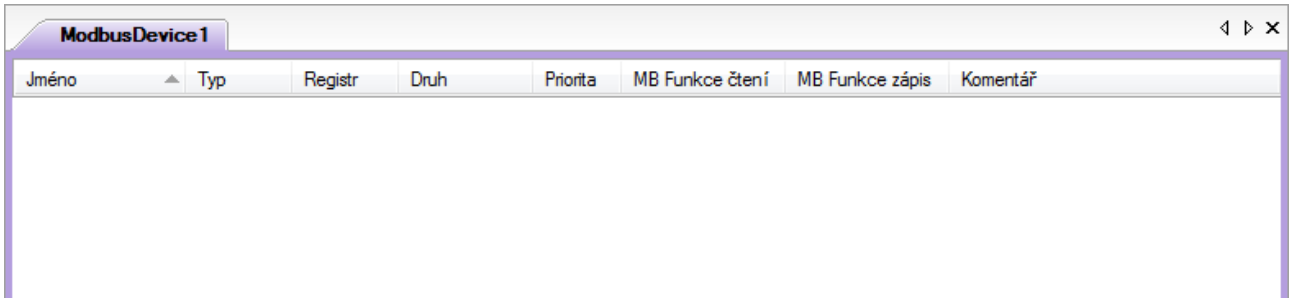
Více informací o nastavení komunikačních parametrů regulátorů AMREG lze nalézt v nápovědě EsiDet.

2.1.2 Definice datových bodů pro komunikaci

Datové body, které se mají s jednotlivými slave zařízeními komunikovat se definují v otevřené záložce objektu **`ModbusDevice`**. Záložka se otevře dvojklikem myší nad konkrétním objektem **`ModbusDevice`** v panelu „Projekt“. Viz Obr. 3.



Obr. 3 – Vybraný objekt **ModbusDevice1**



Obr. 4 – Otevřená záložka objektu **ModbusDevice1**

K datu vytvoření této aplikační poznámky lze definovat následující datové body:

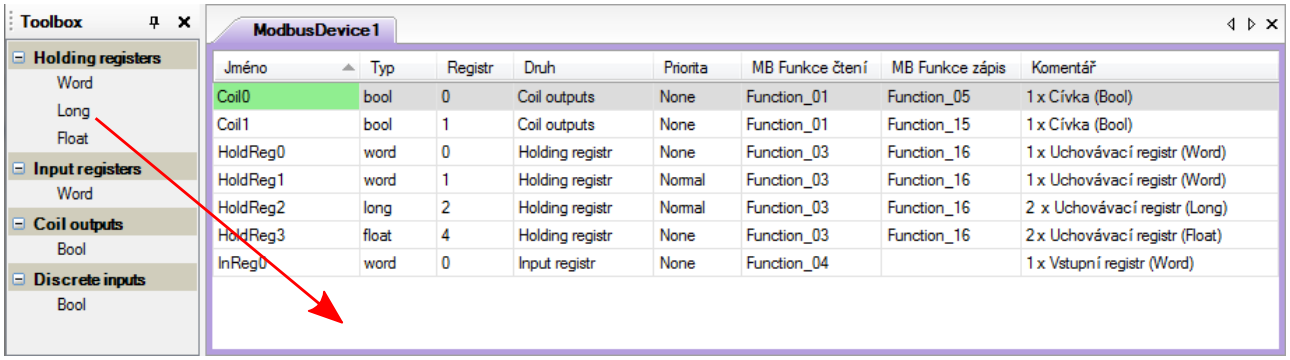
- Uchovávací registry (holding registers) typu Word, Long, Float *),
- Vstupní registry (input registers) typu Word,
- Cívky (coils) typu Bool,
- Diskrétní vstupy (discrete inputs) typu Bool.

*) *Typy Long a Float jsou ukládány do dvojice po sobě jdoucích registrů.*

Podporované funkce pro komunikaci v síti MODBUS RTU jsou:

Funkce	Význam
1	Read coils – čtení stavu cívek
2	Read discrete inputs – čtení diskrétních vstupů
3	Read multiple holding registers – čtení uchovávacích registrů
4	Read input registers – čtení vstupních registrů
5	Write single coil – zápis do jedné cívky
6	Write single holding register – zápis do jednoho uchovávacího registru
15	Write multiple coils – zápis do více cívek
16	Write multiple holding registers – zápis do více uchovávacích registrů

Jednotlivé datové body se definují přetažením požadovaného typu pomocí myši z panelu „Toolbox“ do otevřené záložky objektu **`ModbusDevice`**, viz následující obrázek.



Obr. 5 – Definice datových bodů v záložce objektu **`ModbusDevice1`**

Význam jednotlivých sloupců záložky objektu **`ModbusDevice1`** je následující:

Jméno
Jméno datového bodu, pod kterým bude využíván v rámci projektu. Lze editovat pomocí klávesy ***„F2“***.

Typ
Typ datového bodu (Word, Long, …). Nelze editovat – předvyplněn po vložení datového bodu z panelu „Toolbox“.

Registr
Adresa datového bodu na zařízení typu slave. Lze editovat např. pomocí klávesy ***„F2“***. Datové body Bool (jedna cívka / diskrétní vstup) a Word (jeden vstupní / uchovávací registr) zabírají vždy pouze jednu adresu. Datové body typu Long a Float (dva uchovávací registry) zabírají vždy adresy dvě.

Druh
Druh datového bodu (uchovávací registr, cívka, …). Nelze editovat – předvyplněn po vložení datového bodu z panelu „Toolbox“.

Priorita
Priorita komunikace. Jednotlivým datovým bodům lze nastavit (klávesou ***„F2“***) následující úrovně:

- **None** – v tomto případě se požadavek na komunikaci datového bodu aktivuje na událost, kterou si programově řídí tvůrce kódu ve skriptu procesu, nebo se požadavek na komunikaci aktivuje v případě, kdy daný regulátor má displej, kde je umístěn prvek, který umožňuje zobrazit / editovat hodnotu navázané skalární proměnné / buňky matice. V tomto případě je

perioda nastavení požadavku na komunikaci odvozena od vlastnosti **`RefreshPeriod`** zobrazené obrazovky na displeji (viz text níže).

- **Low** – automatické vkládání požadavků. Jestliže je v projektu definován proces s periodou větší jak 5000 ms, použije se perioda tohoto procesu. V ostatních případech je použita perioda 5000 ms.
- **Normal** – automatické vkládání požadavků s periodou 900 ms.
- **High** – automatické vkládání požadavků. Jestliže je v projektu definován proces s periodou menší než 200 ms, použije se perioda tohoto procesu. V ostatních případech je použita perioda 200 ms.

MB Funkce čtení
Číslo MODBUS funkce, která bude využívána při čtení datového bodu. Nelze editovat – je nastaveno automaticky, v závislosti na druhu datového bodu.

MB Funkce zápis
Číslo MODBUS funkce, která bude využívána pro zápis do datového bodu. Lze editovat pomocí klávesy ***„F2“***.

Komentář
Poznámka k datovému bodu. Lze editovat např. pomocí klávesy ***„F2“***.

