Komunikace v síti MODBUS RTU (PseDet)

Abstrakt

Aplikační poznámka popisuje použití MODBUS RTU protokolu v PseDet řídicích systémech pomocí definice tabulkou. A to jak pro komunikaci s jinými výrobky firmy AMiT pracujících v režimu Slave, tak i pro komunikaci s nadřazenou Master jednotkou.

Autor: Zbyněk Říha
Dokument: ap0008_cz_06.pdf

Příloha

Obsah souboru: ap0008_cz_06.zip

modbs_p1_cz_06.dso	Příklad parametrizace řídicího systému jako master.
modbs_p2_cz_05.dso	Příklad parametrizace řídicího systému jako slave.
modbs_p3_cz_05.dso	Programová obsluha **DMM-DI24**.
modbs_p4_cz_05.dso	Programová obsluha **DMM-DO18**.
modbs_p5_cz_05.dso	Programová obsluha **DMM-RDO12**.
modbs_p6_cz_05.dso	Programová obsluha **DMM-AI12**.
modbs_p7_cz_05.dso	Programová obsluha **DMM-AO8x**.
modbs_p8_cz_05.dso	Programová obsluha **DMM-PDO6Ni6**.
modbs_p9_cz_02.dso	Programová obsluha **DMM-UI8DO8**.
modbs_p10_cz_02.dso	Programová obsluha **DMM-UI8RDO8**.
modbs_p11_cz_02.dso	Programová obsluha **DMM-UI8AO8U**.
modbs_p12_cz_02.dso	Programová obsluha **AMR-OP7x(RH) / AMR-OP6x / AMR-OP4x / AMR-OP3xA(RH)**.
modbs_p13_cz_01.dso	Programová obsluha **AMR-OP7xC**.
modbs_p14_cz_01.dso	Programová obsluha **AMR-OP7xRHC**.
modbs_p15_cz_01.dso	Programová obsluha **AMR-OP40(RH)C**.

Obsah

Historie revizí

Verze	Datum	Autor změny	Změny
001	30. 04. 2008	Říha Z.	Nový dokument.
002	13. 04. 2010	–	Upraveny popisky u příkladů použití modulů, upraveny ukázkové aplikace, opravena tabulka s výsledky komunikací, změna kapitoly 7.
003	20. 04. 2012	–	Do kapitoly 7.1.1 doplněna informace o použití matic v modulu RmtDef. Ukázkové aplikace vytvořeny v DetStudiu verze 1.7.0.
004	05. 08. 2013	–	Úprava kapitoly 5.2, doplnění nových příkladů, úprava popisu příkladů v kapitole 8 Příklady vytvořeny v DetStudiu 1.7.3.44.
005	27. 01. 2015	–	Úprava související dokumentace, kapitol 7.1.5, 8.11, 8.12 a obrázků.
006	05. 02. 2019	Kupčík M.	Celková revize obsahu dokumentu.

Související dokumentace

1. Nápověda k části PseDet vývojového prostředí DetStudio
   soubor: PseDet_cs.chm
2. Katalogový list k **DMM-xxx**
   soubor: dmm-xxx_d_cz_xxx.pdf
3. Návod na obsluhu k **AMR-OPxx**
   soubor: amr-opxx_g_cz_xxx.pdf
4. Popis základní typové aplikace k **AMR-OP3xA**
   soubor: ta-op3x_fw01m_cz_xxx.pdf
5. Aplikační poznámka AP0016 – Zásady používání RS485
   soubor: ap0016_cz_xx.pdf

1 Definice použitých pojmů

PseDet řídicí systém
Jedná se o řídicí systémy a terminály firmy AMiT, u kterých se algoritmy procesů programují v tzv. PseDet součásti prostředí DetStudio. Např. **AMiNi4DW2**, **ADiS**, **AMAP99W3** nebo **APT4000W3**.

Master stanice
Tato stanice aktivně komunikuje se Slave stanicemi. Na jednom komunikačním rozhraní je pouze jediná Master stanice.

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. Slave stanic může být až 247.

RS485
Je poloduplexní sériová sběrnice, umožňující komunikaci více stanic na jednom signálovém páru. Maximální počet připojených stanic na jednom segmentu závisí na typu zařízení. Může se pohybovat v rozsahu 32 až 256. Více informací nalezete v dokumentu AP0016 – Zásady používání RS485.

Datový bod
Jedná se o definici registru (vstupního či výstupního) 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.

Moduly DMM-xxx
Moduly umožňují prostřednictvím protokolu MODBUS RTU rozšířit počet vstupů a výstupů u zařízení, které je naprogramováno jako MODBUS Master. Do jedné sítě MODBUS lze připojit až 63 modulů.

Nástěnné ovladače AMR-OPxx
Nástěnné ovladače mohou na rozhraní RS485 naslouchat jako MODBUS RTU Slave stanice buď díky přednahrané aplikaci z výroby, nahráním určené typové aplikace nebo vytvořením vlastní aplikace, ve které je použit komunikační objekt **`ModbusSlave`** nebo **`SerialBusN`**.

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. 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.

Na druhou stranu protokol přesně definuje např. časové poměry na síti a chybové odpovědi. V případě, že tyto předpisy nebude protistrana respektovat, nebude komunikace s takovými zařízeními fungovat.

Některé implementace protokolu MODBUS podporují dokonce i Multimastering, u systémů AMiT však toto podporováno není. Firma AMiT podporuje komunikaci protokolem MODBUS v rozšiřujících modulech a komunikačních převodnících řady **DMM-xxx** (MODBUS RTU) a ve všech svých řídicích systémech, řídicích terminálech a programovatelných regulátorech a ovladačích vybavených rozhraním RS485 nebo RS232 (MODBUS RTU, v případě PseDet řídicích systémů i MODBUS ASCII). Určení master/slave závisí na konkrétní implementaci.

Poznámka
Komunikační rozhraní řídicího systému, kam je připojena síť MODBUS, již nelze použít pro připojení zařízení komunikujícího jiným komunikačním protokolem.

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“.

2.1 Podporované funkce MODBUS

V PseDet 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 stavu binárních výstupů (coils).
2	Čtení binárních vstupů.
3	Čtení výstupních (holding) registrů.
4	Čtení vstupních (input) registrů.
5	Nastavení jednoho binárního výstupu (coil).
6	Nastavení jednoho výstupního (holding) registru.
15	Nastavení binárních výstupů (coils).
16	Nastavení výstupních (holding) 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 o tomto se dočtete v nápovědě DetStudia „PseDet – tvorba regulačních procesů“ v kapitole „Obsah/Komunikace/Modbus“ v části „Mapování komunikačních bodů“.

