Komunikace v síti MODBUS TCP (Gen1)

Abstrakt

Aplikační poznámka popisuje použití MODBUS TCP protokolu v řídicích systémech Generace 1 pomocí definice tabulkou. A to jak pro komunikaci s jinými výrobky pracujících jako Slave stanice, tak i pro komunikaci s nadřazenou Master stanicí.

Autor: Michal Kupčík
Dokument: ap0059_ap_cz_002.pdf

Příloha

Obsah souboru: ap0059_ap_cz_002.zip

modbstcp_p1_cz_100.dso	Příklad parametrizace řídicího systému jako master.
modbstcp_p2_cz_101.dso	Příklad master komunikace s šesti slave stanicemi bez ošetření nenavázání TCP spojení.
modbstcp_p3_cz_100.dso	Příklad parametrizace řídicího systému jako slave.
modbstcp_p4_cz_100.dso	Příklad master komunikace s šesti slave stanicemi včetně ošetření nenavázání TCP spojení.

Obsah

Obsah ... 2
Historie revizí ... 3
Související dokumentace ... 3
Určení aplikační poznámky ... 3
1 Definice použitých pojmů ... 4
2 Protokol MODBUS ... 5
2.1 Podporované funkce MODBUS ... 5
3 Zapojení komunikační sítě ... 6
4 Časové poměry na síti ... 8
4.1 Doba komunikace ... 8
Příklad ... 8
4.1.1 Priority komunikace ... 9
Automatické priority čtení ... 9
Automatická priorita zápisu ... 9
Manuální priorita komunikace ... 9
4.1.2 Sdružování komunikačních rámců ... 10
Příklad ... 10
4.2 Komunikace při rozpadu spojení ... 10
5 Řídicí systém jako Master ... 11
5.1 Komunikační definice ... 11
5.2 Definice Slave stanice a datových bodů pro komunikaci ... 12
5.3 Automatická komunikace ... 13
5.4 Manuální komunikace ... 14
5.5 Stavy komunikace ... 15
5.6 Příklad parametrizace řídicího systému jako Master ... 16
5.7 Master komunikace s více než 3 Slave stanicemi ... 17
5.8 Master komunikace s více než 3 Slave stanicemi včetně ošetření nenavázání TCP spojení ... 20
6 Řídicí systém jako Slave ... 22
6.1 Komunikační definice ... 22
6.2 Definice datových bodů pro komunikaci ... 23
6.3 Stavy komunikace ... 23
6.4 Příklad parametrizace řídicího systému jako Slave ... 23
7 Dodatek A ... 26
7.1 Kompatibilita s inicializací komunikace pomocí modulů ... 26
7.1.1 MODBUS Master ... 26
7.1.2 MODBUS Slave ... 26
8 Technická podpora ... 27
9 Upozornění ... 28

Historie revizí

Verze	Datum	Autor změny	Změny
001	25. 03. 2019	Kupčík M.	Nový dokument.
002	22. 03. 2022	Kupčík M.	Úpravy textů, nová kapitola 5.8.

Související dokumentace

1. Nápověda k části PseDet vývojového prostředí DetStudio
   soubor: PseDet_cs.chm
2. Aplikační poznámka AP0037 – Zásady používání sítě Ethernet
   soubor: ap0037_ap_cz_xxx.pdf

Ostatní dokumentace dostupná v době vydání tohoto dokumentu:

3. RFC 793 Transmission Control Protocol - Protocol specification
   soubor: https://www.ietf.org/rfc/rfc793.txt

Určení aplikační poznámky

Aplikační poznámka je určena řídicím systémům Generace 1 – řídicím systémům s operačním systémem NOS programovaných v DetStudiu v editoru PseDet. Více informací o generacích systémů lze nalézt na webu amitomation.cz.

1 Definice použitých pojmů

Řídicí systém
Jedná se o řídicí systémy a terminály firmy Generace 1 firmy AMiT, u kterých se algoritmy procesů programují v tzv. PseDet součásti prostředí DetStudio. Např. **AMiNi4DW2**, **AD-CPUW2**, **AMAP99W3** nebo **ART4000W3**.

Master stanice
Tato stanice aktivně komunikuje se Slave stanicemi. V TCP/IP terminologii se označuje jako Client.

Slave stanice
Jedná se o stanici s unikátní adresou, která pasivně naslouchá na komunikačním rozhraní a po příjmu korektního rámce od Mastera na ni odpoví. V TCP/IP terminologii se označuje jako Server.

Datový bod
Jedná se o definici registru (vstupního či uchovávacího) nebo bináru (vstupního či výstupního), který typicky reprezentuje vstup nebo výstup na Slave stanici. Každému datovému bodu se přímo přiřazuje (maticová) proměnná či bit, do kterého se budou zapisovat načtené hodnoty, nebo ze kterých se budou brát hodnoty pro zápis do Slave stanice.

2 Protokol MODBUS

MODBUS je otevřený komunikační protokol vyvinutý firmou Modicon. Původně byl protokol navržen na sběrnici RS232, brzy se ale přešlo na RS485 z důvodu její vyšší spolehlivosti a možnosti propojení více zařízení na větší vzdálenosti. Postupem vývoje byl MODBUS rozšířen i o popis implementace na rozhraní Ethernet. Jako standardní TCP port byl zvolen 502. Protokol je flexibilní, ale zároveň jednoduchý na implementaci, a tak jej brzy začali různí výrobci implementovat do svých zařízení. V současné době umožňují komunikaci MODBUS protokolem nejen mikrokontroléry nebo průmyslová PC, ale také množství inteligentních snímačů, akčních členů a dalších jednoduchých prvků.

Firma AMiT podporuje komunikaci protokolem MODBUS TCP v řídicích systémech s písmenem „W“ v názvu. Určení master/slave závisí na konkrétní implementaci.

Poznámka
Při definici komunikačního protokolu MODBUS TCP lze na Ethernet rozhraní využívat i další komunikační protokoly.

Pozor!
Adresace datových bodů může být různými výrobci chápáno různými způsoby navzdory specifikaci protokolu MODBUS. Více o tomto se dočtete v nápovědě DetStudia „PseDet – tvorba regulačních procesů“ v kapitole „Obsah/Komunikace/Modbus“ v části „Adresování registrů/binárů“.

Pozor!
Pro podporu komunikace protokolem MODBUS TCP musí být v řídicím systému nahrán operační systém NOS minimálně verze 3.70.

2.1 Podporované funkce MODBUS

V řídicích systémech firmy AMiT jsou podporovány následující funkce protokolu MODBUS. Funkce vychází z definice protokolu MODBUS a určují typ použitého rámce.

Funkce č.	Popis
1	Čtení jednoho/více binárních výstupů.
2	Čtení jednoho/více binárních vstupů.
3	Čtení jednoho/více uchovávacích registrů.
4	Čtení jednoho/více vstupních registrů.
5	Zápis jednoho binárního výstupu.
6	Zápis jednoho uchovávacího registru.
15	Zápis více binárních výstupů.
16	Zápis více uchovávacích registrů.

Uvedený popis je pouze obecný a orientační. Vlastní význam jednotlivých funkcí závisí na konkrétním typu zařízení.

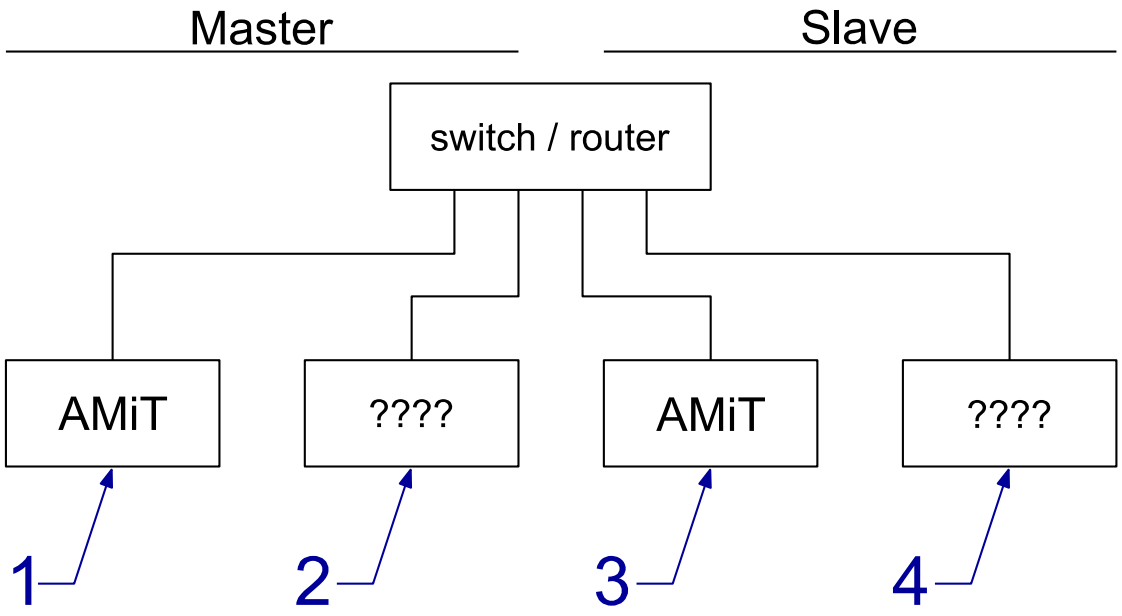
Velice často se pro analogové hodnoty používají dvojice registrů, takže zápis analogových výstupů probíhá pomocí funkce č. 16. Více informací naleznete v nápovědě DetStudia „PseDet – tvorba regulačních procesů“ v kapitole „Obsah/Komunikace/Modbus“ v části „Mapování komunikačních bodů“.