2.2 Ukázková aplikace pro AMREG – master

Součástí této aplikační poznámky je ukázková aplikace pro **AMR-OP84**. Jedná se o soubor **master_rtu_amr_p1_cz_xx.dso**. V aplikaci jsou výše uvedeným postupem nadefinovány datové body dle Obr. 6. Tyto registry mapuje do sítě MODBUS RTU slave zařízení – řídicí systém firmy AMiT.

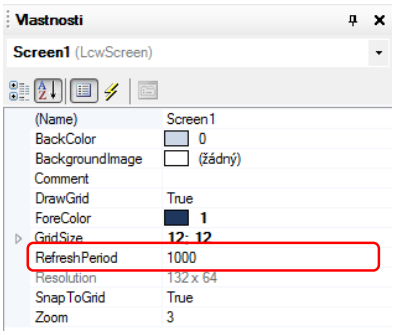
ModbusDevice1

Jméno	Typ	Registr	Druh	Priorita	MB Funkce čtení	MB Funkce zápis	Komentář
Coil0	bool	0	Coil outputs	None	Function_01	Function_05	1 x Cívka (Bool)
Coil1	bool	1	Coil outputs	None	Function_01	Function_15	1 x Cívka (Bool)
HoldReg0	word	0	Holding registr	None	Function_03	Function_16	1 x Uchovávací registr (Word)
HoldReg1	word	1	Holding registr	Normal	Function_03	Function_16	1 x Uchovávací registr (Word)
HoldReg2	long	2	Holding registr	Normal	Function_03	Function_16	2 x Uchovávací registr (Long)
HoldReg3	float	4	Holding registr	None	Function_03	Function_16	2 x Uchovávací registr (Float)
InReg0	word	0	Input registr	None	Function_04		1 x Vstupní registr (Word)

Obr. 6 – Nadefinované datové body

2.2.1 Význam / použití nadefinovaných datových bodů

Registry InReg0 a HoldReg0
Registry **`InReg0`** a **`HoldReg0`** mají nastavenu prioritu typu **None**. Tyto registry jsou navázány na prvky „NumericView“ a „NumericEdit“ obrazovky „Screen1“, která má v panelu „Vlastnosti“ nastavenu vlastnost **`RefreshPeriod`** na hodnotu 1000 ms. Je-li tato obrazovka aktuálně zobrazena, budou se s periodou 1000 ms vkládat požadavky na jejich komunikaci.



Obr. 7 – Hodnota vlastnosti **RefreshPeriod** obrazovky „Screen1“

Registry HoldReg1 a HoldReg2

Registry **HoldReg1** a **HoldReg2** mají nastavenu prioritu typu **Normal**. Automaticky se tedy budou komunikovat s periodou 900 ms.

Registr **HoldReg1** je použit ve skriptu procesu **Process1**. Následující příklad zápisu ukazuje přiřazení hodnoty registru do lokální proměnné vytvořené v paměťovém prostoru **Ram**.

```
Ram.Stavy = ModbusDevice1.HoldReg1;
```

Registr **HoldReg2** je ve skriptu procesu **Process1** plněn hodnotou pomocného čítače následovně.

```
ModbusDevice1.HoldReg2 = Ram.Citac;
```

Registr HoldReg3

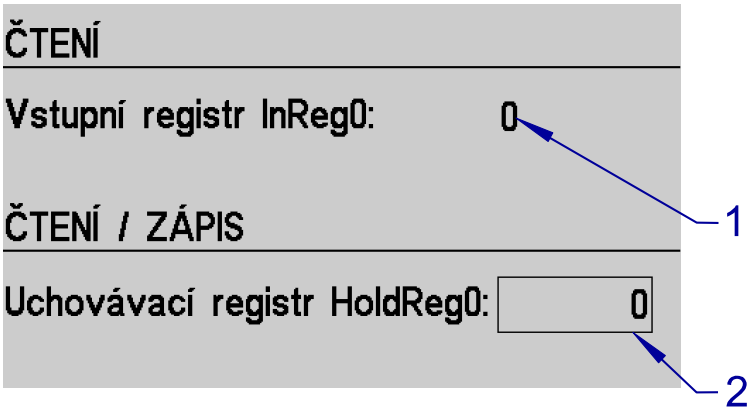
Registr **HoldReg3** je v ukázce využit k událostnímu čtení, které se vykoná po nastavení bitu č. 0 lokální proměnné **Ram.cti**. Jakmile bude nastaven bit č. 0 proměnné **Ram.cti**, bude vykonán ve skriptu procesu kód, který vyvolá požadavek na čtení registru. V následujícím běhu procesu bude do lokální proměnné **Ram.Reg_Flt** uložena hodnota načteného registru.

```
if Ram.cti.0 then
     Ram.cti.0 = false;
     ModbusDevice1.HoldReg3.Refresh();
else
     Ram.Reg_Flt = ModbusDevice1.HoldReg3;
endif;
```

Cívky Coil1 a Coil2

Cívky **Coil1** a **Coil2** jsou použity pro událostní zápis, který se vykoná na nastavení bitu č. 0 lokální proměnné **Ram.zapis**. Jakmile bude nastaven bit č. 0 proměnné **Ram.zapis**, bude vykonán ve skriptu procesu kód, který vyvolá požadavek na zápis do cívek.

```
If Ram.zapis.0 then
     Ram.zapis.0 = false;
     ModbusDevice1.Coil0 = Ram.civky.0;
     ModbusDevice1.Coil1 = Ram.civky.1;
endif;
```

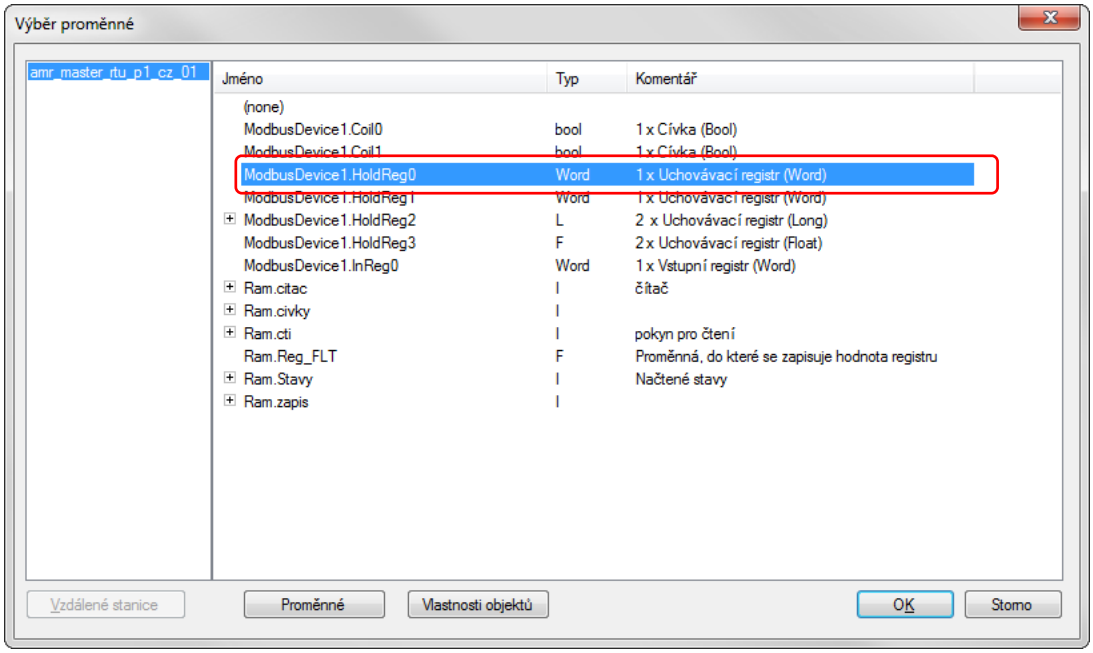


Obr. 8 – Popis položek obrazovky „Screen1“

Legenda

Číslo	Význam
1	Zobrazení hodnoty registru **`InReg0`**
2	Zobrazení / editace hodnoty registru **`HoldReg0`**

Navázání datových bodů / proměnných na obrazovkové prvky pro zobrazení / editaci hodnoty lze provést např. tak, že v návrhu obrazovky se dvojklikem myší nad vybraným prvkem otevře okno „Výběr proměnné“, ve kterém se vybere požadovaný datový bod (proměnná) a následně se potvrdí tlačítkem ***„OK“***, viz následující obrázek.