3 Zapojení komunikační sítě

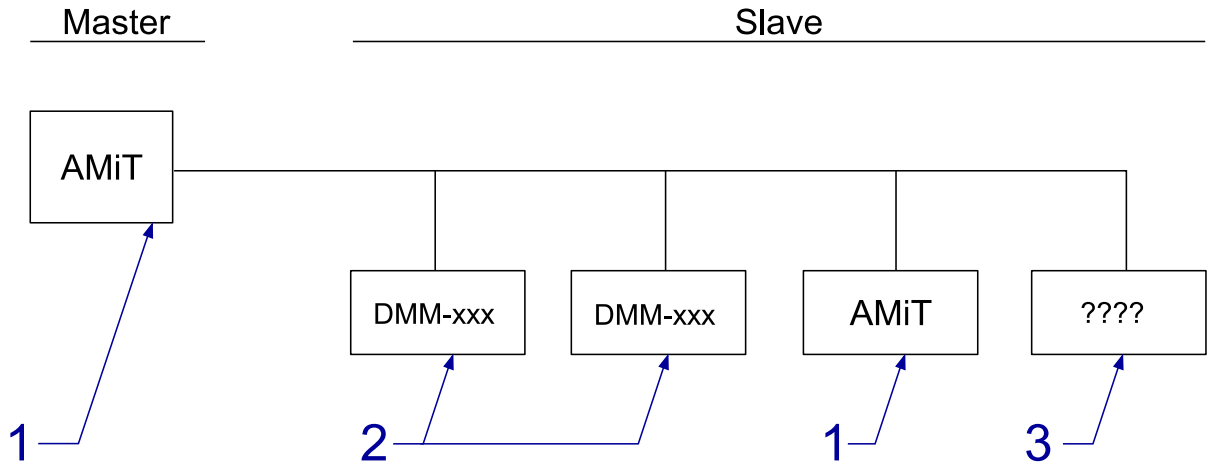
Pro správné plnění požadované funkce celé sítě MODBUS je nutno správně navrhnout, zapojit, nakonfigurovat jednotlivé moduly sítě a naprogramovat komunikaci v řídicích systémech.

Komunikace prostřednictvím protokolu MODBUS probíhá typicky po síti RS485. K řídicímu systému lze jiné zařízení (např. moduly **DMM-xxx** z produkce firmy AMiT) připojit přímo na rozhraní RS485 nebo na rozhraní RS232 přes převodník (např. **DM-232TO485**).

Při zapojování sítě na sériovém rozhraní RS485 je nutno dodržet doporučení uvedená v dokumentu AP0016 – Zásady používání RS485.

Řídicí systém AMiT může v sítí MODBUS vystupovat jako Master nebo jako Slave. V roli Master typicky v kombinaci s **DMM-xxx** moduly, nástěnnými ovladači **AMR-OPxx** a technologickými zařízeními třetích firem (např. akční členy) nebo v roli Slave jako součást rozsáhlejších sítí.

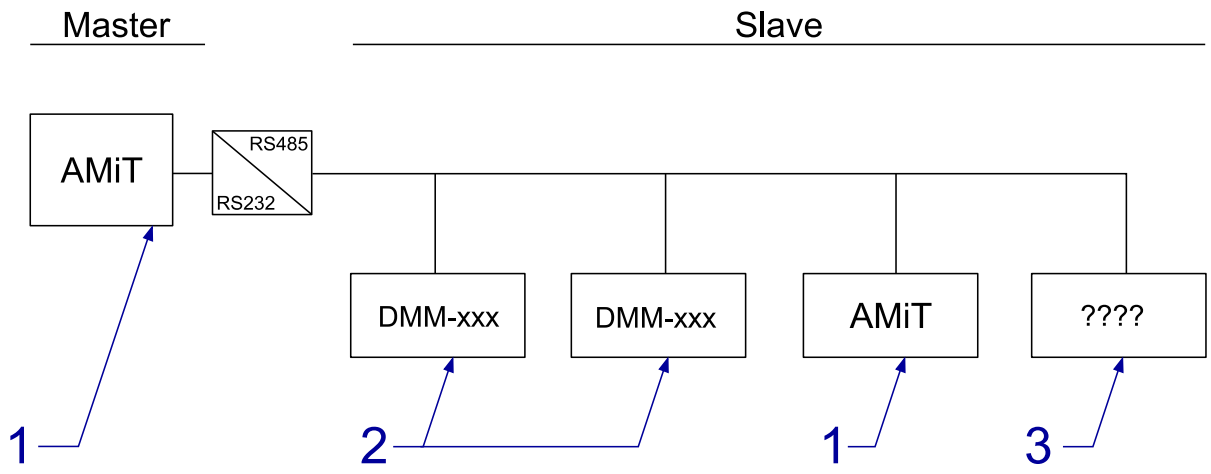
Rozšiřující moduly a komunikační převodníky s protokolem MODBUS označované **DMM-xxx** mají v síti vždy roli Slave.



Obr. 1 – Komunikace pomocí protokolu MODBUS přímo na síti RS485

Legenda

Číslo	Význam
1	Řídicí systém AMiT
2	Rozšiřující vstupně/výstupní moduly firmy AMiT
3	Zařízení jiných výrobců jako Slave stanice



Obr. 2 – Komunikace pomocí protokolu MODBUS přes převodník RS232 na RS485

Legenda

Číslo	Význam
1	Řídicí systém AMiT
2	Rozšiřující vstupně/výstupní moduly firmy AMiT
3	Zařízení jiných výrobců jako Slave stanice

Poznámka
*Převodník **DM-232TO485** připojený k RS232 řídicího systému AMiT se nastavuje jako řízený signálem RTS.*

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

Master postupně zasílá požadavky na jednotlivé datové body v síti a přijímá odpovědi. Doba komunikace vzdálených bodů je dána použitou komunikační funkcí a množstvím přenášených dat. Obecně lze dobu komunikace vypočíst s pomocí následující tabulky.