3 Zapojení komunikační sítě

Pro správné plnění požadované funkce celé sítě MODBUS je nutno správně navrhnout, zapojit, nakonfigurovat jednotlivé moduly sítě a naprogramovat komunikaci v řídicích systémech.

Při zapojování sítě na rozhraní Ethernet je nutno dodržet doporučení uvedená v dokumentu AP0037 – Zásady používání sítě Ethernet.

Řídicí systém AMiT může v sítí MODBUS vystupovat jako Master nebo jako Slave. V roli Master typicky v kombinaci s technologickými zařízeními jiných výrobců (např. akční členy) nebo v roli Slave jako součást rozsáhlejších sítí.

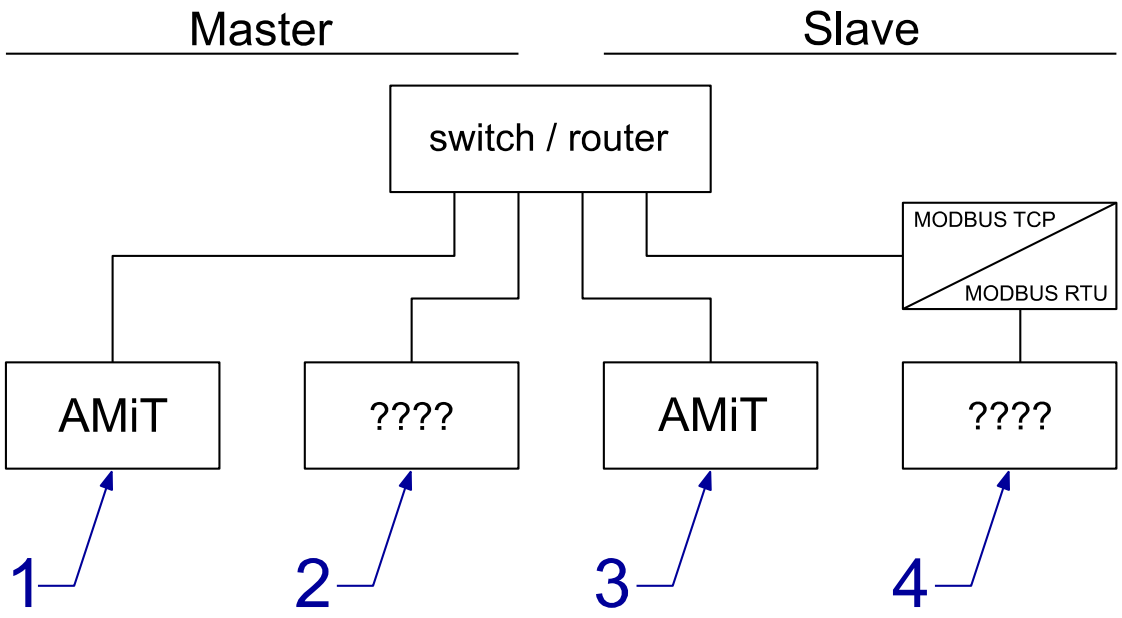


Obr. 1 – Komunikace pomocí protokolu MODBUS TCP přes síťový prvek

Legenda

Číslo	Význam
1	Řídicí systém AMiT jako Master stanice
2	Zařízení jiných výrobců jako Master stanice
3	Řídicí systém AMiT jako Slave stanice
4	Zařízení jiných výrobců jako Slave stanice

Jako Slave stanice může být použit převodník MODBUS TCP / MODBUS RTU.



Obr. 2 – Komunikace pomocí protokolu MODBUS TCP s převodníkem TCP na RTU

Legenda

Číslo	Význam
1	Řídicí systém AMiT jako Master stanice
2	Zařízení jiných výrobců jako Master stanice
3	Řídicí systém AMiT jako Slave stanice
4	MODBUS RTU zařízení jako Slave stanice

4 Časové poměry na síti

Komunikace probíhá s určitou periodou, na jejímž začátku master aktivuje rozhraní (na základě definiční tabulky se vytvoří balík požadavků) a následně se vykomunikují požadavky pro každý vzdálený bod uvedený v definiční tabulce. Aktivita rozhraní končí vykomunikováním posledního rámce.

4.1 Doba komunikace

Doba komunikace po rozhraní Ethernet je obecně silně závislá na ostatním provozu na Ethernet síti. Při větším vytížení dokonce může dojít i ke ztrátě paketů. Protokol TCP má v sobě implementovány mechanismy, které by měly tyto případné ztráty detekovat a provést znovuvyslání ztraceného paketu.

Zatímco v RTU specifikaci protokolu MODBUS jsou jasně dány časové poměry, v případě TCP komunikace tyto jasně dané poměry nejsou. Navíc záleží i na způsobu implementace tzv. TCP stacku na straně Mastera i Slave stanice. Z těchto důvodů nelze obecně určit doby komunikace.

Následující tabulka obsahuje doporučené časy pro výpočty dob komunikací. Tyto však platí pro komunikaci mezi řídicími systémy firmy AMiT a v případě nevelkého vytížení sítě.

Minimální doba komunikace [ms]	Doba komunikace datového bodu [ms]
5	0,2× registr, 0,1× každá započatá osmice binárů

Příklad

Potřebujeme periodicky komunikovat se třemi Slave stanicemi. U každého se vyžaduje komunikace 8 registrů a 18 binárů. Pro každou stanici tedy budou definovány dva řádky komunikace vzdálených bodů (jedna skupina registrů či binárů). Pro výpočet budeme vycházet z tabulkových hodnot. To znamená předpoklad podobné implementace TCP stacku a nevelkého vytížení sítě.

$T_{reg} = 5 + 0,2 \times 8 = 6,6$ ms

$T_{bin} = 5 + 0,1 \times 3 = 5,3$ ms

V případě výpočtu T_{bin} se vycházelo ze skutečnosti, že 18 binárů se rozdělí na 8 + 8 + 2, tedy jako hodnota každé započaté osmice se bere 3.

Celková minimální doba komunikace tedy je:

$T_{celk} = 6,6 \times 3 + 5,3 \times 3 = 35,7$ ms

Doplnění
Zcela identická doba by vyšla, kdyby se komunikovalo pouze s jednou Slave stanicí a u této Slave stanice by se dané rozvržení 3 × 8 registrů a 3 × 18 binárů komunikovalo na šesti řádcích definiční tabulky.

Upozornění!
Výše provedené výpočty je doporučeno aplikovat pouze při komunikaci s max. třemi Slave stanicemi, resp. při komunikaci po max. trojicích Slave stanic. Více o tomto je popsáno v kapitolách 5.7 „Master komunikace s více než 3 Slave stanicemi“ a 5.8 „Master komunikace s více než 3 Slave stanicemi včetně ošetření nenavázání TCP spojení“.

4.1.1 Priority komunikace

Automatické priority čtení

DetStudio nabízí pro automatické čtení hodnot ze Slave stanice tři priority čtení:

Priorita čtení	Perioda komunikace [ms]
Low	5000
Normal	1000
High	200

Volbou priority programátor volí, s jakou periodou se má daný řádek tabulky vyčítat.

Platí, že každých 200 ms operační systém NOS prochází jednotlivé definiční tabulky a v případě, že nalezne řádek s automatickou prioritou čtení a v daném 200 ms cyklu se má tento řádek komunikovat, je daný řádek zařazen do fronty komunikačních požadavků.

Automatická priorita zápisu

V případě automatického zápisu není definována priorita s časovou periodou, k dispozici je pouze přepnutí na prioritu **`Auto`**.

V případě vybrání této priority se po každém zápisu do přiřazené proměnné (byť i stejné hodnoty) provede její označení a je zařazena na začátek fronty komunikačních požadavků. Zápisy se tedy vždy komunikují před čtecími požadavky.