Obr. 9 – Okno „Výběr proměnné“

Poznámka
Pokud jsou datové body navázány pouze na obrazovkové prvky typu „NumericView“ nebo „NumericEdit“ a nejsou využívány pro regulační algoritmy v některém z periodických procesů, doporučujeme v takovém případě nastavit vždy prioritu ***None****.*

2.3 Stav komunikace

Pro vyhodnocení stavu komunikace lze v regulátoru AMREG využít několik vlastností objektů **`ModbusMaster`** a **`ModbusDevice`**, které lze navázat např. na obrazovkové prvky nebo je vyhodnocovat v některém z periodických procesů.

Jedná se o následující vlastnosti:

- **`Disconnected`**,
- **`FrameErrorCounter`**,
- **`FrameOKCounter`**.

Více informací lze nalézt v nápovědě EsiDet u konkrétních komunikačních objektů.

2.4 Ukázková aplikace pro řídicí systém firmy AMiT – slave

Ukázková aplikace (pro řídicí systém) má nadefinovány proměnné, které jsou mapovány do sítě MODBUS RTU v podobě registrů a cívek. Způsoby mapování jsou podrobně popsány v nápovědě k vývojovému prostředí DetStudio (část PseDet).

Ukázková aplikace (soubor **slave_rtu_rs_p2_cz_xx.dso**) je vytvořena pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovou komunikační linkou, pomocí menu DetStudia „Nástroje / Změnit typ stanice…“.

2.4.1 Význam proměnných založených v řídicím systému

Níže uvedená tabulka obsahuje seznam proměnných, založených v řídicím systému, které jsou mapovány do sítě MODBUS RTU.

Proměnná	Typ	Komentář
HoldReg0	I	Uchovávací registr – hodnota je čtena / editována masterem s periodou aktualizace obrazovky.
HoldReg1	I	Uchovávací registr – hodnota je čtena masterem periodicky.
HoldReg2	L	Uchovávací registr – hodnota je zapisována masterem periodicky.
HoldReg3	F	Uchovávací registr – hodnota je čtena masterem událostně.
InReg0	I	Vstupní registr – hodnota je čtena masterem s periodou aktualizace obrazovky.
Coils	I	Cívky – hodnota je zapisována masterem událostně.

3 AMREG – slave

Všechny regulátory AMREG a programovatelné nástěnné ovladače z produkce firmy AMiT lze použít jako slave zařízení na lince MODBUS RTU. Pro parametrizaci regulátoru AMREG komunikujícího v síti MODBUS RTU jako slave lze použít jeden z objektů:

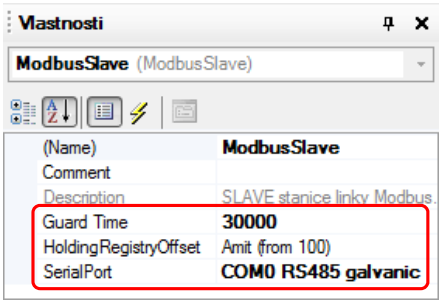
- **`ModbusSlave`** – umožňuje definovat různé typy datových bodů (registry, cívky, diskrétní vstupy) z protokolu MODBUS RTU.
- **`SerialBusN`** – pro síť MODBUS RTU umožňuje definovat pouze uchovávací registry. S výhodou jej však lze využít pro případy, kdy je požadavek na tvorbu uživatelské aplikace, která bude poskytovat obdobná data jak do sítě MODBUS RTU, tak do sítě ARION (objekt **`SerialBusN`** poskytuje nadefinované registry jak do sítě MODBUS RTU, tak do sítě ARION a lze mezi protokoly přepínat).

3.1 SW parametrizace – objekt ModbusSlave

Jako slave zařízení, které je připojeno do sítě MODBUS RTU, je použit regulátor **AMR-OP84**.

3.1.1 Komunikační objekt

Po vložení objektu **`ModbusSlave`** do projektu, se v jeho panelu „Vlastnosti“ nastaví položky označené na Obr. 10.



Obr. 10 – Panel „Vlastnosti“ objektu **`ModbusSlave`**

Význam položek, označených na Obr. 10:

Guard Time
Čas, do kterého musí regulátor AMREG obdržet jakýkoliv platný komunikační MODBUS RTU rámec (může být i s adresou jiného zařízení v síti). Pokud do této doby rámec neobdrží, bude signalizovat rozpad komunikace.

HoldingRegistryOffset
Volba, zda číslování uchovávacích registrů začíná od 0 nebo od 100 (definice zařízení typu slave z produkce firmy AMiT).

SerialPort
Výběr COM portu regulátoru pro komunikaci protokolem MODBUS RTU.

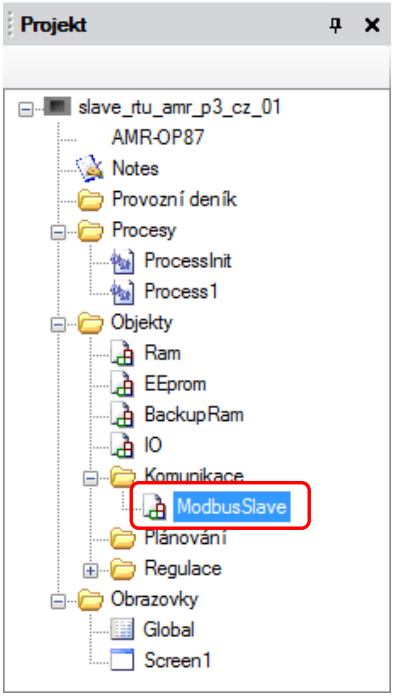
Aby fungovala komunikace v síti MODBUS RTU správně, musí mít každá stanice v síti nastavenu stejnou komunikační rychlost, stejnou paritu a každý slave musí mít nastavenu jedinečnou adresu. Regulátorům AMREG lze adresu, komunikační rychlost a paritu v síti MODBUS RTU nastavit DIP přepínačem (pokud jej regulátor má). Pokud regulátor AMREG DIP přepínač nemá, lze parametry nastavit následovně:

- Pomocí SW AMRConfig (součást instalace prostředí DetStudio).
- Pomocí skriptu EsiDetu, např. v procesu **`ProcessInit`**.
- Prostřednictvím displeje (pouze u vybraných ovladačů typu AMREG).

