Komunikace AMREG s řídicími systémy AMiT (ARION)

Abstrakt

Parametrizace regulátorů AMREG komunikujících v síti ARION jako slave.

Autor: Petr Latina, Zbyněk Říha
Dokument: ap0054_cz_01.pdf

Příloha

Obsah souboru: ap0054_cz_01.zip

slave1_amr_p1_cz_01.dso	Parametrizace AMREG – komunikace registry.
master1_rs_p2_cz_01.dso	Parametrizace řídicího systému – komunikace registry.
slave2_amr_p3_cz_01.dso	Parametrizace AMREG – komunikace kanály AI, DI, AO a DO.
master2_rs_p4_cz_01.dso	Parametrizace řídicího systému – komunikace kanály AI, DI, AO, a DO.

Obsah

Historie revizí

Verze	Datum	Autor změny	Změny
001	20. 07. 2016	Latina Petr Říha Zbyněk	Nový dokument.

Související dokumentace

1. Nápověda k vývojovému prostředí DetStudio
   soubor: Ovladani_cs.chm
2. Nápověda k části PseDet vývojového prostředí DetStudio
   soubor: Psedet_cs.chm
3. Nápověda k části EsiDet vývojového prostředí DetStudio
   soubor: Esidet_cs.chm
4. Aplikační poznámka AP0025 – Komunikace v síti ARION – definice tabulkou
   soubor: ap0025_cz_xx.pdf
5. Aplikační poznámka AP0016 – Zásady používání RS485
   soubor: ap0016_cz_xx.pdf

1 Definice použitých pojmů

Master
Zařízení (v případě sítě ARION 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.

Registr
32 bitová hodnota, přenášená prostřednictvím sítě ARION. Registry mohou být typu LONG nebo typu FLOAT.

Signál
Digitální / analogový vstup či výstup.

Kanál
Definuje skupinu signálů. V síti ARION je možné přenášet různé datové typy: analogové vstupy AI, analogové výstupy AO, digitální vstupy DI, digitální výstupy DO a data speciálního typu.

Uzel
Definuje skupiny kanálů rozšiřujícího modulu v síti ARION. Každý rozšiřující modul sítě ARION může být definován jedním nebo několika uzly. Podle toho se rozlišují jednouzlové a více uzlové moduly.

2 Komunikace v síti ARION

Regulátory AMREG mohou v síti ARION komunikovat pouze jako slave zařízení. Data lze s nadřazeným systémem komunikovat prostřednictvím registrů nebo prostřednictvím kanálů AI, DI, AO a DO konkrétního zařízení (uzlu) na síti ARION.

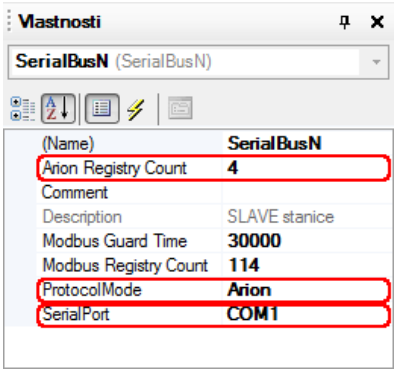
V této aplikační poznámce bude jako slave zařízení v síti ARION použit regulátor **AMR-FCT10**.

3 Komunikace pomocí registrů

3.1 SW parametrizace

3.1.1 Komunikační objekt

Pro komunikaci v síti ARION prostřednictvím registrů je nutné vložit do projektu objekt **`SerialBusN`**. V okně „Vlastnosti“ tohoto objektu se nastaví položky označené na Obr. 1.



Obr. 1 – Okno „Vlastnosti“ objektu **`SerialBusN`**

Arion Registry Count
Počet registrů, které bude regulátor poskytovat do sítě ARION (jeden registr může obsahovat i různé skupiny bitů z více proměnných typu Long). Do sítě ARION lze poskytnout až 9 registrů (číslováno 0 až 8).

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

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

Ostatní položky se netýkají protokolu ARION a není třeba je měnit.