Vlivem vnitřních mechanizmů detekce zápisu do přiřazené proměnné může být daná proměnná použita v definiční tabulce s prioritou **`Auto`** pouze jednou. Je-li např. vyžadováno, aby hodnoty z více buněk maticové proměnné sloužily pro zápis různých registrů či hodnoty bitů celočíselné proměnné sloužily pro zápis různých binárů, lze definici řešit na základě rozložení registrů:

- Jsou-li zápisové registry či binárý v postupné řadě za sebou, nadefinovat pouze jediný definiční řádek a v něm mít nastavenou hodnotu sloupce **`Počet`** na odpovídající hodnotu. Příklad takovéto definice lze nalézt v nápovědě DetStudia „PseDet – tvorba regulačních procesů“ v kapitole „Obsah/Komunikace/Modbus/Master – editor tabulky Device“ v části „Poznámky“.
- Nejsou-li zápisové registry či binárý v postupné řadě za sebou, je nutné daným definičním řádkům nastavit manuální prioritu komunikace. Pro detekci změny hodnoty binárů lze s výhodou využít modulu **`BinDiff`**.

Manuální priorita komunikace

V případě, že automatická priorita komunikace definičních řádků nedovoluje korektní požadovaný způsob komunikace se Slave stanicí, je nutné využít manuální priority komunikace. Tato se nastaví volbou **`--manual--`** v požadovaném sloupci priority komunikace.

Pro spuštění manuální komunikace se využívá modulů:

- **`MdbmMark`** – označení většího množství definičních řádků ke komunikaci,
- **`MdbmRead`** – označení konkrétního definičního řádku ke čtení,
- **`MdbmWrite`** – označení konkrétního definičního řádku k zápisu,
- **`MdbmWrBeg`**, **`MdbmWrFin`** – označení konkrétního definičního řádku k tzv. bezpečnému zápisu.

Moduly pracují s návěštími na konkrétní definici Device a s výjimkou modulu **`MdbmMark`** i návěštími na konkrétní definiční řádek. Více informací o jednotlivých modulech se dočtete v nápovědě DetStudia „PseDet – tvorba regulačních procesů“ v popisu k jednotlivým modulům.

Na rozdíl od automatických priorit komunikace se při manuální komunikaci datové body komunikují okamžitě po vyvolání daného modulu. Použitím uvedených modulů lze tedy docílit i komunikaci s rychlejší periodou, než je 200 ms.

4.1.2 Sdružování komunikačních rámců

Při výpočtu minimální doby komunikace je potřeba počítat i s automatickým sdružováním za sebou jdoucích komunikačních rámců při využívání komunikačních funkcí 1, 2, 3, 4, 15 a 16 (viz kapitola 2.1 „Podporované funkce MODBUS“).

Příklad

Mějme definiční tabulku o dvou řádcích pro vyčítání dvou registrů do dvou proměnných. V případě, že adresy daných registrů nejsou na sebe navazující, např. adresy 0 a 2, bude komunikace probíhat dvěma rámci. Při použití tabulkových hodnot ze začátku této kapitoly vypočteme dobu:

T = 2 × (5 + 0,2 × 1) = 10,4 ms

Avšak pokud budou adresy registrů definovány přímo za sebou, např. adresy 0 a 1, bude komunikace probíhat jediným rámcem. Při použití tabulkových hodnot ze začátku kapitoly vypočteme dobu:

T = 5 + 0,2 × 2 = 5,4 ms

Vrátíme-li se k „**Doplnění**“ původního příkladu na začátku kapitoly (**Příklad**), tak v případě, že by všech 3 × 8 registrů a 3 × 18 binárů bylo definováno v nepřerušené řadě za sebou, např. u registrů adresy 0 až 7, 8 až 15 a 16 až 23, budou se registry i bináry komunikovat každý jedním rámcem. Při použití tabulkových hodnot ze začátku kapitoly vypočteme dobu:

T = (5 + 0,2 × 18) + (5 + 0,1 × 9) = 14,5 ms

4.2 Komunikace při rozpadu spojení

V případě, že není možná komunikace se Slave stanicí, resp. na dotaz nepřišla odpověď, začne se provádět algoritmus komunikace s touto stanicí následovně:

1. Rámec, na který nepřišla odpověď se ještě 2× zopakuje.
2. Pokud stále nepřišla odpověď, ignorují se následující požadavky na komunikaci na dobu 15 sekund. Ignorování požadavků je signalizováno nastavením bitu č. 4 parametru „Status“ modulu **`MdbmReqSt`** (popis viz kapitola 5.5 „Stavy komunikace“) do stavu True.
3. Po uplynutí času ignorování požadavků na komunikaci se projde tabulka komunikačních požadavků dané Slave stanice. Je-li nějaký nalezen, je proveden pokus o jeho komunikaci. Jako první se prohledávají zápisové požadavky. Současně je bit č. 4 parametru „Status“ modulu **`MdbmReqSt`** po dobu komunikace nastaven do stavu False.
4. V případě, že ani nadále nedošla korektní odpověď, jsou ke stávajícímu času ignorování požadavků na komunikaci připočteny 2 sekundy. Opětovně dojde k nastavení bitu č. 4 parametru „Status“ modulu **`MdbmReqSt`** do stavu True.
5. V případě, že se jednalo o zápisový komunikační požadavek, je požadavku zachován příznak pro komunikaci po uběhnutí času zpoždění. Pokud se jednalo o čtecí požadavek, je příznak pro komunikaci zrušen.
6. Po uběhnutí nového času zpoždění se opět algoritmus opakuje od bodu 3. Maximální doba ignorování požadavků na komunikaci je 30 s. Z tohoto tedy plyne řada časů 15 s, 17 s, 19 s, …, 29 s, 30 s, 30 s, ….

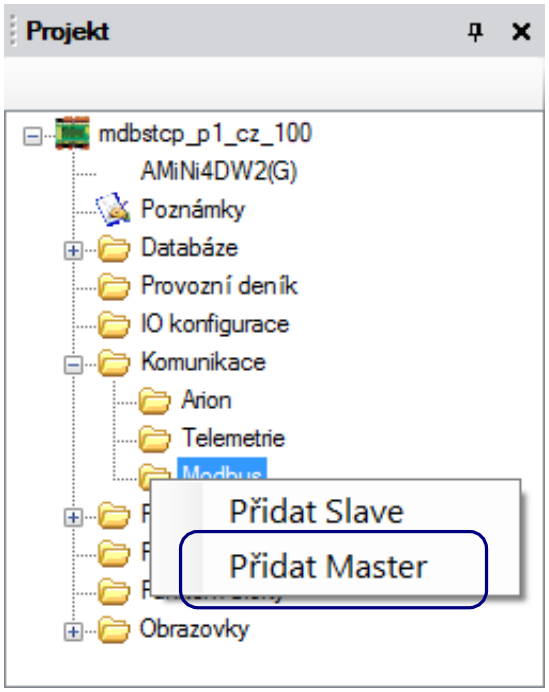
5 Řídicí systém jako Master

Definice komunikace protokolem MODBUS v roli Master probíhá pomocí trojí definice:

- vytvoření komunikační definice protokolu v roli Master,
- vytvoření definice Slave stanice,
- nadefinování datových bodů dané Slave stanice pro komunikaci.

5.1 Komunikační definice

Vytvoření komunikační definice protokolu MODBUS v roli Master reprezentuje vložení definice komunikační položky **`ModbusMaster`** do aplikace. Vložení provedeme v DetStudiu v okně Projekt v uzlu „Projekt/Komunikace/Modbus“. Po vyvolání kontextového menu nad touto položkou vybereme položku **`Přidat Master`**.



Obr. 3 – Položka „Přidat Master“ v definici komunikace „Modbus“

Po vložení Master definice bude vytvořen komunikační uzel **`ModbusMaster0`** s výchozími hodnotami vlastností:

- **`BaudRate`**: 19200
- **`Mode`**: SerialLineRTU
- **`Parity`**: Even
- **`SerialPort`**: 0
- **`StopBit`**: One
- **`ToReceive`**: 30
- **`ToTransmit`**: 4

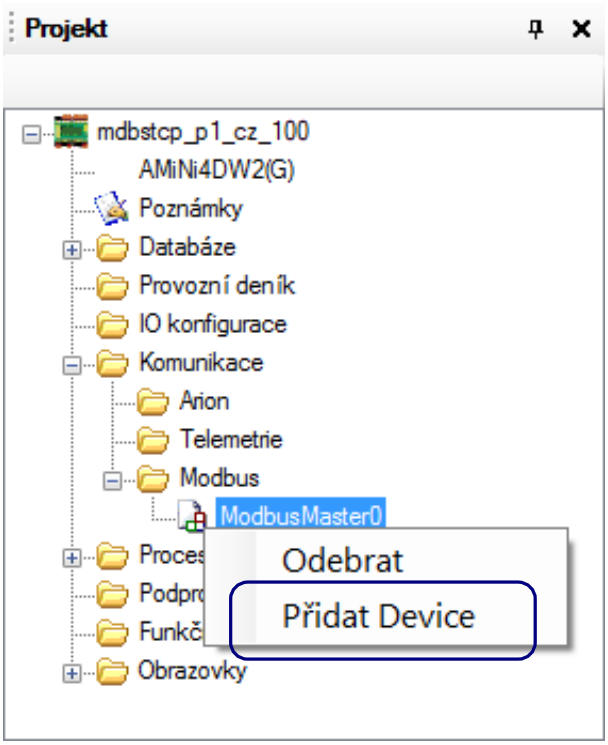
Platí, že pro komunikaci protokolem MODBUS TCP je potřeba nastavit hodnotu vlastnosti **`Mode`** na „Modbus_TCP“.