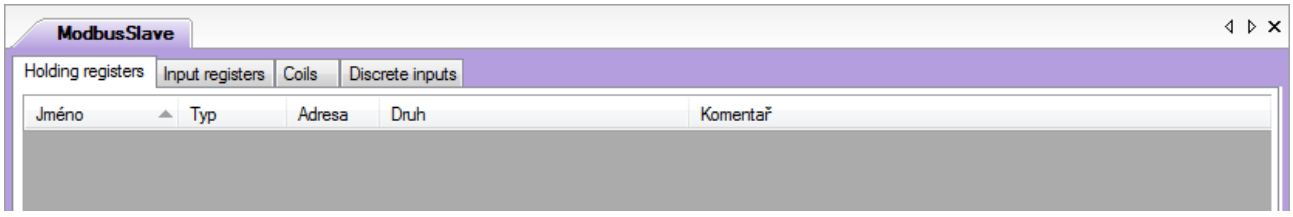
Více informací o nastavení komunikačních parametrů regulátorů AMREG lze nalézt v nápovědě EsiDet.

3.1.2 Definice datových bodů

Datové body se definují v otevřené záložce objektu **`ModbusSlave`**. Záložka se otevře dvojklikem levým tlačítkem myši nad objektem **`ModbusSlave`** v panelu „Projekt“. Viz následující obrázek.



Obr. 11 – Vybraný objekt **`ModbusSlave`**



Obr. 12 – Otevřená záložka objektu **`ModbusSlave`**

K datu vytvoření této aplikační poznámky lze definovat následující datové body:

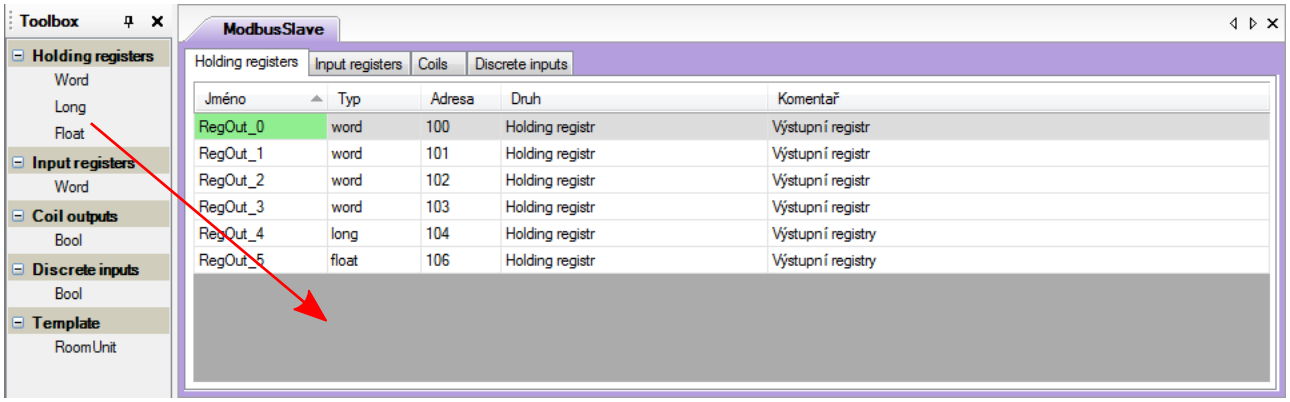
- Uchovávací registry (holding registers) typu Word, Long, Float *),
- Vstupní registry (input registers) typu Word,
- Cívky (coils) typu Bool,
- Diskrétní vstupy (discrete inputs) typu Bool.

*) *Typy Long a Float jsou ukládány do dvojice po sobě jdoucích registrů.*

Podporované funkce pro komunikaci v síti MODBUS RTU jsou:

Funkce	Význam
1	Read coils – čtení stavu cívek
2	Read discrete inputs – čtení diskrétních vstupů
3	Read multiple holding registers – čtení uchovávacích registrů
4	Read input registers – čtení vstupních registrů
5	Write single coil – zápis do jedné cívky
6	Write single holding register – zápis do jednoho uchovávacího registru
15	Write multiple coils – zápis do více cívek
16	Write multiple holding registers – zápis do více uchovávacích registrů

Jednotlivé datové body se definují přetažením požadovaného typu pomocí myši z panelu „Toolbox“ do otevřené záložky objektu **`ModbusSlave`**, viz následující obrázek.



Obr. 13 – Definice datových bodů v záložce objektu **`ModbusSlave`**

Význam jednotlivých sloupců záložky objektu **`ModbusSlave`** je následující:

Jméno
Jméno datového bodu, pod kterým bude využíván v rámci projektu. Lze editovat pomocí klávesy ***„F2“***.

Typ
Typ datového bodu (Word, Long, …). Nelze editovat – předvyplněn po vložení datového bodu z panelu „Toolbox“.

Adresa
Adresa datového bodu na zařízení typu slave. Lze editovat např. pomocí klávesy ***„F2“***. Prvních 100 (0 až 99) uchovávacích registrů je určeno pro systémové účely. Od čísla 100 jsou přístupné uživateli. Datové body Bool a Word lze definovat v číselném pořadí např. 100, 101, 102, 103 atd. Datové body typu Long a Float lze definovat jako dvojici uchovávacích registrů.

Druh
Druh datového bodu (uchovávací registr, cívka, …). Nelze editovat – předvyplněn po vložení datového bodu z panelu „Toolbox“.

Komentář
Poznámka k datovému bodu. Lze editovat např. pomocí klávesy ***„F2“***.

3.2 Ukázková aplikace pro AMREG – slave

Součástí této aplikační poznámky je ukázková aplikace pro regulátor AMREG jako slave. Jedná se o soubor **slave_rtu_amr_p3_cz_xx.dso**. V aplikaci jsou výše uvedeným postupem definovány různé datové body.

3.2.1 Význam / použití nadefinovaných datových bodů

Uchovávací registry

ModbusSlave

Holding registers | Input registers | Coils | Discrete inputs

Jméno	Typ	Adresa	Druh	Komentař
RegOut_0	word	100	Holding registr	Uchovávací registr
RegOut_1	word	101	Holding registr	Uchovávací registr
RegOut_2	word	102	Holding registr	Uchovávací registr
RegOut_3	word	103	Holding registr	Uchovávací registr
RegOut_4	long	104	Holding registr	Uchovávací registry
RegOut_5	float	106	Holding registr	Uchovávací registry

Obr. 14 – Uchovávací registry

Hodnoty uchovávacích registrů jsou nastavovány masterem v síti MODBUS RTU (**`RegOut`**_0 až **`RegOut`**_5). Na displeji lze sledovat hodnoty registrů **`RegOut`**_0, **`RegOut`**_1, **`RegOut`**_4 a **`RegOut`**_5. Viz Obr. 18.