Přenosová rychlost [bps]	Minimální doba komunikace [ms]	Doba komunikace datového bodu [ms]
9600	40	3× registr, 1,5× každá započatá osmice binárů
19200	20	1,5× registr, 0,8× každá započatá osmice binárů
38400	15	1× registr, 0,5× každá započatá osmice binárů
57600	10	0,7× registr, 0,4× každá započatá osmice binárů

Uvedené hodnoty slouží pro stanovení minimální doby komunikace jednoho komunikačního řádku v tabulce v případě dodržení základních časových poměrů stanovených protokolem MODBUS. V případě, že protistraně trvá zpracování komunikačního požadavku déle, bude celá komunikace zpožděna.

Příklad

Potřebujeme periodicky komunikovat s 10 moduly **DMM-xxx**. U pěti se vyžaduje komunikace 8 registrů a u pěti se vyžaduje komunikace 18 binárů. Pro každý modul tedy bude definován jeden řádek komunikace vzdálených bodů (jedna skupina registrů či binárů). Rychlost komunikace požadujeme 19200 bps. Na základě předchozí tabulky určíme následující doby komunikace:

T_{reg} = 20 + 1,5 × 8 = 32 ms

T_{bin} = 20 + 0,8 × 3 = 22,4 ms

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

Celková minimální doba komunikace tedy je:

T_{celk} = 32 × 5 + 22,4 × 5 = 272 ms

Doplnění
Zcela identická doba by vyšla, kdyby se komunikovalo pouze s jednou Slave stanicí a u této Slave stanice by se dané rozvržení 5 × 8 registrů a 5 × 18 binárů komunikovalo na deseti řádcích definiční tabulky.

4.1.1 Priority komunikace

Automatické priority čtení

DetStudio nabízí pro automatické čtení hodnot ze Slave stanicí 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 komunikovat. Je tedy zřejmé, že v definičních řádcích tabulky dle předchozího případu nelze všude využít komunikaci s prioritou **High**, neboť by rozhraní bylo zcela přetíženo komunikačními požadavky.

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 tento řádek zařazen do fronty komunikačních požadavků.

Pozor!
*Komunikační požadavky se vyřizují postupně, dokud není fronta komunikačních požadavků vyřízená do konce. To znamená, že pokud se v jednom okamžiku označí např. dva řádky s prioritou **High** a dalších 20 s prioritou **Low** a celá tato komunikace trvá např. 600 ms, tak zmíněné dva řádky s prioritou **High** budou opětovně komunikovány **až po** dokončení komunikace všech označených řádků. Nedojde tedy k dodržení zvolené priority **High**.*
*V případě, že musí být udržena perioda komunikace 200 ms pro řádky s prioritou **High** za všech okolností, musí být všechny ostatní komunikace spouštěny manuálně a v takových kvantech, aby doba komunikace daného kvanta komunikačních požadavků a řádků s prioritou **High** nepřekročila 200 ms.*

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, je toto možné 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.

Tyto modulu 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. Lze tímto způsobem 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 rychlosti 19200 bps to bude znamenat dobu:

T = 2 × (20 + 1,5 × 1) = 43 ms

Avšak pokud budou adresy registrů definovány přímo za sebou, např. adresy 0 a 1, bude komunikace probíhat jediným rámcem. Při rychlosti 19200 bps to bude znamenat dobu:

T = 20 + 1,5 × 2 = 23 ms

Vrátíme-li se tedy k „**Doplnění**“ původního příkladu na začátku této kapitoly (**Příklad**), tak v případě, že by všech 5 × 8 registrů a 5 × 18 binárů bylo definováno v nepřerušené řadě za sebou, např. u registrů adresy 0 až 7, 8 až 15, 16 až 23, 24 až 31 a 32 až 39, budou se registry i bináry komunikovat každý jedním rámcem. Při rychlosti 19200 bps to bude znamenat dobu:

T = (20 + 1,5 × 40) + (20 + 0,8 × 15) = 112 ms

Při požadavku na co nejrychlejší komunikaci by se tedy v tomto případě dalo využít automatické priority čtení **`High`**.

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. Toto je signalizováno nastavením bitu č. 4 parametru „Status“ modulu **`MdbmReqSt`** (popis viz kapitola 5.5 „Stavy komunikace“) do stavu True.
3. Po tomto čase se projde tabulka komunikačních požadavků dané Slave stanice. Je-li nějaký nalezen, je proveden pokus o tuto komunikaci. Jako první se prohledávají zápisové požadavky. Současně je bit č. 4 parametru „Status“ modulu **`MdbmReqSt`** po dobu této 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 tomuto 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, ….

4.3 Detekce ztráty spojení u modulů DMM-xxx

Pro případ fyzického přerušení sítě mají všechny výstupní moduly **DMM-xxx** implementován jednoduchý mechanizmus vypnutí výstupů. Pokud modul nezachytí během 10 sekund žádný platný rámec na síti (pro jakýkoliv modul), nastaví na všech výstupech bezpečný stav. Toto chování je na modulech **DMM-xxx** pevně dané a nelze ho změnit.

Bezpečný stav pro různé typy výstupů

Typ výstupů	**Bezpečný stav**
Digitální výstupy	0 V
Reléové výstupy	Rozepnuto
Analogové výstupy	0 V / 0 mA

V případě rozpadu komunikace a nastavení výstupů na bezpečné stavy dojde po obnovení komunikace k nastavení výstupů na požadované hodnoty. Nejdříve však za dobu rovnou periodě komunikace s moduly.

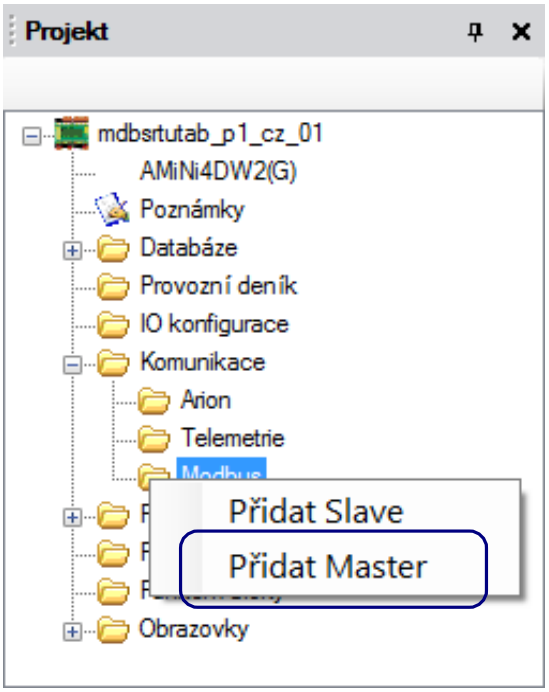
5 Řídicí systém jako Master

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 projektu. Toto se provádí v DetStudiu v okně Projekt v uzlu „Projekt/Komunikace/Modbus“. Po vyvolání kontextového menu nad touto položkou se vybere položka **`Přidat Master`**.