Popis ostatních vlastností je k dispozici v nápovědě DetStudia „PseDet – tvorba regulačních procesů“ v kapitole „Obsah/Komunikace/Modbus/Master – založení a nastavení obecných parametrů“.

5.2 Definice Slave stanice a datových bodů pro komunikaci

Pro komunikaci s jednotlivými Slave stanicemi musíme nadefinovat její adresu v síti MODBUS a seznam komunikačních bodů.

Jednotlivé Slave stanice, tzv. *Device*, se v okně Projekt definují přímo v uzlu konkrétní **`ModbusMasterX`** definice. Po vyvolání kontextového menu nad vybranou položkou vybereme položku **`Přidat Device`**.



Obr. 4 – Položka „Přidat Device“ v definici uzlu „ModbusMasterX“

Po vložení definice bude vytvořen komunikační uzel **`ModbusDevice0`** s výchozími hodnotami vlastností:

- **`Address`**: 1
- **`ByteOrder`**: 0-1-2-3 (Modbus default)
- **`ClientLabel`**: -1

Hodnota vlastnosti **`Address`** je typicky bez významu u definic Slave stanic, která neslouží jako převodníky MODBUS TCP na MODBUS RTU.

Vzhledem k tomu, že 32 bitové typy (Long a Float) nejsou v protokolu MODBUS nijak definovány, je pouze na volbě výrobce daného zařízení, jakým způsobem naimplementuje (pokud vůbec) tyto nadstavbové registry. Vzhledem k tomu, že pořadí bytů v 16 bitových slovech je definováno jako Big-Endian, je u produktů firmy AMiT zvolen stejný způsob kódování i na výše zmíněné 32 bitové typy.

V případě, že komunikací 32 bitových hodnot se v proměnných objevují výrazně jiné hodnoty, než jsou reálně ve Slave stanici, je doporučeno zkusit změnit hodnotu vlastnosti **`ByteOrder`**, typicky na volbu **`2-3-0-1`**.

Hodnota vlastnosti **`ClientLabel`** se využívá při manuální komunikací se Slave stanicí a při určování stavů komunikace (viz kapitoly 5.4 „Manuální komunikace“ a 5.5 „Stavy komunikace“).

Po nadefinování uzlu **ModbusDeviceX** je možné na něj v okně Projekt dvakrát kliknout a tím vyvolat definiční tabulku komunikačních datových bodů této Slave stanice.

V následujících třech kapitolách budeme uvažovat komunikaci se Slave stanicí, u které je požadavek na čtení analogových hodnot a zápis binárních hodnot. V návodu na obsluhu k tomuto zařízení nechť je uvedeno, že analogové hodnoty se čtou funkcí 4 – čtení vstupních (input) registrů a zápisy na digitální výstupy se provádějí funkcí 5 – nastavení jednoho binárního výstupu (coil) nebo 15 – nastavení binárních výstupů (coils). Z uvedených funkcí vyplývá, že v definiční tabulce uzlu **ModbusDeviceX** budou použity záložky „Input registers“ a „Coils“.

Definici jednotlivých datových bodů lze provést např. přetažením z okna ToolBox. Více informací o definici datových bodů lze nalézt v nápovědě DetStudia „PseDet – tvorba regulačních procesů“ v kapitole „Obsah/Komunikace/Modbus/Master – editor tabulky Device“.

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Návěští
0 (30001)	7 (30008)	8	MTCP_AI	--manual--	

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Priorita zápisu	Funkce zápisu	Návěští
0 (1)	7 (8)	8	MTCP_DO	--manual--	--manual--	normal Modbus	

Obr. 5 – Základní definice datových bodů a přiřazených proměnných

5.3 Automatická komunikace

Po nadefinování datových bodů jsou v definiční tabulce ve sloupcích „Priorita čtení“ a „Priorita zápisu“ předvybrány položky **--manual--**. Pro automatickou komunikaci je potřeba změnit jejich nastavení na některou z automatických priorit zmíněných v kapitole 4.1.1 „Priority komunikace“.

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Návěští
0 (30001)	7 (30008)	8	MTCP_AI	Normal	

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Priorita zápisu	Funkce zápisu	Návěští
0 (1)	7 (8)	8	MTCP_DO	--manual--	Auto	normal Modbus	

Obr. 6 – Definice datových bodů pro automatickou komunikaci

Doporučení
V případě, kdy není žádoucí, aby k zápisu docházelo, i když se do proměnné zapíše stejná hodnota, je možné doporučit následující kód, který provede zápis do komunikační proměnné až v okamžiku, kdy se hodnota pomocné pracovní proměnné změní:

```
If MTCP_DO != MTCP_DO_pr
  Let MTCP_DO_pr = MTCP_DO
EndIf
```

V aplikaci se bude tato pomocná pracovní proměnná využívat v kódu aplikace, zatímco komunikační proměnná bude využívaná pouze v definiční tabulce.

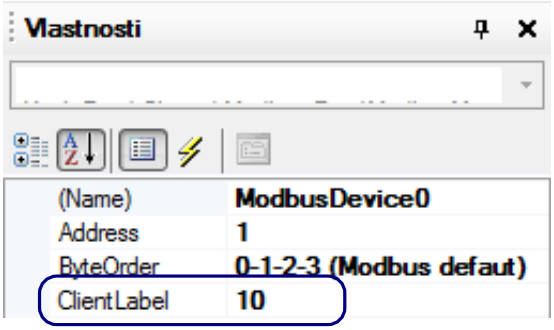
Poznámka
Vlivem vnitřních algoritmů pro použití automatické priority zápisu není vhodné mít na jednom definičním řádku automatickou prioritu čtení i zápisu. Více o tomto lze nalézt v nápovědě DetStudia „PseDet – tvorba regulačních procesů“ v kapitole „Obsah/Komunikace/Modbus/Master – editor tabulky Device“ v části „Poznámky“.

5.4 Manuální komunikace

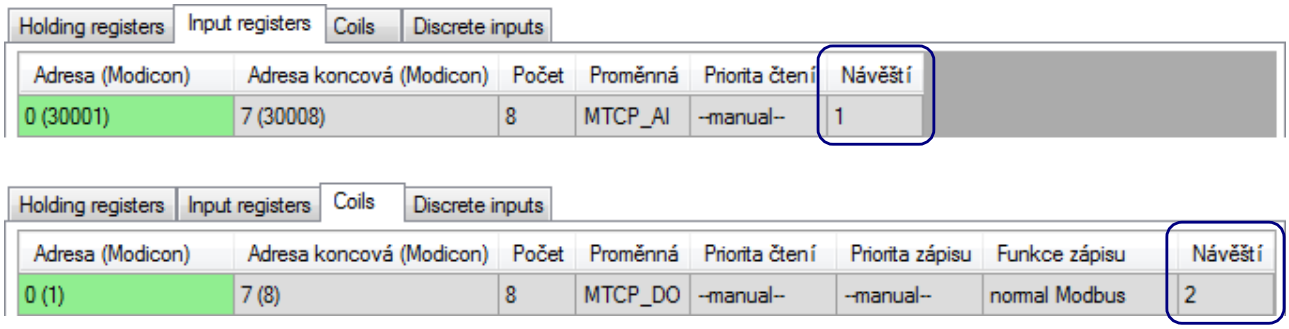
Pro manuální komunikaci je potřeba nadefinovat návěští. Návěští jsou dvojího typu:

- návěští definice Slave stanice – vlastnost **`ClientLabel`**,
- návěští definičního řádku – sloupec „Návěští“.

Platné hodnoty návěští musí být nezáporné. Návěští **`ClientLabel`** musí být unikátní v rámci aplikace, návěští v definiční tabulce datových bodů musí být unikátní v rámci dané tabulky.



Obr. 7 – Definice návěští **`ClientLabel`**



Obr. 8 – Definice návěští datových bodů

Jakmile jsou návěští definována, je možné využívat moduly **`Mdbm***`**, zmíněné v kapitole 4.1.1 „Priority komunikace“, část „Manuální priorita komunikace“.