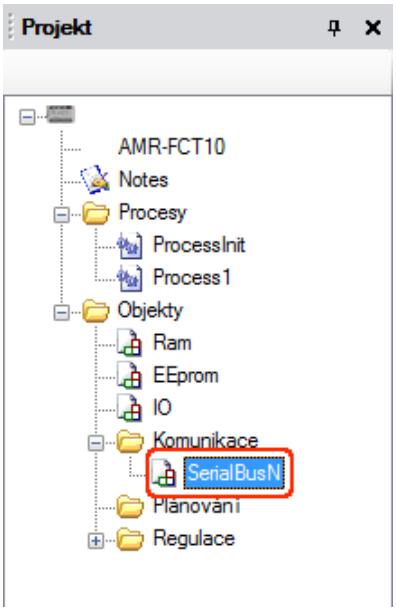
Aby fungovala komunikace v síti ARION správně, musí mít každá stanice v síti nastavenu stejnou komunikační rychlost a každý slave musí mít nastavenu jedinečnou adresu. Regulátorům AMREG lze komunikační rychlost a číslo stanice v síti ARION nastavit DIP přepínačem (pokud jej regulátor má). Pokud regulátor 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).

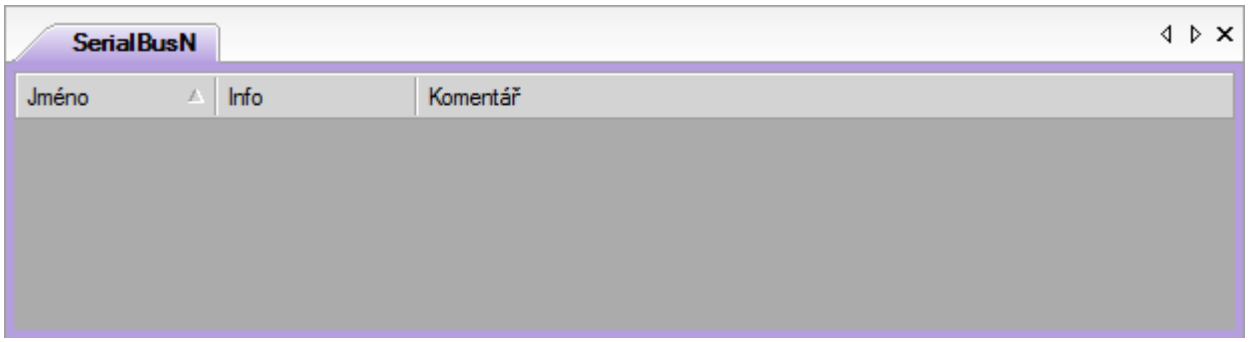
Více informací o nastavení komunikačních parametrů regulátorů AMREG lze nalézt v nápovědě EsiDet.

3.1.2 Mapování proměnných do komunikačních registrů

Do komunikačních registrů lze mapovat proměnné typu Long nebo proměnné typu Float. Mapování se definuje v otevřené záložce objektu **SerialBusN**. Záložka se otevře dvojklikem myší nad objektem **SerialBusN** v okně „Projekt“, viz Obr. 2.

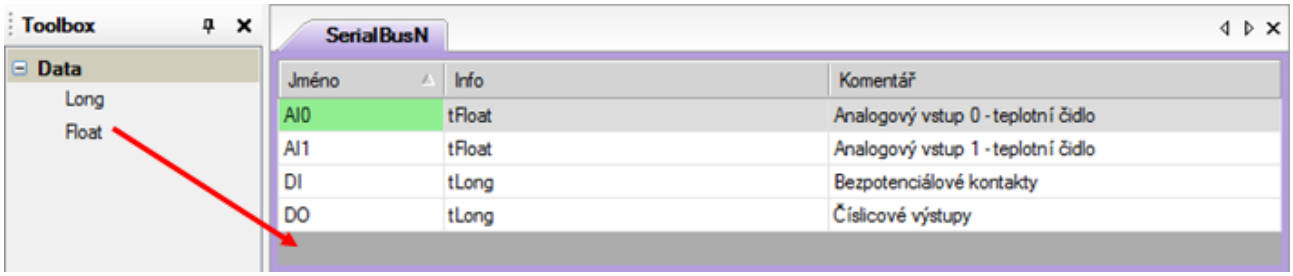


Obr. 2 – Vybraný objekt **SerialBusN**



Obr. 3 – Otevřená záložka objektu **SerialBusN**

Proměnné se mapují přetažením požadovaného typu proměnné pomocí myši z okna „Toolbox“ do otevřené záložky objektu **SerialBusN**, viz Obr. 4. Všechny registry jsou vstupně výstupní.