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

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

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

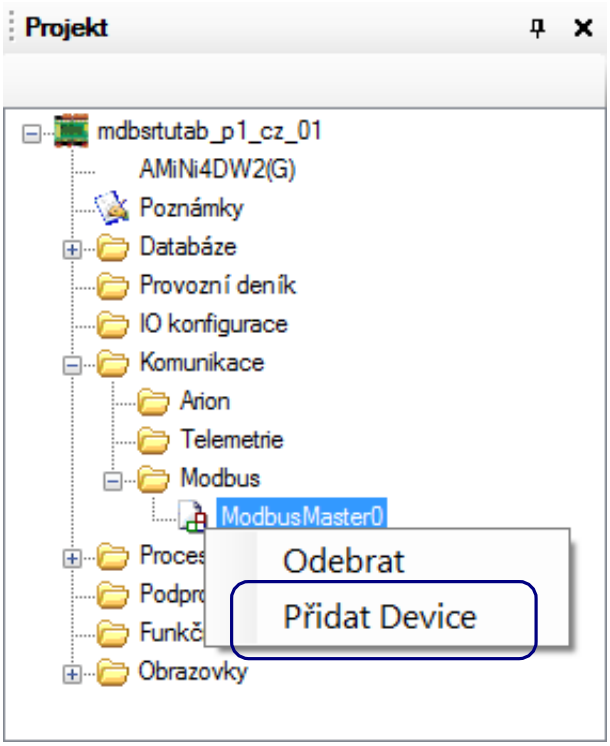
Platí, že pro komunikaci po rozhraní RS232 je potřeba nastavit hodnotu vlastnosti **`SerialPort`** na 0, pro případ rozhraní RS485 je potřeba nastavit hodnotu této vlastnosti na 1.

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 se musí nadefinovat její adresa v síti MODBUS a seznam komunikačních bodů.

Jednotlivé Slave stanice, tzv. *Device*, se v okně Projekt definují přímo v uzlu „Projekt/ Komunikace/Modbus“ konkrétní ModbusMasterX definice. Po vyvolání kontextového menu nad touto položkou se vybere položka **`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

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 firmou AMiT zvolen tento 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 předpokládat komunikaci s **DMM-UI8DO8**, u kterého je požadavek na čtení hodnot analogových vstupů a zápis digitálních výstupů. V návodu na obsluhu k tomuto modulu 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 tohoto tedy 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	DMM1_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	DMM1_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	DMM1_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	DMM1_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 ta stejná hodnota, je možné doporučit následující kód, který provede zápis do komunikační proměnné až v okamžiku, kdy se hodnota pomocné pracovní proměnné změní:

```
If DMM1_DO != DMM1_DO_pr
  Let DMM1_DO_pr = DMM1_DO
EndIf
```

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

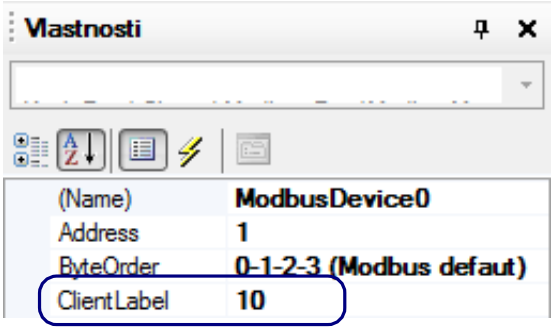
Poznámka
Vlivem vnitřních algoritmů pro použití automatické priority zápisu není vhodné mít na jednom definičním řádku automatickou prioritu čtení i zápisu. Více o tomto lze nalézt v nápovědě DetStudia „PseDet – tvorba regulačních procesů“ v kapitole „Obsah/Komunikace/Modbus/Master – editor tabulky Device“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

5.4 Manuální komunikace

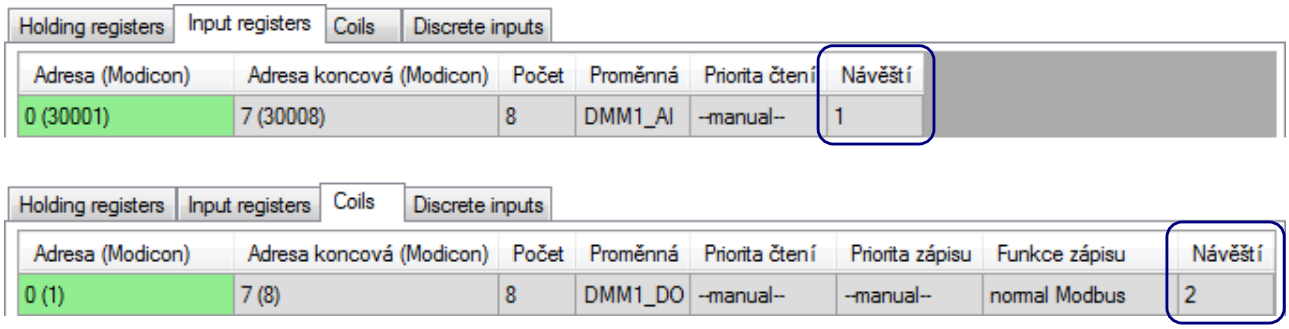
Pro manuální komunikaci je potřeba nadefinovat návěští. Návěští jsou dvojího typu:

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

Platné hodnoty návěští musí být nezáporné. Návěští **`ClientLabel`** musí být unikátní v rámci projektu, 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, DMM1_AI_rslt
         |   |   └ Výsledek provedení modulu
         |   └ Návěští definičního řádku
         └ Návěští ClientLabel
```

nebo:

```
MdbmMark 1, 4, 0, 8, DMM1_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. Tento 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, DMM1_DO_rslt
          |   |  └ Výsledek provedení modulu
          |   └ Návěští definičního řádku
          └ Návěští ClientLabel
```

nebo:

```
MdbmMark 1, 5, 0, 8, DMM1_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 komunikace po daném komunikačním rozhraní
- **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, DMM1_ClSt, DMM1_CS_rslt
          |   |          └ Výsledek provedení modulu
          |   └ Stav klienta, resp. komunikačního rozhraní
          └ Návěští ClientLabel

MdbmReqSt 10, 1, DMM1_RqSt, DMM1_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 parametru stavu komunikačního požadavku získat informaci o případné chybě při komunikaci.