Pro označení řádku s návěštím 1 pro čtení analogových vstupů lze použít definici:

```
MdbmRead 10, 1, MTCP_AI_rslt
         |   |   └ Výsledek provedení modulu
         |   └ Návěští definičního řádku
         └ Návěští ClientLabel
```

nebo:

```
MdbmMark 1, 4, 0, 8, MTCP_AI_rslt
         |  |  |  |   └ Výsledek provedení modulu
         |  |  |  └ Počet adres k označení
         |  |  └ Startovací adresa označení
         |  └ Definice komunikační funkce
         └ Návěští ClientLabel
```

Je zřejmé, že při použití modulu **MdbmMark** není nutné mít nadefinováno návěští na konkrétní řádek. Modul tedy umožňuje i hromadné označení většího množství definičních řádků jedné skupiny datových bodů.

Pro označení řádku s návěštím 2 pro zápis digitálních výstupů lze použít definici:

```
MdbmWrite 10, 2, MTCP_DO_rslt
          |   |  └ Výsledek provedení modulu
          |   └ Návěští definičního řádku
          └ Návěští ClientLabel
```

nebo:

```
MdbmMark 1, 5, 0, 8, MTCP_DO_rslt
```

V tomto příkladu není potřeba uvažovat nutnost použití tzv. bezpečného zápisu pomocí modulů **MdbmWrBeg** a **MdbmWrFin**, neboť daný definiční řádek slouží pouze pro zápis.

Poznámka
Na jeden definiční řádek v tabulce je možné mít zároveň nadefinovánu automatickou i manuální komunikaci. Tyto dvě komunikace se nijak nevylučují.

5.5 Stavy komunikace

Pro zjištění stavů komunikace se využívají následující moduly:

- **MdbmCliSt** – zjištění stavu komunikačního rozhraní; jako komunikační rozhraní je chápaná kombinace IP adresa + TCP port,
- **MdbmReqSt** – zjištění stavu komunikace konkrétního komunikačního požadavku.

Modul **MdbmReqSt** lze využít pro detekci rozpadu komunikace se Slave stanicí pomocí bitu č. 4 (viz text pod tabulkami). Aby bylo možné rozpad komunikace detekovat, využívá se návěští nejčastěji komunikovaného definičního řádku tabulky.

```
MdbmCliSt 10, MTCP_ClSt, MTCP_CS_rslt
          |   |          └ Výsledek provedení modulu
          |   └ Stav klienta, resp. komunikačního rozhraní
          └ Návěští ClientLabel

MdbmReqSt 10, 1, MTCP_RqSt, MTCP_RS_rslt
          |   |  |          └ Výsledek provedení modulu
          |   |  └ Stav komunikačního požadavku
          |   └ Stav definičního řádku
          └ Návěští ClientLabel
```

Využití modulu **MdbmReqSt** lze jednoznačně doporučit při ladění komunikace, kdy lze na základě hodnoty stavu komunikačního požadavku získat informaci o případné chybě při komunikaci.

Hodnota stavu komunikačního požadavku nabývá různých bitově kódovaných hodnot v závislosti na aktuálním stavu vložení požadavku na komunikaci vzdáleného bodu a na aktuálním stavu komunikace dle následující tabulky.

Bit	Význam
0	Má hodnotu 1, pokud právě probíhá komunikace.
1	Má hodnotu 1, pokud poslední ukončená komunikace skončila úspěšně.
2	Má hodnotu 1, pokud poslední ukončená komunikace skončila chybou.
4	Má hodnotu 1, pokud je další pokus o komunikaci ignorován.
6	Má hodnotu 1, pokud je požadavek označen ke čtení a čeká na komunikaci.
7	Má hodnotu 1, pokud je požadavek označen k zápisu a čeká na komunikaci.
8	Má hodnotu 1, pokud je požadavek zablokován modulem **MdbmWrBeg**.
9	Má hodnotu 1, pokud je požadavek opakovaně automaticky označen k zápisu a čeká na komunikaci.
10	Má hodnotu 1, pokud je právě komunikovaný požadavek zápis.
11	Má hodnotu 1, pokud poslední ukončená komunikace byl zápis.
12 až 15	Pokud komunikace skončila chybou (bit 2 má hodnotu 1), obsahují tyto bity kód chyby komunikace podle následující tabulky. Jinak není hodnota těchto bitů definována.

Kód chyb v bitech 12 až 15

Kód chyby	Význam
0	Stanice odpověděla negativně s blíže neurčenou chybou.
1	Stanice odpověděla: „Chybná funkce“.
2	Stanice odpověděla: „Chybná adresa registru/bináru“.
3	Stanice odpověděla: „Chybná hodnota dat.“.
4	Neznámá, blíže nespecifikovaná chyba.
5	Stanice neodpověděla během požadované doby.
6	Chyba přenosu (špatné CRC, špatná délka odpovědi, apod.).
7	Chyba navázání spojení, typicky v případě MODBUS TCP komunikace.

Vzhledem k tomu, že bit č. 4 pouze signalizuje ignoraci následné komunikace, lze doporučit využití modulu **RS** pro signalizaci rozpadu komunikace s danou Slave stanicí:

```
RS MTCP_RS_rslt.4, MTCP_RS_rslt.1, MTCP_Problem.0
```

5.6 Příklad parametrizace řídicího systému jako Master

Uvažujme aplikaci, ve které jeden řídicí systém **AMiNi4DW2** má sloužit jako Slave stanice pro druhý řídicí systém **AMiNi4DW2**. Rozložení registrů Slave stanice je uvedeno v kapitole 6.4 „Příklad parametrizace řídicího systému jako Slave“.

Je známé rozložení holding registrů:

- adresa 0 – DI,
- adresy 1 až 8 – AI_Integer,
- adresy 10 až 25 – AI_Float,
- adresa 100 – DO,
- adresy 101 až 104 – AO.

Nejdříve vytvoříme proměnné, které budou přiřazeny daným definičním řádkům:

- **AMiNi_DI** – typu I,
- **AMiNi_AI** – typu MI, dimenze [1×8],
- **AMiNi_AI_F** – typu MF, dimenze [1×8],
- **AMiNi_DO** – typu I,
- **AMiNi_AO** – typu MI, dimenze [1×4].

Dalším krokem je vytvoření definičního uzlu **ModbusMaster0** a v něm vytvoření definice **ModbusDevice0**. IP adresa řídicího systému nechť je 192.168.1.2, komunikační port výchozí 502.

V definici tabulky **ModbusDevice0** nadefinujeme 5 řádků na záložce „Holding registers“. V definičních řádcích zvolíme priority například takto:

- adresa 0 – automatické čtení s prioritou **Normal**,
- adresy 1 až 8 – automatické čtení s prioritou **Low**,
- adresy 10 až 25 – automatické čtení s prioritou **Low**,
- adresa 100 – manuální zápis,
- adresy 101 až 104 – automatický zápis.

Protože je nadefinován manuální zápis a je požadavek na zjišťování stavu připojení Slave stanice, nadefinujeme v definici **ModbusDevice0** parametr **ClientLabel,** např. na hodnotu 10. Pro zjišťování stavu připojení Slave stanice nadefinujeme návěští v řádku registru s adresou 0 a pro manuální spouštění komunikace nadefinujeme návěští i v řádku registru s adresou 100.

Vzhledem k tomu, že produkty firmy AMiT podporují komunikační rámce 6 i 16, je ve sloupci „Funkce zápisu“ zachována volba **normal Modbus**.

Výsledná tabulka vypadá dle následujícího obrázku.

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Priorita zápisu	Funkce zápisu	Návěští
0 (40001)	0 (40001)	1	AMiNi_DI	Normal	--manual--	normal Modbus	1
1 (40002)	8 (40009)	8	AMiNi_AI	Low	--manual--	normal Modbus	
10 (40011)	25 (40026)	16	AMiNi_AI_F	Low	--manual--	normal Modbus	
100 (40101)	100 (40101)	1	AMiNi_DO	--manual--	--manual--	normal Modbus	2
101 (40102)	104 (40105)	4	AMiNi_AO	--manual--	Auto	normal Modbus	

Obr. 9 – Základní definice datových bodů a přiřazených proměnných

Posledním krokem je nadefinování modulů **MdbmReqSt** a **MdbmWrite**. Abychom zamezili nadbytečným komunikacím registru 100, využijeme algoritmus uvedený v kapitole 5.4 „Manuální komunikace“.