Vstupní registry

ModbusSlave

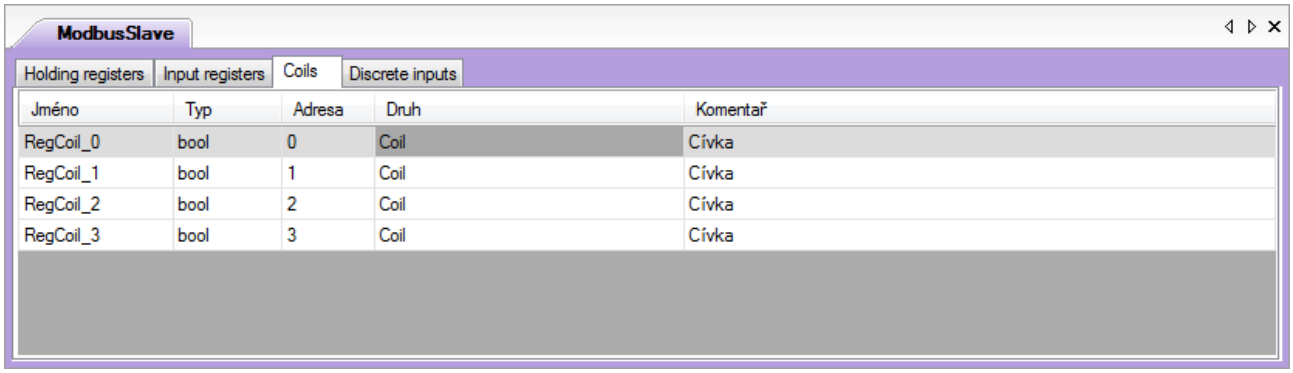
Holding registers | Input registers | Coils | Discrete inputs

Jméno	Typ	Adresa	Druh	Komentař
RegIn_0	word	0	Input registr	Vstupní registr
RegIn_1	word	1	Input registr	Vstupní registr
RegIn_2	word	2	Input registr	Vstupní registr
RegIn_3	word	3	Input registr	Vstupní registr

Obr. 15 – Vstupní registry

Hodnoty vstupních registrů jsou čteny masterem v síti MODBUS RTU. Zároveň je lze nastavovat buď z displeje regulátoru (**RegIn**_0 a **RegIn**_1), viz Obr. 18, nebo je hodnota registrů nastavována ve skriptu procesu **Process1** (**RegIn**_2 a **RegIn**_3).

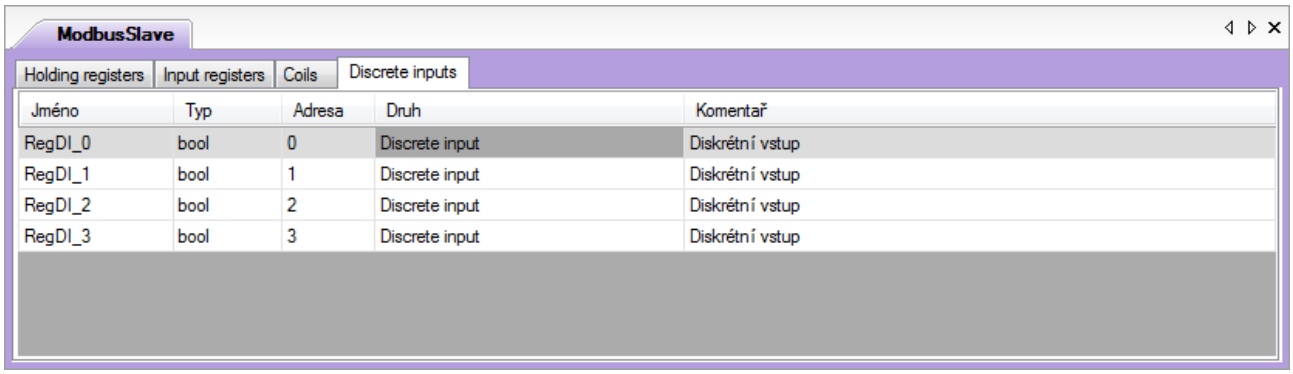
Cívky



Obr. 16 – Cívky

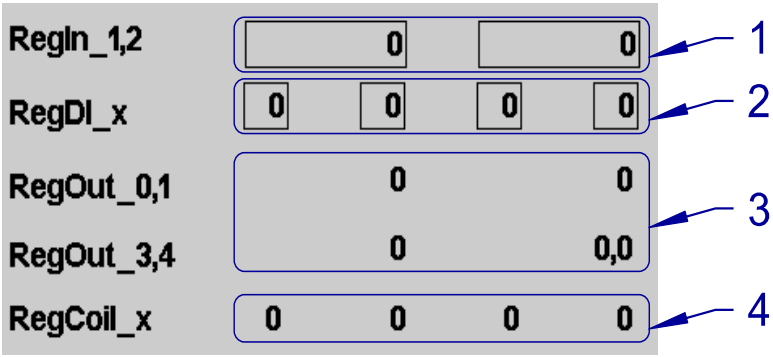
Stavy cívek jsou nastavovány masterem na síti MODBUS RTU. Lze je také sledovat na displeji regulátoru, viz Obr. 18.

Diskrétní vstupy



Obr. 17 – Diskrétní vstupy

Stavy diskrétních vstupů jsou čteny masterem v síti MODBUS RTU. Zároveň je lze nastavovat z displeje regulátoru, viz Obr. 18.



Obr. 18 – Popis položek obrazovky

Legenda

Číslo	Význam
1	Vstupní registry, editace
2	Diskrétní vstupy, editace
3	Uchovávací registry, zobrazení hodnoty
4	Cívky, zobrazení hodnoty

Navázání registrů pro zobrazení nebo editaci jejich hodnoty na příslušné obrazovkové prvky, je ukázáno výše v kapitole „2.2.1 Význam / použití nadefinovaných “.

3.3 Stav komunikace

Pro vyhodnocení stavu komunikace lze v regulátoru AMREG využít několik vlastností objektu **`ModbusSlave`**, které lze navázat např. na obrazovkové prvky nebo je vyhodnocovat v některém z periodických procesů.

Jedná se o následující vlastnosti:

- **`Connected`**,
- **`Incoming`**,
- **`Outgoing`**.

Více informací lze nalézt v nápovědě EsiDet u objektu **`ModbusSlave`**.

3.4 Ukázková aplikace pro řídicí systém firmy AMiT – master

Ukázková aplikace pro řídicí systém firmy AMiT čte / zapisuje z / do datových bodů nadefinovaných ve slave stanici. Parametrizace řídicích systémů firmy AMiT pro komunikaci v síti MODBUS RTU jako Master je podrobně popsána v aplikační poznámce AP0008.

Ukázková aplikace (soubor **master_rtu_rs_p4_cz_xx.dso**) je vytvořena pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovou komunikační linkou, pomocí menu DetStudia „Nástroje / Změnit typ stanice…“.

3.4.1 Význam proměnných založených v řídicím systému

Význam proměnných založených v řídicím systému firmy AMiT je uveden v následující tabulce.