Obr. 4 Mapování proměnných do komunikačních registrů

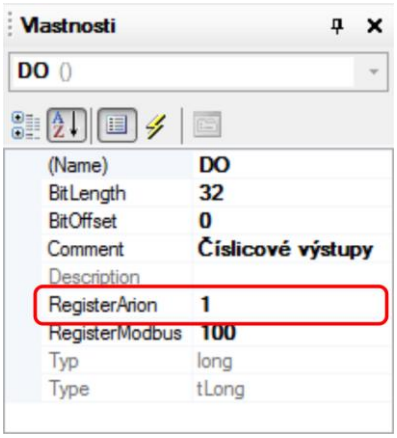
Význam jednotlivých sloupců záložky **`SerialBusN`** je následující:

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***“.

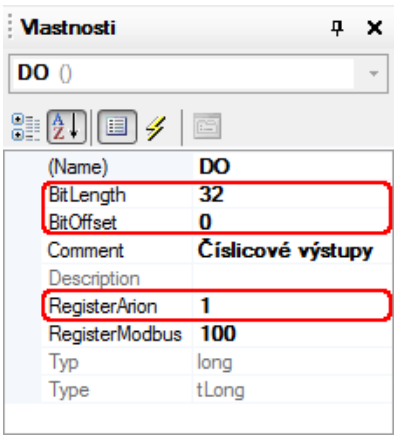
Nadefinovaným proměnným je nutné přiřadit číslo registru, do kterého bude proměnná mapována. Číslo registru se zadá prostřednictvím položky **`RegisterArion`** v panelu „Vlastnosti“ daného registru viz Obr. 5.



Obr. 5 – Okno „Vlastnosti“ definovaného registru

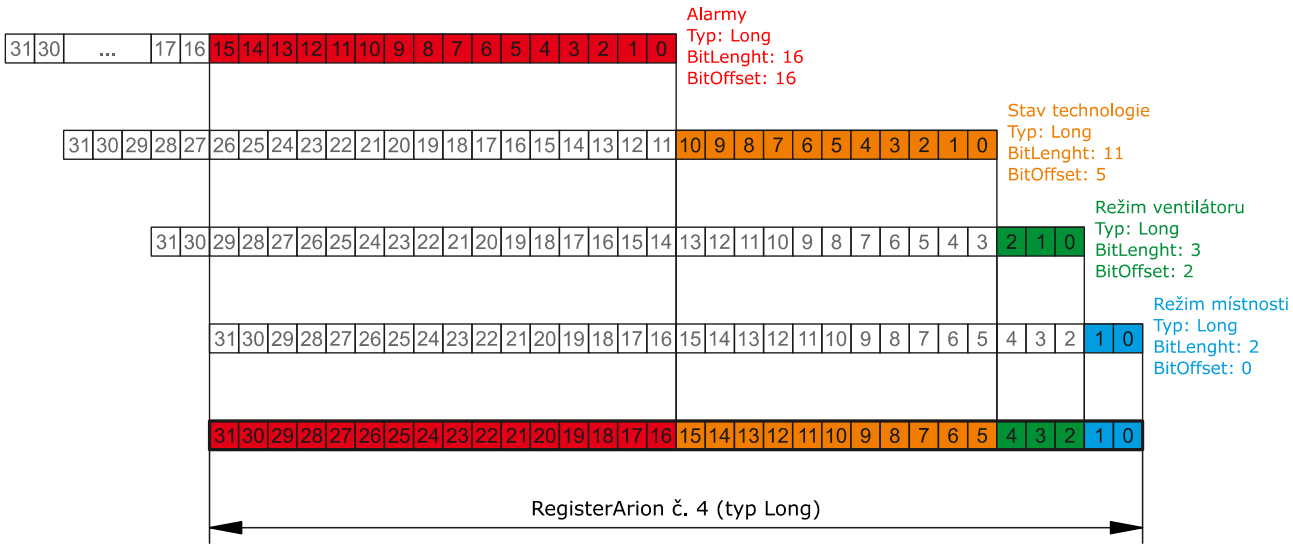
Do registru lze namapovat:
- Jednu proměnnou typu Float.
- Jednu proměnnou typu Long.
- Více nezávislých skupin bitů (až 32) z různých proměnných typu LONG.