Výsledný kód v periodickém procesu bude znít:

```
MdbmReqSt 10, 1, AMiNi_RqSt, AMiNi_RS_rs

If not AMiNi_RqSt.4
  If AMiNi_DO != AMiNi_DO_pr
    Let AMiNi_DO_pr = AMiNi_DO
    MdbmWrite 10, 2, AMiNi_DO_rs
  EndIf
EndIf
```

Poznámka
V kódu aplikace by mělo být následně řešeno přepočítávání hodnot analogových veličin, což se však v tomto příkladu neřeší.

Uvedený algoritmus je součástí přílohy ap0059_ap_cz_xxx.zip. Jedná se o ukázkovou aplikaci s názvem „modbstcp_p1_cz_xxx.dso“ vytvořený ve vývojovém prostředí DetStudio. Aplikace je vytvořena pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém s písmenem „W“ v názvu pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

5.7 Master komunikace s více než 3 Slave stanicemi

V případě, že se má komunikovat s více než třemi Slave stanicemi, dochází vlivem omezení implementovaného TCP stacku k přepínání mezi jednotlivými definovanými IP adresami.

Za takovýchto okolností může docházet k občasným výpadkům komunikace některých definičních řádků dle zvolené priority.

V případě, že k popisovanému chování opakovaně dochází, je doporučeno pro odstranění zmíněného náhodného chování rozdělit jednotlivé Slave stanice do skupin po jednom až třech stanicích a dané skupiny postupně komunikovat s manuální prioritou komunikace.

K tomuto se dá s výhodou využít modulu **`Switch`** v kombinaci s pomocnou počítací proměnnou.

Jako příklad použijeme komunikaci s šesti řídicími systémy **AMiNi4DW2** s nahranou aplikaci popsanou v kapitole 6.4 „Příklad parametrizace řídicího systému jako Slave“. Jako základ bude využita aplikace vytvořená v předchozí kapitole.

Aby nebylo potřeba definovat pro každou Slave stanici vlastní kompletní sadu proměnných, využije se vlastnosti přiřazené maticové proměnné, kdy načtené registry se zapisují po sloupcích a až po zápisu do posledního sloupce se pokračuje na dalším řádku. Načítací proměnné upravíme na:

- **`AMiNi_DI`** – typu MI, dimenze [6×1],
- **`AMiNi_AI`** – typu MI, dimenze [6×8],
- **`AMiNi_AI_F`** – typu MF, dimenze [6×8].

Pro zápis DO se využívá porovnávání s předchozí hodnotou a proto mohou být i příslušné proměnné převedeny na maticové:

- **`AMiNi_DO`** – typu MI, dimenze [6×1],
- **`AMiNi_DO_pr`** – typu MI, dimenze [6×1],

U zápisu AO by šlo provést taktéž převod do matice o šesti řádcích. Změna na jedinou maticovou proměnnou by však znamenala nutnost současného manuálního periodického zápisu do všech Slave stanic. Aby byla zachována funkčnost původní **`Auto`** priority zápisu, bude využit modul **`VarWStat`** pro zjišťování zápisu do proměnné a až na základě této informace dojde k zápisu do příslušné Slave stanice. Pro korektní funkčnost musí mít každá Slave stanice vlastní **`AMiNiX_AO`** proměnnou.

Po úpravě proměnných je potřeba v definici Slave stanice upravit přiřazené proměnné, resp. buňky definující pracovní řádek dané proměnné a nastavit priority čtení i zápisu na **`--manual--`**.

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Priorita zápisu	Funkce zápisu	Návěští
0 (40001)	0 (40001)	1	AMiNi_DI[0,0]	--manual--	--manual--	normal Modbus	1
1 (40002)	8 (40009)	8	AMiNi_AI[0,0]	--manual--	--manual--	normal Modbus	
10 (40011)	25 (40026)	16	AMiNi_AI_F[0,0]	--manual--	--manual--	normal Modbus	
100 (40101)	100 (40101)	1	AMiNi_DO[0,0]	--manual--	--manual--	normal Modbus	
101 (40102)	104 (40105)	4	AMiNi1_AO	--manual--	--manual--	normal Modbus	

Obr. 10 – Upravená definice přiřazených proměnných

Po úpravě definice Slave stanice dle tohoto vzoru vytvoříme dalších pět definic. V pěti **`ModbusMasterX`** uzlech vytvoříme vždy po jednom uzlu **`ModbusDeviceX`**. U všech musí být nadefinované vlastnosti **`ClientLabel`** s unikátními hodnotami.

Následuje vytvoření algoritmu pro postupnou komunikaci s jednotlivými Slave stanicemi.

Nejprve je doporučeno vypočítat maximální dobu komunikace v nejhorším případě komunikace všech datových bodů a bez sdružování komunikačních rámců:

$T = (5 + 0{,}2 \times 1) + (5 + 0{,}2 \times 8) + (5 + 0{,}2 \times 16) + (5 + 0{,}2 \times 1) + (5 + 0{,}2 \times 4) = 31$ ms

Dále zvolíme skupiny Slave stanic pro komunikaci. V tomto příkladu zvolíme, že v jedné skupině je pouze jedna stanice. Z předchozího výpočtu plyne, že daný algoritmus přepínání komunikací Slave

stanic může být v periodickém procesu o periodě minimálně 31 ms. Zvolíme, že daný algoritmus bude v běžném periodickém procesu. Běžné periodické procesy mají nejmenší možnou periodu 100 ms. Nelze však použít tuto nejnižší dobu jako periodu procesu, neboť doba navázání TCP spojení trvá nějaký čas. Minimální doporučitelná doba je 3 000 ms.

Dalším krokem je zápis kódu pro komunikaci se stanicí. Kód může vypadat následovně.

```
// Periodické čtení
MdbmMark 10, 3, 0, 26, NONE
// Událostní zápisy
// DO
If AMiNi_DO[0,0] != AMiNi_DO_pr[0,0]
     Let AMiNi_DO_pr[0,0] = AMiNi_DO[0,0]
     MdbmMark 10, 6, 100, 1, NONE
EndIf
// AO
VarWStat AMiNi1_AO, @AO_w, 0
If @AO_w
     MdbmMark 10, 6, 101, 4, NONE
EndIf
```

Posledním krokem je rozkopírování kódu do šestí **Case** větví modulu **Switch**, úpravy jednotlivých kódů, aby byly korektně navázány parametry **Client**, buňky matic a přiřazené proměnné a zajištění postupného přepínání větví.

```
Switch SlaveNdx
     Case 0
          // Periodické čtení
          MdbmMark 10, 3, 0, 26, NONE
          …
     EndCase

     Case 1
          // Periodické čtení
          MdbmMark 20, 3, 0, 26, NONE
          …
     EndCase

     Case 2
          …
     EndCase

     …

     Case 5
          …
     EndCase

EndSwitch

Let SlaveNdx = If(SlaveNdx < 5, SlaveNdx + 1,0)
```

Uvedený algoritmus je součástí přílohy ap0059_ap_cz_xxx.zip. Jedná se o ukázkovou aplikaci s názvem „modbstcp_p2_cz_xxx.dso“ vytvořený ve vývojovém prostředí DetStudio. Aplikace je vytvořena pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém s písmenem „W“ v názvu pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

5.8 Master komunikace s více než 3 Slave stanicemi včetně ošetření nenavázání TCP spojení

Jednoduchý algoritmus uvedený v předchozí kapitole je však možné použít pouze v případě, že je garantováno, že Slave stanice jsou neustále k dispozici pro TCP komunikaci. V případě, kdy může docházet k rozpadům komunikací např. vlivem výpadků napájení či přetížení Ethernet sítě, může v závislosti na délce tohoto výpadku dojít i k rozpadu komunikace s ostatními připojenými Slave stanicemi ze strany řídicího systému.

Aby k tomuto nedošlo, musí se zajistit, aby se po rozpadu TCP spojení s jednou Slave stanicí následný pokus o komunikaci s touto stanicí provedl až za delší dobu. Specifikace TCP protokolu definuje tzv. „MSL“ čas s délkou 2 minuty. Avšak pro kompletní přechod z TCP stavu „TIME-WAIT“ je potřeba počkat alespoň dvojnásobek této doby, tzv. dobu „2MSL“. Aby bylo zajištěno kompletní ukončení komunikace na tomto TCP spojení, počká se ještě jednu minutu navíc. Po rozpadu komunikace se tedy další pokus o komunikaci s touto Slave stanicí má provést nejdříve až za 5 minut.

Za standardních okolností je v řídicím systému rezervována kapacita pro tři TCP spojení. Z tohoto lze vyvodit, že v případě, že se komunikace s jednou Slave stanicí rozpadne, bude jedno z těchto TCP spojení obsazeno po dobu minimálně výše zmíněného času „2MSL“. Tedy rozpad komunikace s jednou Slave stanicí není pro řídicí systém kritický. V případě, že dojde k rozpadu i s druhou Slave stanicí, může to znamenat vážnější problémy na síti a za takové situace je doporučeno provést kompletní odmlčení na všech TCP spojeních po doporučovanou dobu 5 minut.