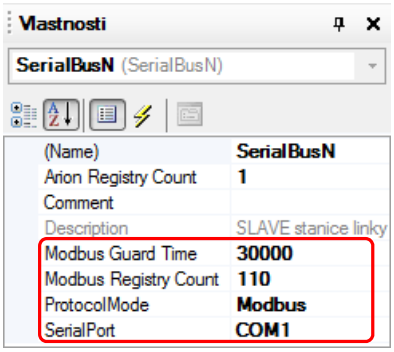
Proměnná	Typ	Komentář
MB_Aloc	I	Počet datových bodů správně nadefinovaných v řídicím systému.
MB_Count	I	Počet datových bodů nadefinovaných v řídicím systému.
Coils	I	Cívky, do kterých řídicí systém zapisuje.
DI	I	Diskrétní vstupy, ze kterých řídicí systém čte stav.
InReg	MI	Vstupní registry, ze kterých řídicí systém čte hodnotu.
HoldReg	MI	Uchovávací registry, do kterých řídicí systém zapisuje hodnoty.
HoldRegF	F	Uchovávací registry, do kterých řídicí systém zapisuje jednu hodnotu typu Float.
HoldRegL	L	Uchovávací registry, do kterých řídicí systém zapisuje jednu hodnotu typu Long.

3.5 SW parametrizace – objekt SerialBusN

Jako slave zařízení, které je připojeno do sítě MODBUS RTU, je použit regulátor **AMR-FCT10**.

3.5.1 Komunikační objekt

Po vložení objektu **`SerialBusN`** do projektu, se v jeho panelu „Vlastnosti“ nastaví položky označené na Obr. 19.



Obr. 19 – Okno „Vlastnosti“ objektu **`SerialBusN`**

Význam položek, označených na Obr. 19:

Modbus Guard Time
Čas, do kterého musí regulátor AMREG obdržet jakýkoliv platný komunikační MODBUS RTU rámec (může být i s adresou jiného zařízení v síti). Pokud do této doby rámec neobdrží, bude signalizovat rozpad komunikace.

Modbus Registry Count
Počet uchovávacích registrů, které bude regulátor poskytovat do sítě MODBUS RTU (jeden uchovávací registr může obsahovat i různé skupiny bitů z více proměnných typu Long). Prvních 100 (0 až 99) uchovávacích registrů je určeno pro systémové účely. Od čísla 100 jsou přístupné uživateli. Objekt dovoluje nadefinovat pouze sudé množství uchovávacích registrů v posloupnosti 100, 102, 104 atd. (viz nápověda k části EsiDet prostředí DetStudio).

ProtocolMode
Volba komunikačního protokolu, kterým bude zařízení komunikovat. Pro komunikaci protokolem MODBUS RTU, je nutné této položce nastavit komunikační protokol MODBUS.

SerialPort
Výběr COM portu regulátoru pro komunikaci protokolem, který je zvolen v položce **ProtocolMode**.

Ostatní položky se netýkají protokolu MODBUS RTU a není třeba je měnit.

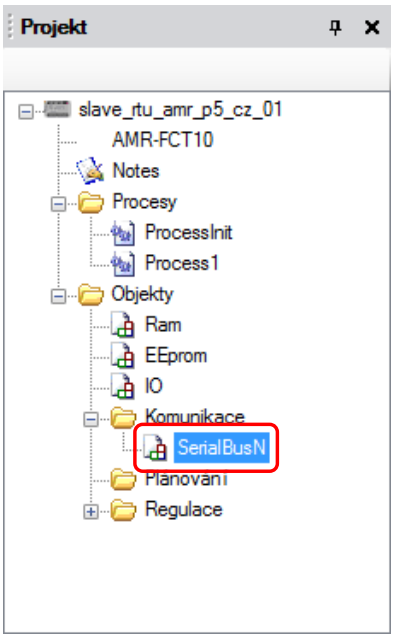
Aby fungovala komunikace v síti MODBUS RTU správně, musí mít každá stanice v síti nastavenu stejnou komunikační rychlost, stejnou paritu a každý slave musí mít nastavenu jedinečnou adresu. Regulátorům AMREG lze adresu, komunikační rychlost a paritu v síti MODBUS RTU nastavit DIP přepínačem (pokud jej regulátor má). Pokud regulátor AMREG DIP přepínač nemá, lze parametry nastavit následovně:

- Pomocí SW AMRConfig (součást instalace prostředí DetStudio).
- Pomocí skriptu EsiDetu, např. v procesu **`ProcessInit`**.
- Prostřednictvím displeje (pouze u vybraných ovladačů typu AMREG).

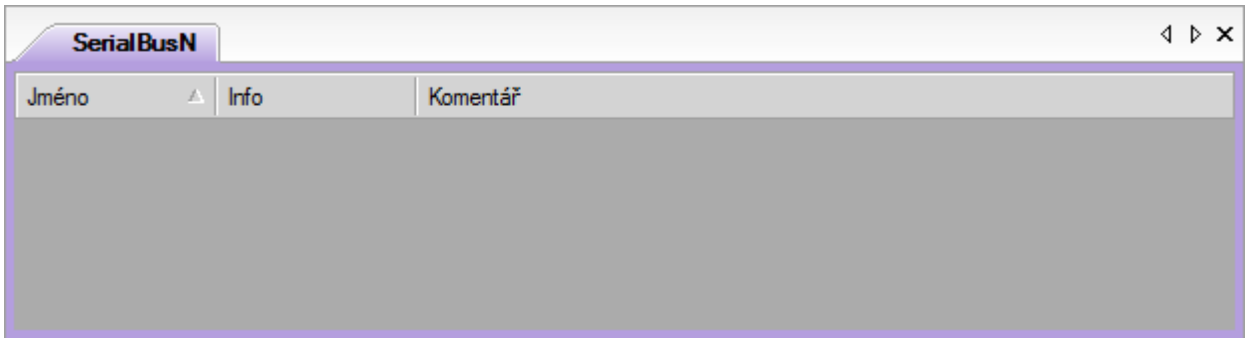
Více informací o nastavení komunikačních parametrů regulátorů AMREG lze nalézt v nápovědě EsiDet.

3.5.2 Mapování proměnných do uchovávacích registrů

Mapování se provádí v otevřené záložce objektu **`SerialBusN`**. Záložka se otevře dvojklikem myší nad objektem **`SerialBusN`** v okně „Projekt“, viz Obr. 20.



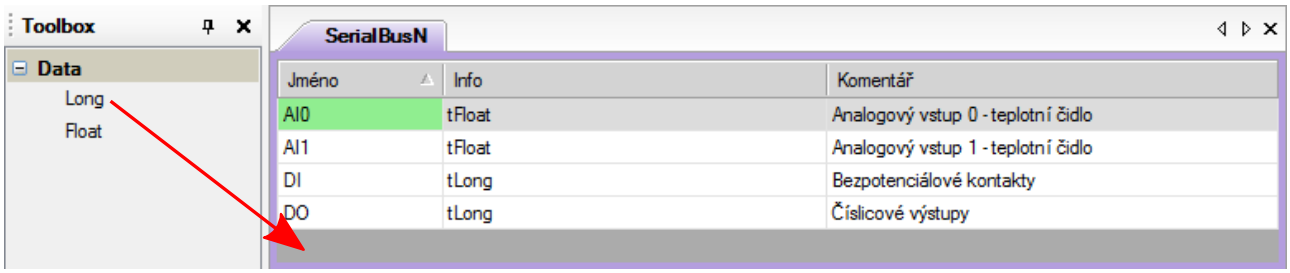
Obr. 20 – Vybraný objekt **`SerialBusN`**



Obr. 21 – Otevřená záložka objektu **SerialBusN**

Do uchovávacích registrů lze mapovat proměnné typu Long nebo proměnné typu Float. Jelikož se jedná o 32bit proměnné, probíhá mapování vždy do dvojice po sobě jdoucích uchovávacích registrů.