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



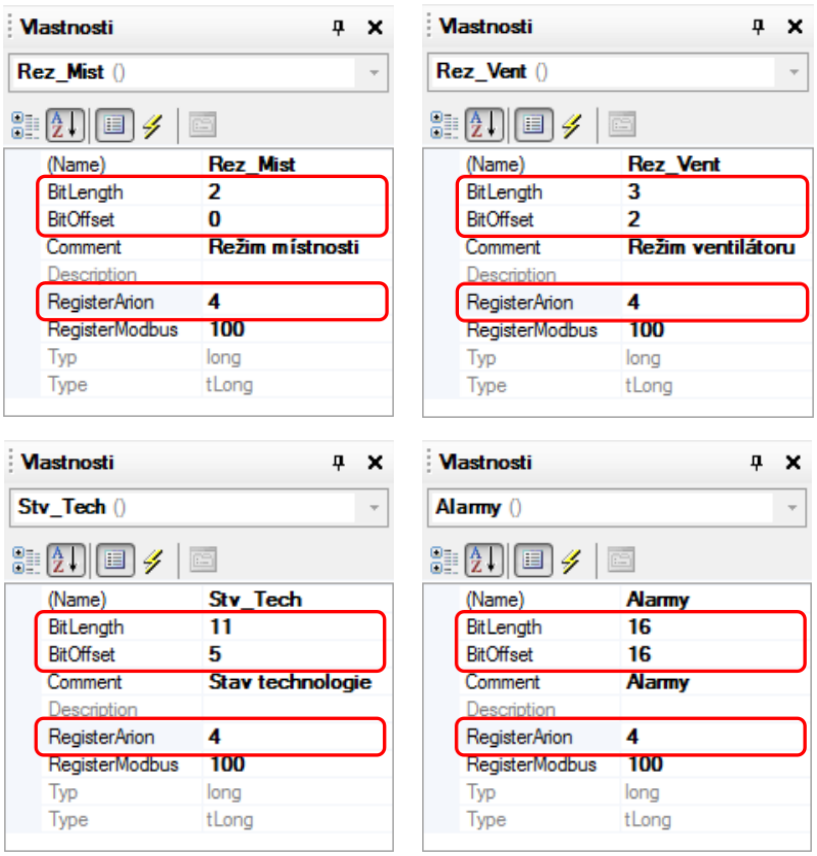
Obr. 6 – Rozdělení registru na více nezávislých skupin

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 jednom registru. Níže uvedený obrázek ukazuje mapování 4 různých proměnných (každá využívá různý počet bitů) do jednoho registru (číslo 4) typu Long.



Obr. 7 – Ukázka využití položek BitOffset a BitLenght

Mapování různých proměnných do jednoho registru (viz obrázek výše) v panelu vlastností ukazuje následující obrázek. Každé proměnné odpovídá jeden panel vlastností.



Obr. 8 – 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.

3.2 Ukázková aplikace pro AMREG – komunikace registry ARION

Součástí této aplikační poznámky je ukázková aplikace pro regulátor **AMR-FCT10/01**. Jedná se o soubor **slave1_amr_p1_cz_xx.dso**. V aplikaci jsou výše uvedeným postupem nadefinovány proměnné dle Obr. 9., které jsou navázány na komunikační registry 0, 1, 2, 3 a 4.

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. 9 Nadefinované proměnné

3.2.1 Význam / použití mapovaných proměnných

Proměnná DI (registr č. 0)
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 (registr č. 1)
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 (registr č. 2), AI1 (registr č. 3)
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 (registr č. 4)
Aplikačně nejsou nijak využívány. Slouží pouze jako ukázka mapování více proměnných do jednoho registru.

3.3 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**.

3.4 Ukázková aplikace pro řídicí systém firmy AMiT jako master – komunikace registry

Ukázková aplikace (soubor **master1_rs_p2_cz_xx.dso**) pro řídicí systém, který po síti ARION vyčítá / zapisuje z / do komunikačních registrů nadefinovaných ve slave zařízení. Ukázková aplikace 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 Definice sítě ARION a uzlu (regulátoru) s komunikačními registry

Obecně lze v řídicím systému definovat síť ARION:

- Pomocí tabulky (viz aplikační poznámka AP0025).
- Pomocí SW modulu ARION v procesu INIT (viz nápověda k části PseDet prostředí DetStudio).
- Kombinací výše uvedených možností.