Hodnota tohoto parametru 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 DMM1_RS_rslt.4, DMM1_RS_rslt.1, DMM1_Problem.0
```

5.6 Příklad parametrizace řídicího systému jako Master

Uvažujme aplikaci, ve které jeden řídicí systém AMiNi4W2 má sloužit jako Slave stanice pro druhý řídicí systém AMiNi4W2. 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 se vytvoří 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**.

V definici tabulky **ModbusDevice0** se nadefinuje 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, je v definici **ModbusDevice0** nadefinován parametr **ClientLabel,** např. na hodnotu 10. Pro zjišťování stavu připojení Slave stanice je nadefinováno návěští v řádku registru s adresou 0 a pro manuální spouštění komunikace je nadefinováno 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**. Aby se zamezilo nadbytečným komunikacím registru 100, využije se 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 ap0008_cz_xx.zip. Jedná se o ukázkový projekt s názvem „modbs_p1_cz_xx.dso“ vytvořený ve vývojovém prostředí DetStudio. Projekt je vytvořen pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

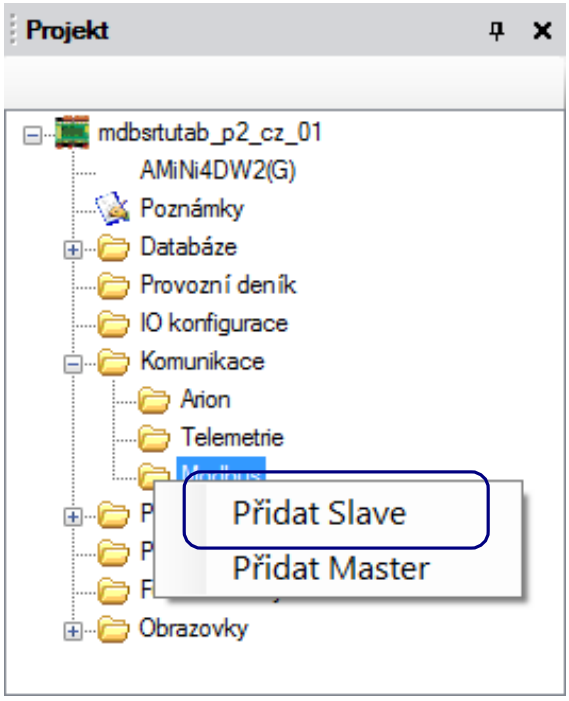
6 Řídicí systém jako Slave

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 projektu. Toto 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. 10 – 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 po rozhraní RS232 je potřeba nastavit hodnotu vlastnosti **`SerialPort`** na 0, pro případ rozhraní RS485 je potřeba nastavit hodnotu této vlastnosti na 1.

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 této 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.

Tuto 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. Musí být alespoň 3.25.
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 AMiNi4W2 pracovat jako modul vzdálených vstupů a výstupů komunikující protokolem MODBUS RTU. Zároveň má být aplikace dostatečně univerzální, aby bylo možné analogové hodnoty komunikovat v desetinné podobě i v celočíselné podobě.

1. Vytvoří se proměnné, dle následující tabulky.

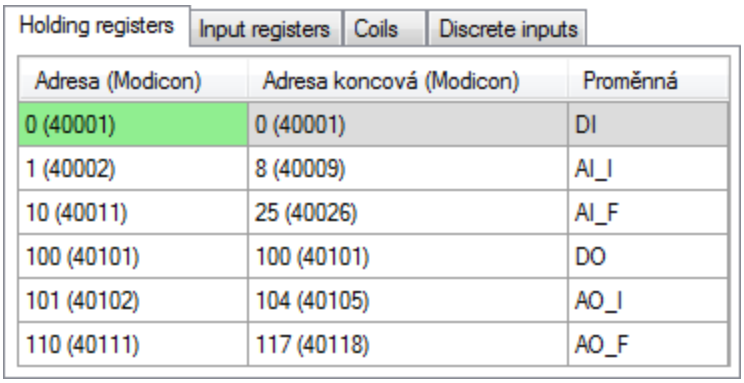
Proměnná	Typ	Komentář
DI	I	Digitální vstupy.
AI_I	MI[1,8]	Integer hodnoty analogových vstupů.
AI_F	MF[1,8]	Float hodnoty analogových vstupů.
DO	I	Digitální výstupy.
AO_I	MI[1,4]	Integer hodnoty analogových výstupů.
AO_F	MF[1,4]	Float hodnoty analogových výstupů.

Proměnná	Typ	Komentář
tempF	F	Pomocná Float proměnná.
Mdbs_Err	I	Kód poslední chyby při komunikaci.

2. Zvolí se 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

3. Na základě předchozí tabulky se nadefinují v uzlu **`Modbus0`** definiční řádky.



Obr. 11 – Definice MODBUS Slave datových bodů a přiřazených proměnných

4. Vytvoření algoritmu v periodickém procesu, 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 ap0008_cz_xx.zip. Jedná se o ukázkový projekt s názvem „modbs_p2_cz_xx.dso“ vytvořený ve vývojovém prostředí DetStudio. Projekt je vytvořen pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

7 Dodatek A

7.1 Kompatibilita s inicializací komunikace pomocí modulů

7.1.1 MODBUS Master

Pro inicializaci MODBUS Master RTU komunikace se dají využívat i moduly **`RmtDef`**, **`RmtAct`** a **`MODBS_R`**, jenž se typicky umisťují do inicializačního a periodického procesu.

Z pohledu interní funkčnosti se však jedná o jinou komunikaci, než je definice tabulkou. Z tohoto důvodu **nelze** kombinovat obě definice komunikace na stejném COM rozhraní.

V případě, že každá inicializace komunikace bude definována na různá COM rozhraní, budou obě komunikace plně funkční.

7.1.2 MODBUS Slave

Pro inicializaci MODBUS Master RTU komunikace se dají využívat i moduly **`MODBS_Var`** a **`MODBS_RSl`**, 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 COM 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 Dodatek B

8.1 Programová obsluha DMM-xxx

Adresa datových bodů v jednotlivých modulech je vždy dána číslem daného vstupu/výstupu modulu **DMM-xxx**.

8.1.1 DMM-DI24

Modul **DMM-DI24** lze provozovat ve dvou režimech (nezávisle na sobě).

- Práce s klasickými digitálními vstupy (hodnoty True/False) – možnost čtení jednoho nebo více vstupů.
- Práce s čítačovými vstupy (čítání impulzů do frekvence 25 Hz) – možnost čtení a zápisu jednoho nebo více čítačů.

Definiční tabulka plné komunikace s modulem může vypadat 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)	23 (40024)	24	M_DI24_cnt[0,0]	Low	--manual--	normal Modbus	2

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Návěští
0 (10001)	23 (10024)	24	M_DI24_DI.0	Normal	1

Obr. 12 – Příklad definice komunikace s DMM-DI24

Dle dané definice se stavy vstupů budou ukládat do bitů 0 až 23 proměnné **`M_DI24_DI`** typu Long každých 1000 ms. Hodnoty čítačů se budou ukládat do buněk proměnné **`M_DI24_cnt`** typu Matice Int (např. dimenze [1×24]) každých 5000 ms.

Pro zjišťování stavu komunikace s modulem se využije modul **`MdbmReqSt`** navázaný na návěští řádku čtení stavu vstupů. Pro použití v příkladu zvolíme hodnotu vlastnosti **`ClientLabel`** 10.