Protože TCP handshake může trvat nějakou dobu, je potřeba po vyslání požadavku na komunikaci čekat jistou dobu pro samotné vykomunikování. Časové poměry zmíněné v kapitole 4.1 „Doba komunikace“ platí pro již navázané TCP spojení. S rezervou lze doporučit čas 10 sekund jako maximální čas čekání mezi aktivací komunikačního požadavku na novou Slave stanici a vyhodnocením, zda bylo TCP spojení úspěšně navázáno.

Vzhledem k tomu, že u zápisového požadavku dochází v případě neúspěšné komunikace k opakovaným pokusům o komunikaci (viz kapitola 4.2 „Komunikace při rozpadu spojení“), musí být zajištěno, aby se nejprve provedl pokus o vyčtení hodnoty některého registru či bináru a až v případě úspěšné komunikace byly provedeny následující komunikace čtecích a zápisových požadavků.

Na základě výše uvedených informací se v předchozí kapitole vytvořený algoritmus upraví tak, aby pro jednu Slave stanici byly dvě **`Case`** větve řešící první pokusnou čtecí komunikaci a v případě úspěšného TCP spojení pak následovně byla provedena druhá komunikace zbývajících registrů či binárů. V případě, že TCP spojení nebylo v pořádku, bude následné chování řízeno tím, jestli se jedná o první nebo druhou rozpadlou TCP komunikaci. V případě prvního rozpadu se provede uložení indexu této problematické Slave stanice pro její následovné přeskakování. V případě druhého rozpadu se celý komunikační algoritmus dočasně přeruší. Zjišťování stavu TCP spojení je prováděno modulem **`MdbmCliSt`**.

Algoritmus pro obsluhu jedné Slave stanice může vypadat následovně:

```
// jedna Slave stanice je definována dvěma Case stavy
// první Case stav spouštějící komunikaci se Slave stanicí
Case 0
     If TCP_SkipSl != 0
          // má se komunikovat
               // spuštění první manuální čtecí komunikace s danou Slave stanicí
          MdbmRead 10, 1, NONE
          Let @SecondComm = false // jedná se o první komunikaci se Slave stanicí
          Let TCP_CliSt = 0 // nulování posledního stavu TCP komunikace
               // čas pro vytvoření TCP spojení a vyslání rámce
          Let TCP_Wait = TCP_Wait_10s
          Let TCP_Ndx = 1 // přechod na stav vyhodnocení komunikace
     Else
```

```
          // první rozpad TCP - tato Slave stanice se přeskakuje
          Let TCP_Ndx = 2
     EndIf
EndCase

// druhý Case stav ověřující, že došlo ke korektnímu TCP spojení
Case 1
     // zjištění stavu TCP spojení
     MdbmCliSt 10, TCP_CliSt, NONE
     If (TCP_Wait > 0)
          If TCP_CliSt == 2
               // TCP spojení proběhlo v pořádku
               If @SecondComm
                    // i zbývající komunikace v pořádku
                    // - přechod na další Slave stanici
                    Let TCP_Ndx = 2
               Else
                    // druhá komunikace - všechny zbývající čtecí a zápisové registry
                    Let @SecondComm = true

                    // Zbývající čtení
                    MdbmMark 10, 3, 1, 25, NONE
                    // Událostní zápisy
                    // DO
                    If AMiNi_DO[0,0] != AMiNi_DO_pr[0,0]
                         Let AMiNi_DO_pr[0,0] = AMiNi_DO[0,0]
                         MdbmMark 10, 6, 100, 1, NONE
                    EndIf
                    // AO
                    VarWStat AMiNi1_AO, @AO_w, 0
                    If @AO_w
                         MdbmMark 10, 6, 101, 4, NONE
                    EndIf

                    Let TCP_Wait = TCP_Wait_10s    // čas pro vykomunikování

                    Let TCP_CliSt = 0
               EndIf
          EndIf
     Else
          // ani za 10 s nedošlo ke korektnímu TCP spojení
          // -  problém při TCP komunikaci
          If TCP_SkipSl == -1
               // první rozpad TCP - bude se přeskakovat tato Slave stanice
                    // uložení problematické Slave stanice (hodnota Case)
               Let TCP_SkipSl = 0
               Let TCP_Ndx = 2 // přechod na další Slave stanici
          Else
               // druhý rozpad TCP - kompletní odmlčení na 5 minut
               Let TCP_Ndx = 102 // hodnota začínající 100 značí chybu
          EndIf
          // nastavení 5 minutového timeoutu
          Let TCP_TMO = TCP_TMO_5m
     EndIf
EndCase
```

Uvedený algoritmus je součástí přílohy ap0059_ap_cz_xxx.zip. Jedná se o ukázkovou aplikaci s názvem „modbstcp_p4_cz_xxx.dso“ vytvořený ve vývojovém prostředí DetStudio. Aplikace je vytvořena pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém s písmenem „W“ v názvu pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

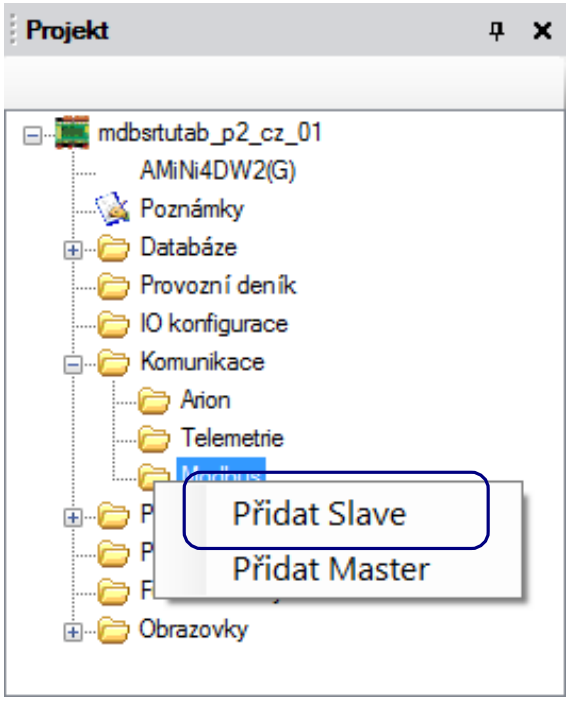
6 Řídicí systém jako Slave

Definice komunikace protokolem MODBUS v roli Slave probíhá pomocí dvojí definice:

- vytvoření komunikační definice protokolu v roli Slave,
- definice datových bodů pro komunikaci.

6.1 Komunikační definice

Vytvoření komunikační definice protokolu MODBUS v roli Slave reprezentuje vložení definice komunikační položky **`ModbusSlave`** do aplikace. Vložení se provádí v DetStudiu v okně Projekt v uzlu „Projekt/Komunikace/Modbus“. Po vyvolání kontextového menu nad touto položkou se vybere položka **`Přidat Slave`**.



Obr. 11 – Položka „Přidat Slave“ v definici komunikace „Modbus“

Po vložení definice bude vytvořen komunikační uzel **`Modbus0`** s výchozími hodnotami vlastností:

- **`Address`**: 1
- **`BaudRate`**: 19200
- **`DataBits`**: 8
- **`LastError`**: NONE
- **`Mode`**: SerialLineRTU
- **`Parity`**: Even
- **`SerialPort`**: 0
- **`StopBit`**: One

Platí, že pro komunikaci protokolem MODBUS TCP je potřeba nastavit hodnotu vlastnosti **`Mode`** na „Modbus_TCP“.

Popis ostatních vlastností je k dispozici v nápovědě DetStudia „PseDet – tvorba regulačních procesů“ v kapitole „Obsah/Komunikace/Modbus/Slave – založení a nastavení obecných parametrů“.

6.2 Definice datových bodů pro komunikaci

Po nadefinování uzlu **ModbusX** je možné na něj v okně Projekt dvakrát kliknout a tím vyvolat definiční tabulku komunikačních datových bodů.

Do tabulky se definují v jednotlivých záložkách skupin datových bodů (Holding registers, Input registers, Coils a Discrete inputs) proměnné řídicího systému, které mají být dostupné pod zvolenými adresami.

Definici jednotlivých datových bodů lze provést např. přetažením z okna ToolBox. Více informací o definici datových bodů lze nalézt v nápovědě DetStudia „PseDet – tvorba regulačních procesů“ v kapitole „Obsah/Komunikace/Modbus/Slave – editor tabulky“.

6.3 Stavy komunikace

Stav komunikace ze strany Mastera je k dispozici po přiřazení proměnné vlastnosti **LastError** v definici komunikace **ModbusX**. Očekává se proměnná typu I.