Jelikož v tabulce pro síť ARION není k dispozici volně konfigurovatelný modul, který by umožňoval práci pouze se samostatnými registry, je nutné síť ARION nadefinovat v procesu INIT.

```
// Definice komunikační sítě ARION
:1000     ARION 1, 38400, 3
```

Uvedený kód definuje síť ARION na komunikačním portu č. 1 (u většiny řídicích systémů se jedná o rozhraní RS485), s komunikační rychlostí 38400 bps.

Pro komunikaci registrů lze na straně řídicího systému využít:

- Komunikaci pomocí kanálu DI, který signalizuje zápis do registru ze strany regulátoru (viz nápověda k části PseDet prostředí DetStudio).
- Periodická komunikace registrů.

Komunikace registrů pomocí kanálu DI zajišťuje optimální přenos dat. Řídicí systém se periodicky dotazuje na stav kanálu DI (po lince RS485 se tedy přenáší minimum dat), který je automaticky vytvořen na straně regulátoru po vložení objektu **SerialBusN** (viz nápověda k části EsiDet prostředí DetStudio). Kanál DI svým bitem č. 1 informuje o zápisu do některého z registrů ze strany regulátoru. Teprve poté, co řídicí systém (periodickým čtením kanálu DI) zjistí, že byl proveden zápis do registru, provede načtení požadovaných registrů. Zápis do registrů regulátoru ze strany řídicího systému lze provádět např. událostně (při změně).

Periodická komunikace registrů (bez využití kanálu DI), která je řešena v rámci této aplikační poznámky, je programátorsky přívětivější, nicméně daleko více zatěžuje komunikační linku RS485. Spočívá v nadefinování uzlu s určitým předem definovaným počtem vstupně výstupních registrů s pevně definovanou periodou komunikace. Definice uzlu se provádí taktéž v procesu INIT.

```
// Definice komunikačních registrů na uzlu s adresou 1
ARI_RgNode :1000, 1, 1000, 2000, 5, 5
```

Uvedený kód definuje uzel s adresou 1, který obsahuje 5 vstupně výstupních registrů. Registry budou automaticky komunikovány s periodou 1000 ms.

Více informací o definici sítě ARION a uzlu s komunikačními registry lze nalézt v nápovědě PseDet u modulů **ARION** a **ARI_RgNode**.

3.4.2 Čtení / zápis registrů v řídicím systému AMiT

Význam proměnných založených v ukázkové aplikaci pro řídicí systém firmy AMiT je uveden v následující tabulce.

Proměnná	Typ	Komentář
AI	MF	Matice, do které se načítají analogové hodnoty univerzálních vstupů.
DI	L	Proměnná, do které se načítají stavy bezpotenciálových kontaktů.
DO	L	Proměnná, jejíž hodnota se zapisuje na digitální výstupy.
Stavy	L	Proměnná, do které se načítá registr, složený z více hodnot.
Stavy_Poz	L	Proměnná, jejíž bity 0 až 5 se zapisují do registru, složeného z více hodnot.

Pro vyčtení obsahu registrů do lokálních proměnných lze použít SW modul **Ari_RegIn**. Pro zápis obsahu proměnných do registrů lze použít SW modul **Ari_RegOut**. Oba moduly se vkládají do některého z periodických procesů. Příklad kódu, který čte / zapisuje do registrů, může vypadat následovně.

```
// Vyčtení stavu univerzálních vstupů jako bezpotenciálový kontakt
ARI_RegIn 1, 0, 1, DI, NONE[0,0], 3

// Vyčtení analogových hodnot
ARI_RegIn 1, 2, 2, AI[0,0], NONE[0,0], 5

// Vyčtení registru s více hodnotami
ARI_RegIn 1, 4, 1, Stavy, NONE[0,0], 3

// Zápis stavu na digitální výstupy
ARI_RegOut 1, 1, 1, DO, NONE[0,0], 3
```

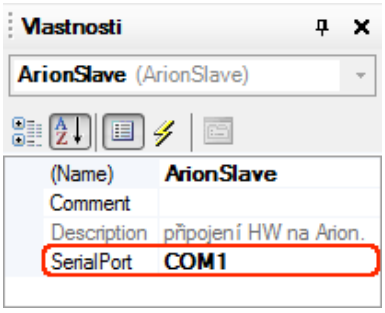
Více informací o čtení / zápisu z / do registrů lze nalézt v nápovědě PseDet u modulu **ARI_RegIn** a **ARI_RegOut**.