```
MdbmReqSt 10, 1, M_DI24_Stat, NONE
```

Práce s klasickými digitálními vstupy

V algoritmu aplikace lze rovnou využívat komunikační proměnnou **`M_DI24_DI`** pracující s Discrete inputs na adresách 0 až 23.

Práce s čítačovými vstupy

Modul **DMM-DI24** umožňuje využívat funkci čítání příchozích impulzů na kterémkoliv z jeho vstupů. Toto alespoň částečně řeší problémy se zachycováním krátkých impulzů. Hodnoty čítačů jsou dostupné v Holding registrech na adresách 0 až 23.

Při použití této funkce je ovšem nutno brát v potaz jistá omezení:

- Maximální možná načítaná hodnota je 32767 (číslo 15 bit). Po přičtení dalšího pulzu se začíná čítat opět od nuly. Je nutno programově ošetřit přetečení interního čítače.
- Maximální možná frekvence příchozích impulsů může být 25 Hz. Při vyšší frekvenci není možné zaručit, že všechny příchozí impulsy budou zaznamenány.
- Interní čítač modulu je nulován při odpojení napájecího napětí, případně je možno jej nulovat programově.

Na základě definice v tabulce je vyřešeno periodické načítání hodnoty čítačů. V případě, že je žádoucí v průběhu čítání ručně nulovat hodnoty čítačů, lze použít následující algoritmus s využitím modulů pro bezpečný zápis.

```
If @DI24_cntRst and not M_DI24_Stat.4
     Let @DI24_cntRst = false
     MdbmWrBeg 10, 2, NONE
     For i, 0.000, 23.000, 1.000
          Let M_DI24_cnt[0,i] = 0
     EndFor
     MdbmWrFin 10, 2, NONE
EndIf
```

Moduly pro bezpečný zápis jsou v tomto případě jednoznačně potřeba, neboť nelze jednoduše určit přesný okamžik načtení hodnot ze Slave stanice. Pokud by tento okamžik nastal mezi vykonáním řádku s modulem **`EndFor`** a následným modulem **`MdbmWrite`**, byly by do modulu zapsány ty identické hodnoty, které se právě načetly.

Pozor
*U čítačů je nutno ošetřit jejich přetečení! Rozsah čítače je 0 až 32767. Čítač tedy vytváří řadu 0, 1, 2, …, 32766, 32767, 0, 1, 2, … Bude tedy třeba čítat počet přetečení „**p**“ čítače a výsledná suma pulzů pak bude rovna: **suma = p*32768 + čítač**.*

Uvedené algoritmy jsou součástí přílohy ap0008_cz_xx.zip. Jedná se o ukázkový projekt s názvem modbus_p3_cz_xx.dso vytvořený ve vývojovém prostředí DetStudio. Projekt je vytvořen pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

8.1.2 DMM-DO18

Modul **DMM-DO18** lze provozovat ve dvou režimech.

- Práce s klasickými digitálními výstupy (hodnoty True/False) – možnost čtení a zápisu jednoho nebo více výstupů.
- Práce s výstupy v režimu PWM – možnost čtení a zápisu jednoho nebo více parametrů PWM výstupu.

Definiční tabulka plné komunikace s modulem může vypadat 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)	18 (40019)	19	M_DO18_PWM[0,0]	--manual--	Auto	normal Modbus	2

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Priorita zápisu	Funkce zápisu	Návěští
0 (1)	17 (18)	18	M_DO18_DO.0	--manual--	Auto	normal Modbus	1

Obr. 13 – Příklad definice komunikace s DMM-DO18

Dle dané definice se stavy výstupů budou měnit dle bitů 0 až 17 proměnné **`M_DO18_DO`** typu Long po každém zápisu do této proměnné. Parametry PWM výstupů se budou měnit dle buněk proměnné **`M_DO18_PWM`** typu Matice Int (např. dimenze [1×19]) po každém zápisu do této proměnné.

Pro zjišťování stavu komunikace s modulem se využije modul **`MdbmReqSt`** navázaný na návěští řádku zápisu výstupů. Pro použití v příkladu zvolíme hodnotu vlastnosti **`ClientLabel`** 20.

```
MdbmReqSt 20, 1, M_DO18_St_D, NONE
```

nebo

```
MdbmReqSt 20, 2, M_DO18_St_P, NONE
```

v závislosti na tom, která komunikace je častější.

Práce s klasickými digitálními výstupy

V algoritmu aplikace lze rovnou využívat komunikační proměnnou **M_DO18_DO** pracující s Coils na adresách 0 až 17.

Aby nedocházelo k nadbytečným komunikacím stále stejných hodnot, doporučuje se využít kód zmíněný v kapitole 5.3 „Automatická komunikace“, část „Doporučení“.

Práce s výstupy v režimu PWM

V režimu PWM lze měnit jednotlivým výstupům modulu **DMM-DO18** střídu. Tato je dostupná v Holding registrech na adresách 0 až 17. Perioda PWM se zadává na adresu 18 a je společná pro všechny digitální výstupy.

Hodnoty pro střídu jsou interpretovány tak, že **0 až 32767** odpovídá **0 % až 100 %** rozsahu střídy. Hodnota periody PWM je interpretována tak, že **0 až 32767** odpovídá **0 s až 100 s**.

Pro převod hodnot lze použít následující algoritmus s využitím modulů **VarWStat** a **Interpol**.