Proměnné se mapují přetažením požadovaného typu proměnné pomocí myši z okna „Toolbox“ do otevřené záložky objektu **SerialBusN**, viz Obr. 22. Všechny uchovávací registry jsou vstupně výstupní.



Obr. 22 – Mapování proměnných do komunikačních registrů

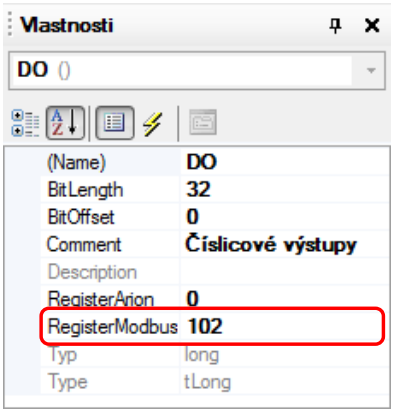
Význam jednotlivých sloupců záložky **SerialBusN** je následující:

Jméno
Jméno proměnné, pod kterým bude využívána v rámci projektu. Lze editovat např. pomocí klávesy *„F2“*.

Info
Typ nadefinované proměnné. Nelze editovat – předvyplněn po vložení typu z panelu „Toolbox“.

Komentář
Poznámka k dané proměnné. Lze editovat např. pomocí klávesy *„F2“*.

Nadefinovaným proměnným je nutné přiřadit číslo počátečního uchovávacího registru, od kterého bude proměnná mapována. Číslo počátečního uchovávacího registru se zadá prostřednictvím položky **RegisterModbus** v panelu „Vlastnosti“ daného registru viz Obr. 23.

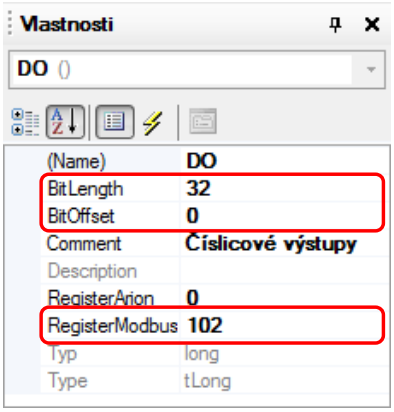


Obr. 23 – Okno „Vlastnosti“ definovaného registru

Do dvojice uchovávacích registrů lze namapovat:

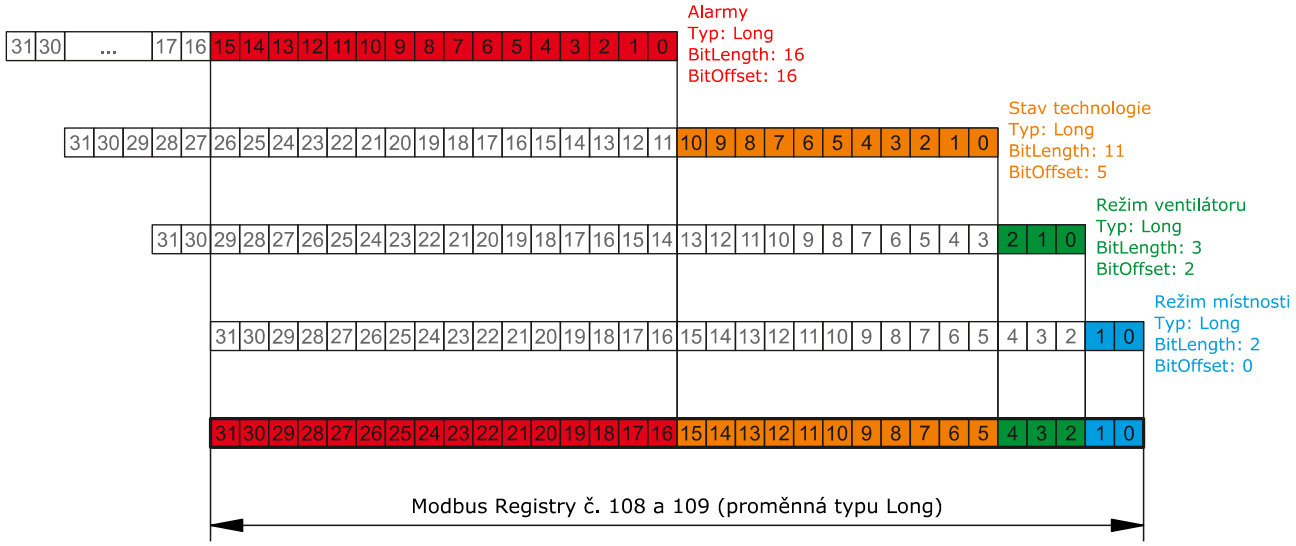
- Jednu proměnnou typu Float.
- Jednu proměnnou typu Long.
- Více nezávislých skupin bitů (až 32) z různých proměnných typu LONG.

Při požadavku na mapování více skupin bitů z různých proměnných do dvojice uchovávacích registrů je nutné u každé proměnné nastavit položce **`RegisterModbus`** stejné číslo a pomocí položek **`BitLength`** a **`BitOffset`** provést vlastní mapování do jednotlivých bitů do dvojice uchovávacích registrů.



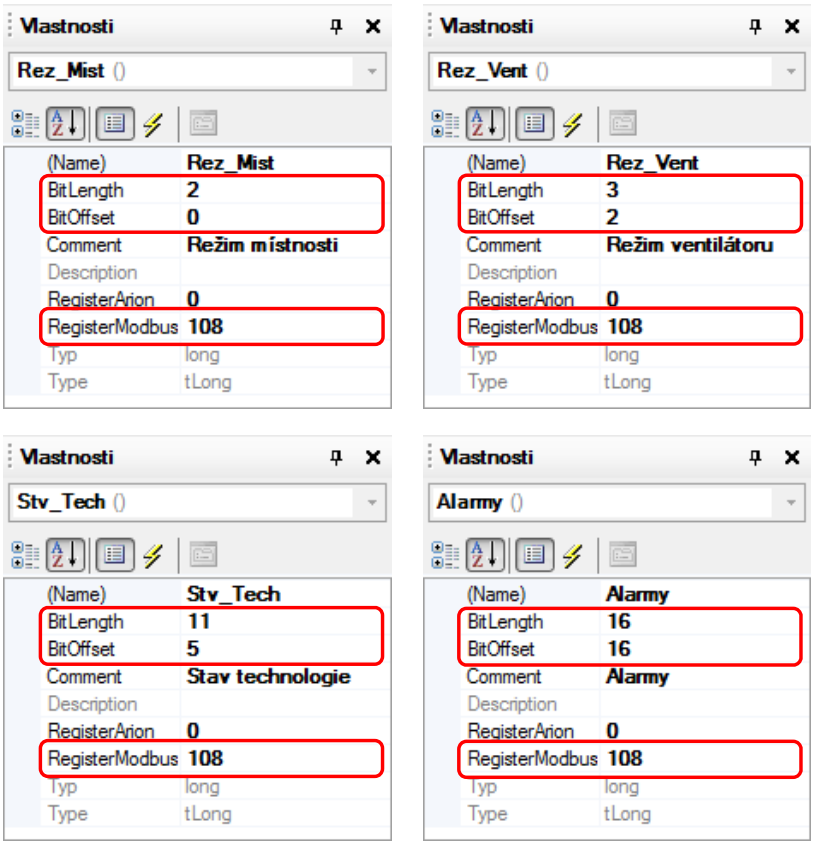
Obr. 24 – Parametry pro mapování výběru bitů do dvojice uchovávacích registrů

Tento postup usnadní práci v případech, kdy je v uživatelské aplikaci potřeba pracovat pouze s některými bity proměnné, nezávisle na sobě nebo je požadavek na přenos více informací v jedné dvojici uchovávacích registrů. Níže uvedený obrázek ukazuje mapování 4 různých proměnných (každá využívá různý počet bitů) do jedné dvojice uchovávacích registrů 108 a 109.