4 Komunikace pomocí kanálů AI, DI, AO a DO

4.1 SW parametrizace

4.1.1 Komunikační objekt

Pro komunikaci v síti ARION prostřednictvím vstupně výstupních kanálů je nutné vložit do projektu regulátoru AMREG objekt **`ArionSlave`**. V okně „Vlastnosti“ objektu **`ArionSlave`** se nastaví položka označená na Obr. 10.



Obr. 10 Okno „Vlastnosti“ objektu **`ArionSlave`**

SerialPort
Výběr COM portu regulátoru pro komunikaci protokolem ARION.

Aby fungovala komunikace v síti ARION správně, musí mít každá stanice v síti nastavenu stejnou komunikační rychlost a každý slave musí mít nastavenu jedinečnou adresu. Regulátorům AMREG lze komunikační rychlost a číslo stanice v síti ARION 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.

4.1.2 Definice komunikačních kanálů

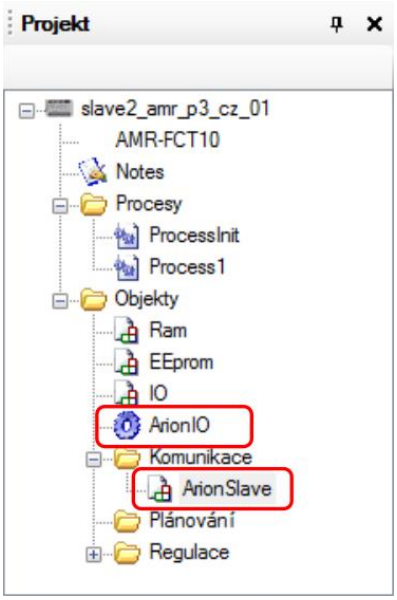
Objekt **`ArionSlave`** automaticky vytvoří virtuální kanály AI, DI, AO a DO, ke kterým lze následně z řídicího systému přistupovat např. pomocí SW modulů **`Ari_AnIn`**, **`Ari_DigIn`**, **`Ari_AnOut`**, **`Ari_DigOut`**. Tímto způsobem lze jednoduše vytvořit z regulátoru AMREG např. vstupně výstupní rozšiřující modul, obdobný rozšiřujícím modulům **DM-xxx**, který do sítě ARION poskytuje nejen hodnoty a stavy vstupů či výstupů, ale také libovolné další hodnoty, důležité pro předání potřebných informací.

Objekt **`ArionSlave`** automaticky vytvoří následující počet kanálů, které jsou k dispozici jako jeho vlastnosti.

- 24 × AI,
- 24 × DI,
- 24 × AO,
- 24 × DO.

Současně s vložením objektu **ArionSlave** se u regulátorů disponujícím alespoň jedním COM portem automaticky vytvoří v projektu objekt **ArionIO** (viz Obr. 11), který poskytuje hodnoty, měřené na analogových vstupech regulátoru tak, aby je bylo možno číst z řídicího systému přímo pomocí SW modulů **Ari_AnIn** (viz aplikační poznámka AP0025). Obdobně pracuje také s hodnotami, které řídicí systém požaduje zapsat, pomocí SW modulů **Ari_AnOut** (viz aplikační poznámka AP0025), na analogové výstupy regulátoru AMREG.

Pro práci s digitálními vstupy / výstupy se objekt **ArionIO** nevyužívá.