```
VarWStat DO18_P_per, @DO18_P_p_w, 0
If @DO18_P_p_w
     Interpol DO18_P_per, tmpF2, Params100_32
     Let M_DO18_PWM[0,18] = Int(tmpF2)
EndIf
```

Poznámka
*Reálný výstup na DO modulu **DMM-DO18** je logickým součtem hodnoty DO a PWM.*

Zpětné čtení výstupů a parametrů PWM

Zatímco zápis parametrů je díky prioritě **Auto** automatický po zápisu hodnoty do definované proměnné, čtení má definovánu prioritu **--manual--**. To znamená, že pro neperiodické vyčtení hodnot musí být využito modulů **MdbmRead** nebo **MdbmMark**.

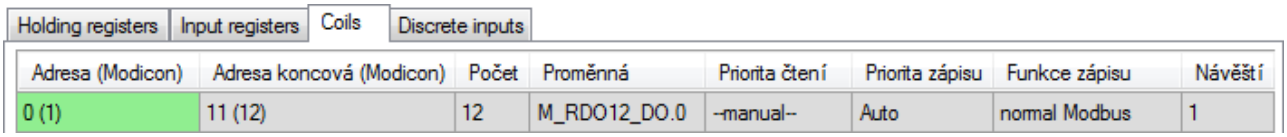
Pozor
*Pokud modul **DMM-DO18** nedetekuje na sběrnici žádný rámec (s jakoukoli adresou) po dobu 10 s, vyhodnotí rozpad komunikace a přejde do bezpečného stavu. V případě, že je na síti se vstupními moduly, doporučujeme volit periodu komunikace se vstupními moduly max. 5 s (automatická priorita **Low**). Je-li na síti pouze s výstupními moduly, je nutno zajistit označení proměnné, zapisované na výstupy, pro komunikaci alespoň jedenkrát za 5 s. Toto lze provést typicky zapsáním libovolné hodnoty (i stejné) do dané proměnné.*

Uvedené algoritmy jsou součástí přílohy ap0008_cz_xx.zip. Jedná se o ukázkový projekt s názvem modbus_p4_cz_xx.dso vytvořený ve vývojovém prostředí DetStudio. Projekt je vytvořen pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

8.1.3 DMM-RDO12

Modul **DMM-RDO12** lze provozovat v režimu práce s klasickými digitálními výstupy (hodnoty True/False) – možnost čtení a zápisu jednoho nebo více výstupů.

Definiční tabulka plné komunikace s modulem může vypadat 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 (1)	11 (12)	12	M_RDO12_DO.0	--manual--	Auto	normal Modbus	1

Obr. 14 – Příklad definice komunikace s DMM-RDO12

Dle dané definice se stavy výstupů budou měnit dle bitů 0 až 11 proměnné **`M_RDO12_DO`** typu Int po každém zápisu do této proměnné.

Pro zjišťování stavu komunikace s modulem se využije modul **`MdbmReqSt`** navázaný na návěští řádku zápisu stavu výstupů. Pro použití v příkladu zvolíme hodnotu vlastnosti **`ClientLabel`** 30.

```
MdbmReqSt 30, 1, M_RDO12_Stat, NONE
```

Práce s digitálními výstupy

V algoritmu aplikace lze rovnou využívat komunikační proměnnou **`M_RDO12_DO`** pracující s Coils na adresách 0 až 11.

Aby nedocházelo k nadbytečným komunikacím stále stejných hodnot, doporučuje se využít kód zmíněn v kapitole 5.3 „Automatická komunikace“, část „Doporučení“.

Pozor
*Pokud modul **DMM-RDO12** nedetekuje na sběrnici žádný rámec (s jakoukoli adresou) po dobu 10 s, vyhodnotí rozpad komunikace a přejde do bezpečného stavu. V případě, že je na síti se vstupními moduly, doporučujeme volit periodu komunikace se vstupními moduly max. 5 s (automatická priorita **`Low`**). Je-li na síti pouze s výstupními moduly, je nutno zajistit označení proměnné, zapisované na výstupy, pro komunikaci alespoň jedenkrát za 5 s. Toto lze provést typicky zapsáním libovolné hodnoty (i stejné) do dané proměnné.*

Zpětné čtení výstupů

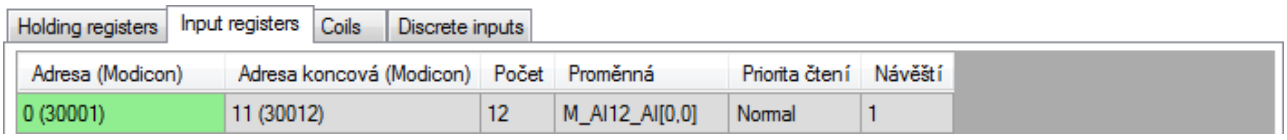
Zatímco zápis parametrů je díky prioritě **`Auto`** automatický po zápisu hodnoty do definované proměnné, čtení má definovánu prioritu **`--manual--`**. To znamená, že pro neperiodické vyčtení hodnot musí být využito modulů **`MdbmRead`** nebo **`MdbmMark`**.

Uvedené algoritmy jsou součástí přílohy ap0008_cz_xx.zip. Jedná se o ukázkový projekt s názvem modbus_p5_cz_xx.dso vytvořený ve vývojovém prostředí DetStudio. Projekt je vytvořen pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

8.1.4 DMM-AI12

Modul **DMM-AI12** umožňuje čtení jednoho nebo více vstupů.

Definiční tabulka plné komunikace s modulem může vypadat dle následujícího obrázku.



Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Návěští
0 (30001)	11 (30012)	12	M_AI12_AI[0,0]	Normal	1

Obr. 15 – Příklad definice komunikace s DMM-AI12

Načítané hodnoty vstupů se budou ukládat do buněk 0 až 11 proměnné **`M_AI12_AI`** typu Matice Int (např. dimenze [1×12]) každých 1000 ms.

Pro zjišťování stavu komunikace s modulem se využije modul **`MdbmReqSt`** navázaný na návěští řádku čtení hodnot vstupů. Pro použití v příkladu zvolíme hodnotu vlastnosti **`ClientLabel`** 40.