Zmíněnou vlastnost doporučujeme využívat zvláště při ladění komunikace, kdy lze na základě její hodnoty získat informaci o případné chybě při komunikaci. Přiřazená proměnné bude nabývat hodnot dle následující tabulky.

Kód chyby	Význam
0	Bez chyby.
1	Příliš nízká verze NOS.
2	Chyba alokace systémového časovače.
3	Chyba alokace komunikačního portu.
4	Poslední přijatý rámec měl špatný kontrolní součet.
5	Poslední přijatý rámec měl chybnou délku.
6	Poslední přijatý rámec obsahoval požadavek na nenamapovanou adresu.
7	Poslední přijatý rámec obsahoval požadavek na nepodporovanou funkci.
8	Poslední přijatý rámec požadoval více dat, než se vejde do rámce odpovědi.
9	Poslední přijatý rámec obsahoval špatná data (funkce 6 ON, OFF).
10	Mezi příchozími znaky je příliš veliká mezera.
11	Chyba při ASCII příjmu : – příliš dlouhý rámec, – neočekávaný znak (uvnitř rámce musejí být pouze textově hexa cifry), – po CR nepřišlo LF.
12	Modul nebyl dosud spuštěn (nedošlo k vyhodnocení parametrů a alokoaci komunikačního portu).

6.4 Příklad parametrizace řídicího systému jako Slave

Je požadavek na vytvoření aplikace, po jejímž nahrání bude řídicí systém **AMiNi4DW2** pracovat jako modul vzdálených vstupů a výstupů komunikující protokolem MODBUS TCP. Zároveň má být aplikace dostatečně univerzální, aby bylo možné analogové hodnoty komunikovat v desetinné podobě i v celočíselné podobě.

7. Zvolíme IP adresu řídicího systému. Např. 192.168.1.2.

8. Vytvoříme proměnné, dle následující tabulky.

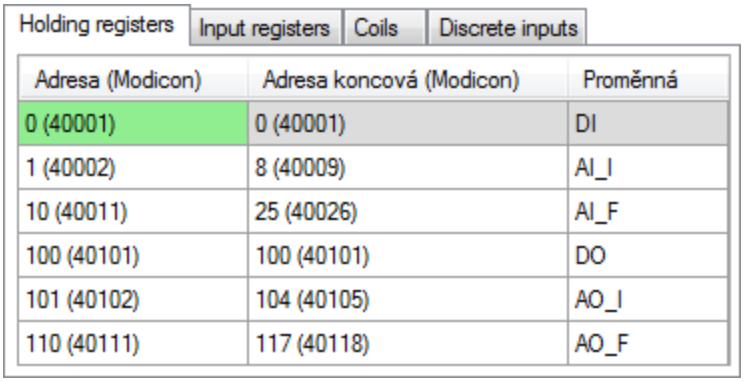
Proměnná	Typ	Komentář
DI	I	Digitální vstupy.
AI_I	MI[1,8]	Integer hodnoty analogových vstupů.
AI_F	MF[1,8]	Float hodnoty analogových vstupů.
DO	I	Digitální výstupy.
AO_I	MI[1,4]	Integer hodnoty analogových výstupů.

Proměnná	Typ	Komentář
AO_F	MF[1,4]	Float hodnoty analogových výstupů.
tempF	F	Pomocná Float proměnná.
Mdbs_Err	I	Kód poslední chyby při komunikaci.

9. Zvolíme adresy a typy datových bodů, které mají reprezentovat dané proměnné dle následující tabulky.

Proměnná	Adresa	Typ datového bodu
DI	0	Holding register
AI_I	1 až 8	Holding register
AI_F	10 až 25	Holding register
DO	100	Holding register
AO_I	101 až 104	Holding register
AO_F	110 až 117	Holding register

10. Na základě předchozí tabulky nadefinujeme v uzlu **`Modbus0`** definiční řádky.



Obr. 12 – Definice MODBUS Slave datových bodů a přiřazených proměnných

11. V periodickém procesu vytvoříme algoritmus, který bude načítat vstupy a zapisovat výstupy na základě hodnot proměnných. Kód může vypadat následovně.

```
// ------------------------------------    DI    ------------------------------------

DigIn #0, DI, 0x0000

// ------------------------------------    AI    ------------------------------------

// AI0 a AI1 jako Ni1000 / 6180 ppm
// Teplota 12,45°C odpovídá hodnotě 1245 v Int registru
Ni1000 #Ni10001_0, AI_F[0,0], 6180
Let AI_I[0,0] = Int(AI_F[0,0] * 100)

Ni1000 #Ni10001_1, AI_F[0,1], 6180
Let AI_I[0,1] = Int(AI_F[0,1] * 100)

// AI2 a AI3 jako Pt1000 / 3850 ppm
// Teplota 12,45°C odpovídá hodnotě 1245 v Int registru
Pt1000 #Ni10001_2, AI_F[0,2], 3850
Let AI_I[0,2] = Int(AI_F[0,2] * 100)

Pt1000 #Ni10001_3, AI_F[0,3], 3850
Let AI_I[0,3] = Int(AI_F[0,3] * 100)

// AI4 a AI5 pro měření napětí 0 až 10 V
// Hodnota 3,678 V odpovídá hodnotě 3678 v Int registru
```

```
AnIn #AI00_4, AI_F[0,4], 10.000, 0.000, 10.000, 0.000, 10.000
Let AI_I[0,4] = Int(AI_F[0,4] * 1000)

AnIn #AI00_5, AI_F[0,5], 10.000, 0.000, 10.000, 0.000, 10.000
Let AI_I[0,5] = Int(AI_F[0,5] * 1000)

// AI6 a AI7 pro měření proudu 0(4) až 20 mA
// Hodnota 15,345 mA odpovídá hodnotě 15345 v Int registru
AnIn #AI00_6, AI_F[0,6], 20.000, 0.000, 20.000, 0.000, 20.000
Let AI_I[0,6] = Int(AI_F[0,6] * 1000)

AnIn #AI00_7, AI_F[0,7], 20.000, 0.000, 20.000, 0.000, 20.000
Let AI_I[0,7] = Int(AI_F[0,7] * 1000)

// -----------------------------------    DO    -----------------------------------

DigOut DO, #0, 0x0000

// -----------------------------------    AO    -----------------------------------

// AO0 až AO3 s výstupem 0 až 10 V
// Žádaná hodnota 2,456 V musí být zapsána v Int registru jako hodnota 2456
// Buď se využívají Float registry 110-111 až 116-117 nebo Int registry 101 až 104
Let tempF = If(AO_F[0,0] == 0, AO_I[0,0] / 1000, AO_F[0,0])
AnOut #AO00_0, tempF, 10.000, 0.000, 10.000, 0.000, 10.000

Let tempF = If(AO_F[0,1] == 0, AO_I[0,1] / 1000, AO_F[0,1])
AnOut #AO00_1, tempF, 10.000, 0.000, 10.000, 0.000, 10.000

Let tempF = If(AO_F[0,2] == 0, AO_I[0,2] / 1000, AO_F[0,2])
AnOut #AO00_2, tempF, 10.000, 0.000, 10.000, 0.000, 10.000

Let tempF = If(AO_F[0,3] == 0, AO_I[0,3] / 1000, AO_F[0,3])
AnOut #AO00_3, tempF, 10.000, 0.000, 10.000, 0.000, 10.000
```

Uvedený algoritmus je součástí přílohy ap0059_ap_cz_xxx.zip. Jedná se o ukázkovou aplikaci s názvem „modbstcp_p3_cz_xxx.dso“ vytvořený ve vývojovém prostředí DetStudio. Aplikace je vytvořena pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém s písmenem „W“ v názvu pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

7 Dodatek A

7.1 Kompatibilita s inicializací komunikace pomocí modulů

7.1.1 MODBUS Master

MODBUS TCP Master komunikaci není možné definovat pomocí modulů v inicializačním nebo periodickém procesu.

7.1.2 MODBUS Slave

Pro inicializaci MODBUS TCP Slave komunikace je možné využívat i moduly **`MODBS_Var`** a **`MODBS_ISl`**, jenž se typicky umisťují do inicializačního procesu.

Z pohledu interní funkčnosti se jedná o identickou definici komunikace, jako je definice tabulkou. Z tohoto důvodu **lze** kombinovat obě definice komunikace na stejném komunikačním rozhraní.

Více informací o tomto se lze nalézt v nápovědě DetStudia „PseDet – tvorba regulačních procesů“ v kapitole „Obsah/Komunikace/Modbus“ v části „Zpětná kompatibilita“.

8 Technická podpora

Veškeré informace ohledně komunikace v síti MODBUS TCP v řídicích systémech firmy AMiT, Vám poskytne oddělení technické podpory firmy AMiT. Technickou podporu můžete kontaktovat nejlépe prostřednictvím emailu na adrese **support@amit.cz**.

9 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.