Obr. 11 Objekt **ArionIO** a **ArionSlave** v okně „Projekt“

4.1.3 Příklad načtení měřené teploty do kanálu AI

Načtení měřené teploty z univerzálního vstupu regulátoru AMREG např. do kanálu AI0 se provede pomocí vlastností **Ni1000VoltageX** objektu **ArionIO**, která obsahuje přepočtenou hodnotu do takového rozsahu, aby ji bylo možné zpracovat v řídicím systému modulem **Ari_AnIn** v kombinaci s modulem **Ni1000U2T**. Zápis kódu v periodickém procesu bude následovný.

```
ArionSlave.AI0 = ArionIO.Ni1000Voltage0;
```

4.1.4 Příklad načtení hodnoty analogového vstupu do kanálu AI

Načtení hodnoty analogového vstupu (např. 0 V až 10 V) regulátoru AMREG do kanálu AI0 se provede pomocí vlastností **RawAIx** objektu **ArionIO**, která obsahuje přepočtenou hodnotu do takového rozsahu, aby ji bylo možné zpracovat v řídicím systému modulem **Ari_AnIn**. Zápis kódu v periodickém procesu bude následovný.

```
ArionSlave.AI0 = ArionIO.RawAI0;
```

4.1.5 Příklad zápisu hodnoty na analogový výstup z kanálu AO

Zápis hodnoty z kanálu AO0 na analogový výstup (např. 0 V až 10 V) regulátoru AMREG se provede pomocí vlastností **RawAOx** objektu **ArionIO**, do které se zapisuje hodnota v takovém rozsahu, aby bylo možné ji ovlivňovat z řídicího systému modulem **Ari_AnOut**. Zápis kódu v periodickém procesu bude následovný.

```
ArionIO.RawAO0 = ArionSlave.AO0;
```

4.1.6 Příklad načtení stavu digitálního vstupu do kanálu DI

Načtení stavu z univerzálního vstupu regulátoru AMREG, vyhodnoceného jako bezpotenciálový kontakt, do kanálu DI se provede pomocí vlastnosti **DIx** objektu **IO**, který obsahuje hodnotu odpovídajícího vstupu. Zápis kódu v periodickém procesu bude následovný.

```
ArionSlave.DI0 = IO.DI0;
```

4.1.7 Příklad zápisu hodnoty na digitální výstup z kanálu DO

Zápis stavu z kanálu DO na digitální výstup (např. reléový) regulátoru AMREG se provede např. pomocí vlastnosti **RLx** objektu **IO**, do kterého se zapisuje požadovaný stav digitálního výstupu. Zápis kódu v periodickém procesu bude následovný.

```
IO.RL0 = ArionSlave.DO0;
```

Poznámka
*Název vlastností objektu IO, určených pro zápis na digitální výstupy regulátoru AMREG se liší v závislosti na typu digitálních výstupů, kterými je regulátor osazen (**DOx**, **TRIx**, apod.).*

4.2 Ukázková aplikace pro AMREG – komunikace vstupně výstupními kanály

Součástí této aplikační poznámky je ukázková aplikace pro regulátor **AMR-FCT10**. Jedná se o soubor **slave2_amr_p3_cz_xx.dso**. V ukázkové aplikaci je pro daný regulátor řešeno:

- Načtení stavu univerzálních vstupů vyhodnocených jako bezpotenciálový kontakt do kanálu DI.
- Načtení měřených teplot do kanálu AI.
- Zápis stavů kanálu DO na výstupy regulátoru.

Výše uvedené lze zapsat do periodického procesu následovně.

```
// Stav univerzálních vstupů vyhodnocených jako bezpotenciálový kontakt
ArionSlave.DI0 = IO.DI0;
ArionSlave.DI1 = IO.DI1;

// Hodnoty měřených teplot
ArionSlave.AI0 = ArionIO.Ni1000Voltage0;
ArionSlave.AI1 = ArionIO.Ni1000Voltage1;

// Zápis na relé
IO.RL0 = ArionSlave.DO0;
IO.RL1 = ArionSlave.DO1;
IO.RL2 = ArionSlave.DO2;

// Zápis na triakové výstupy
IO.TRI0 = ArionSlave.DO3;
IO.TRI1 = ArionSlave.DO4;
```

4.3 Stav komunikace

Pro vyhodnocení stavu komunikace lze využít vlastnost **`Connected`** objektu **`ArionSlave`**, kterou lze vyhodnocovat v některém z periodických procesů, typicky rozpad komunikace s nadřazeným systémem.