Obr. 25 – Ukázka využití položek BitOffset a BitLength

Mapování různých proměnných do jedné dvojice uchovávacích registrů (viz obrázek výše) v panelu vlastností ukazuje následující obrázek. Každé proměnné odpovídá jeden panel vlastností.



Obr. 26 – Panel vlastností mapovaných proměnných

K jednotlivým proměnným lze v procesech přistupovat přímo pomocí jména **SerialBusN.xxx** následovně:

```
SerialBusN.Rez_Mist = 2;
SerialBusN.Rez_Vent = 4;
SerialBusN.Stav_Tech = 1;
SerialBusN.Alarmy = 0;
```

Obdobným způsobem lze, pomocí jména proměnných, načítat hodnotu zaslanou nadřazeným systémem.

3.6 Ukázková aplikace pro AMREG – slave

Součástí této aplikační poznámky je ukázková aplikace pro regulátor **AMR-FCT10/01**. Jedná se o soubor **slave_rtu_amr_p5_cz_xx.dso**. V aplikaci jsou výše uvedeným postupem nadefinovány proměnné dle Obr. 27, které jsou navázány na uchovávací registry 100 až 109.

SerialBusN

Jméno	Info	Komentář
AI0	tFloat	Analogový vstup 0 - teplotní čidlo
AI1	tFloat	Analogový vstup 1 - teplotní čidlo
Alarmy	tLong	Alarmy
DI	tLong	Bezpotenciálové kontakty
DO	tLong	Číslicové výstupy
Rez_Mist	tLong	Režim místnosti
Rez_Vent	tLong	Režim ventilátoru
Stv_Tech	tLong	Stav technologie

Obr. 27 – Nadefinované proměnné

3.6.1 Význam / použití mapovaných proměnných

Proměnná DI (uchovávací registry č. 100 a 101)
Do proměnné **DI** jsou v periodickém procesu zapisovány stavy univerzálních vstupů, vyhodnocených jako bezpotenciálový kontakt, dle následujícího kódu.

```
// Stav univerzálních vstupů vyhodnocených jako bezpotenciálový kontakt
SerialBusN.DI.0 = IO.DI0;
SerialBusN.DI.1 = IO.DI1;
```

Proměnná DO (uchovávací registry č. 102 a 103)
Jednotlivé bity proměnné **DO** slouží k nastavení požadovaného stavu reléových a triakových výstupů regulátoru. Zápis kódu v periodickém procesu může být následovný.

```
// Zápis na digitální výstupy
IO.RL0 = SerialBusN.DO.0;
IO.RL1 = SerialBusN.DO.1;
IO.RL2 = SerialBusN.DO.2;
IO.TRI0 = SerialBusN.DO.3;
IO.TRI1 = SerialBusN.DO.4;
```

Proměnné AI0 (uchovávací registry č. 104 a 105), AI1 (uchovávací registry č. 106 a 107)
Do proměnných **AI0** a **AI1** je v periodickém procesu zapisována měřená hodnota daného univerzálního vstupu, ke kterému je připojeno teplotní čidlo. Zápis kódu je následující.

```
// Analogové vstupy (připojeny teplotní čidla)
SerialBusN.AI0 = IO.AI0;
SerialBusN.AI1 = IO.AI1;
```

Proměnné Rez_Mist, Rez_Vent, Stv_Tech a Alarmy (uchovávací registry č. 108 a 109)
Aplikačně nejsou nijak využívány. Slouží pouze jako ukázka mapování více proměnných do jednoho registru.

3.7 Stav komunikace

Pro vyhodnocení stavu komunikace lze využít několik vlastností objektu **SerialBusN**, které lze vyhodnocovat v některém z periodických procesů, typicky rozpad komunikace s nadřazeným systémem.

Jedná se o následující vlastnosti:

- **Connect,**
- **Incoming,**
- **Outgoing.**

Více informací lze nalézt v nápovědě EsiDet u objektu **SerialBusN**.

3.8 Ukázková aplikace pro řídicí systém firmy AMiT – master

Ukázková aplikace pro řídicí systém firmy AMiT čte / zapisuje z / do uchovávacích registrů nadefinovaných ve slave stanici. Parametrizace řídicích systémů firmy AMiT pro komunikaci v síti MODBUS RTU jako Master je podrobně popsána v aplikační poznámce AP0008.

Ukázková aplikace (soubor **master_rtu_rs_p6_cz_xx.dso**) je vytvořena pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovou komunikační linkou, pomocí menu DetStudia „Nástroje / Změnit typ stanice…“.

3.8.1 Význam proměnných založených v řídicím systému

Význam proměnných založených v řídicím systému firmy AMiT je uveden v následující tabulce.

Proměnná	Typ	Komentář
MB_Aloc	I	Počet správně nadefinovaných registrů v řídicím systému.
MB_Count	I	Počet nadefinovaných registrů v řídicím systému.
DI	L	Uchovávací registr, ze kterého řídicí systém čte stavy DI regulátoru.
DO	L	Uchovávací registr, do kterého řídicí systém zapisuje požadovaný stav DO regulátoru.
AIx	MF	Uchovávací registry, ze kterých řídicí systém čte hodnotu AI regulátoru.
Stavy	L	Uchovávací registry, ze kterých řídicí systém čte požadované režimy a stavy.
Stavy_zap	L	Uchovávací registry, do kterých řídicí systém zapisuje požadované režimy místnosti a ventilátoru.

4 Technická podpora

Veškeré informace ohledně komunikace regulátorů AMREG s řídicími systémy AMiT, Vám poskytne oddělení technické podpory firmy AMiT. Technickou podporu můžete kontaktovat nejlépe prostřednictvím emailu na adrese **support@amit.cz**.

5 Upozornění

AMiT, spol. s r. o. poskytuje informace v tomto dokumentu, tak jak jsou, nepřejímá žádné záruky, pokud se týče obsahu tohoto dokumentu a vyhrazuje si právo měnit obsah dokumentu bez závazku tyto změny oznámit jakékoli osobě či organizaci.

Tento dokument může být kopírován a rozšiřován za následujících podmínek:

1. Celý text musí být kopírován bez úprav a se zahrnutím všech stránek.
2. Všechny kopie musí obsahovat označení autorského práva společnosti AMiT, spol. s r. o. a veškerá další upozornění v dokumentu uvedená.
3. Tento dokument nesmí být distribuován za účelem dosažení zisku.

V publikaci použité názvy produktů, firem apod. mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.