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

4.4 Ukázková aplikace pro řídicí systém firmy AMiT jako master – komunikace vstupně výstupními kanály

Ukázková aplikace (soubor **master2_amr_p4_cz_xx.dso**) pro řídicí systém, který po síti ARION vyčítá / zapisuje z / do kanálů nadefinovaných ve slave zařízení. Ukázková aplikace 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…“.

4.4.1 Definice sítě ARION a uzlu s kanály AI, DI, AO, DO

Obecně lze v řídicím systému definovat síť ARION:

- Pomocí tabulky (viz aplikační poznámka AP0025).
- Pomocí SW modulu ARION v procesu INIT (viz nápověda k části PseDet prostředí DetStudio).
- Kombinací výše uvedených možností.

Jelikož v tabulce pro síť ARION není k dispozici volně konfigurovatelný modul, který by umožňoval nejen nastavení počtu vstupů a výstupů, ale také způsob zpracování vstupních analogových hodnot (bipolární / unipolární), je nutné síť ARION nadefinovat v procesu INIT.

```
// Definice komunikační sítě ARION
:1000     ARION 1, 38400, 3
```

Uvedený kód definuje sít ARION na komunikačním portu č. 1 (u většiny řídicích systémů se jedná o rozhraní RS485), s komunikační rychlostí 38400 bps.

Definice uzlu se provádí taktéž v procesu INIT. Pro každý kanál (AI, AO, DI, DO), který má řídicí systém zpracovávat je nutné použít jeden modul, který bude mít nastavenou stejnou adresu uzlu.

```
// Definice kanálů AI
ARI_AINode :1000, 1, 1000, 2500, 2, 0x000C
// Definice kanálů DI
ARI_DINode :1000, 1, 500, 2000, 2
// Definice kanálů DO
ARI_DONode :1000, 1, 500, 2000, 5
```

Uvedený kód definuje uzel s adresou 1, který obsahuje:

- 2 analogové vstupy,
- 2 digitální vstupy,
- 5 digitálních výstupů.

Analogové vstupy budou automaticky čteny s periodou 1000 ms, digitální vstupy a výstupy se budou automaticky komunikovat s periodou 500 ms.

Více informací o definici sítě ARION a uzlu s komunikačními registry lze nalézt v nápovědě PseDet u modulů **`ARION`** a **`ARI_xxNode`**.

4.4.2 Čtení / zápis kanálů řídicím systém AMiT

Význam proměnných založených v ukázkové aplikaci pro řídicí systém firmy AMiT je uveden v následující tabulce.

Proměnná	Typ	Komentář
AI	MF	Matice, do které se načítají analogové hodnoty kanálu AI.
DI	L	Proměnná, do které se načítají jednotlivé stavy kanálu DI.
DO	L	Proměnná, jejíž hodnota se zapisuje na kanál DO.

Pro vyčtení obsahu kanálů AI a DI do lokálních proměnných lze použít např. SW moduly **Ari_AnIn** a **Ari_DigIn**. Pro zápis obsahu lokálních proměnných do kanálů AO a DO lze použít např. SW moduly **Ari_AnOut** a **Ari_DigOut**. Moduly se vkládají do některého z periodických procesů. Příklad kódu, který čte / zapisuje do kanálů, může vypadat následovně.

```
// Čtení kanálu DI - stavy bezpotenciálových kontaktů
ARI_DigIn 1, 0, DI, 0x00000000

// Čtení kanálu AI - měřené teploty v rozsahu 0 V až 5 V
// Hodnoty ukládány do buněk 0. sloupce matice AI
ARI_AnIn 1, 0, 2, AI[0,0], NONE[0,0], 5.000, 0.000, 5.000, 0.000, 5.000
// Přepočet změřeného napětí na teplotu
// Přepočtené hodnoty ukládány do buněk 1. sloupce matice AI
Ni1000U2T AI[0,0], AI[0,1], 6180, 15.000, 3920.000
Ni1000U2T AI[1,0], AI[1,1], 6180, 15.000, 3920.000

// Zápis na kanál DO
ARI_DigOut 1, 0, 5, DO, 0x00000000
```

Více informací o čtení / zápisu vstupně výstupních kanálů lze nalézt v nápovědě PseDet u modulů **ARI_AnIn**, **ARI_DigIn**, **ARI_AnOut**, **ARI_DigOut** a v aplikační poznámce AP0025.

5 Technická podpora

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

6 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.