```
MdbmReqSt 40, 1, M_AI12_Stat, NONE
```

Práce s analogovými vstupy

Hodnoty vstupů v Input reigstrech na adresách 0 až 11 jsou interpretovány tak, že **0 až 32767** odpovídá **0 % až 100 %** rozsahu vstupu.

Pro převod hodnoty na rozsah 0 V až 5 V lze použít např. následující algoritmus s využitím modulu **`Interpol`**.

```
Let AI12_f[0,0] = M_AI12_AI[0,0]
Interpol AI12_f[0,0], AI12_AI[0,0], Range0_5V
```

Uvedené algoritmy (včetně ukázky převodu měřené hodnoty na 0 V až 10 V, 0 mA až 20 mA a převodu na teplotu měřenou pomocí čidla Ni1000 a Pt1000) jsou součástí přílohy ap0008_cz_xx.zip. Jedná se o ukázkový projekt s názvem modbus_p6_cz_xx.dso vytvořený ve vývojovém prostředí DetStudio. Projekt je vytvořen pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

8.1.5 DMM-AO8x

Modul **DMM-AO8x** umožňuje čtení a zápis jednoho nebo více výstupů a parametrů chování LED.

Definiční tabulka plné komunikace s modulem může vypadat 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)	7 (40008)	8	M_AO8_AO[0,0]	--manual--	Auto	normal Modbus	1
8 (40009)	9 (40010)	2	M_AO8_Leds[0,0]	--manual--	Auto	normal Modbus	

Obr. 16 – Příklad definice komunikace s DMM-AO8x

Dle dané definice se hodnoty výstupů budou měnit dle buněk 0 až 7 proměnné **`M_AO8_AO`** typu Matice Int (např. dimenze [1×8]) po každém zápisu do této proměnné. Parametry chování LED se budou měnit dle buněk proměnné **`M_AO8_Leds`** typu Matice Int (např. dimenze [1×2]) po každém zápisu do této proměnné.

Pro zjišťování stavu komunikace s modulem se využije modul **`MdbmReqSt`** navázaný na návěští řádku zápisu hodnot výstupů. Pro použití v příkladu zvolíme hodnotu vlastnosti **`ClientLabel`** 50.

```
MdbmReqSt 50, 1, M_AO8_Stat, NONE
```

Práce s analogovými výstupy

Hodnoty pro analogové výstupy jsou interpretovány tak, že **0 až 32767** odpovídá **0 % až 100 %** rozsahu analogového výstupu. Tyto jsou dostupné v Holding registrech na adresách 0 až 7.

Pro převod hodnot lze použít následující algoritmus s využitím modulů **`VarWStat`** a **`Interpol`**.

```
VarWStat AO8x_AO, @AO8x_AO_w, 0
If @AO8x_AO_w
     For i, 0.000, 7.000, 1.000
          Let tmpF1 = AO8x_AO[0,i]
          Interpol tmpF1, tmpF2, Rozsah_AO
          Let M_AO8_AO[0,i] = Int(tmpF2)
     EndFor
EndIf
```

Práce s LED

U modulu **DMM-AO8x** lze programově prostřednictvím MODBUS určit chování LED diod na modulu které odpovídají jednotlivým analogovým výstupům. Toto chování lze nastavit pomocí dvou Holding registrů na adresách 8 a 9. Chování LED pak bude následující:

1. Hodnota na výstupech bude větší jak hodnota registru na adrese 9 – LED bliká.
2. Hodnota na výstupech bude větší jak hodnota registru na adrese 8 a menší jak hodnota registru na adrese 9 – LED svítí.
3. Hodnota na výstupech bude menší jak hodnota registru na adrese 8 – LED je zhasnuta.

Druhá podmínka se zpracovává pouze při nesplnění prvé. Není-li splněna ani druhá, příslušná LEDx nesvítí.
Obě meze mohou být stejné a dokonce nulové. Tedy není nutné je zadávat. Pak pro nulový výstup LEDx nesvítí a pro nenulový svítí. Do režimu blikání se při rovnosti mezí LEDx nedostane.
Pokud se zvolí hodnota registru 9 větší než hodnota registru 8 a bude-li hodnota na výstupech menší jak hodnota registru na pozici 8, bude LEDx zhasnuta.
Bude-li hodnota na výstupech větší jak hodnota registru 8 a menší jak hodnota registru 9, LEDx svítí.
Bude-li hodnota na výstupech větší jak hodnota registru na pozici 9, LEDx bliká.

Opět platí, že **0 až 32767** odpovídá **0 % až 100 %** rozsahu analogového výstupu.

Zpětné čtení výstupů a parametrů chování LED

Zatímco zápis hodnot a parametrů je díky prioritě **`Auto`** automatický po zápisu hodnoty do definované proměnné, čtení má definovánu prioritu **`--manual--`**. To znamená, že pro neperiodické vyčtení hodnot musí být využito modulů **`MdbmRead`** nebo **`MdbmMark`**.

Pozor
*Pokud modul **DMM-AO8U** nedetekuje na sběrnici žádný rámec (s jakoukoli adresou) po dobu 10 s, vyhodnotí rozpad komunikace a přejde do bezpečného stavu. V případě, že je na síti se vstupními moduly, doporučujeme volit periodu komunikace se vstupními moduly max. 5 s (automatická priorita **`Low`**). Je-li na síti pouze s výstupními moduly, je nutno zajistit označení proměnné, zapisované na výstupy, pro komunikaci alespoň jedenkrát za 5 s. Toto lze provést typicky zapsáním libovolné hodnoty (i stejné) do dané proměnné.*

Uvedené algoritmy jsou součástí přílohy ap0008_cz_xx.zip. Jedná se o ukázkový projekt s názvem modbus_p7_cz_xx.dso vytvořený ve vývojovém prostředí DetStudio. Projekt je vytvořen pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

8.1.6 DMM-PDO6NI6

Modul **DMM-PDO6NI6** umožňuje čtení jednoho nebo více RTD vstupů. Pro práci s digitálními výstupy jej lze provozovat ve dvou režimech.

- Práce s klasickými digitálními výstupy (hodnoty True/False) – možnost čtení a zápisu jednoho nebo více výstupů.
- Práce s výstupy v režimu PWM – možnost čtení a zápisu jednoho nebo více parametrů PWM výstupu.

Definiční tabulka plné komunikace s modulem může vypadat 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)	6 (40007)	7	M_PDONi_PWM[0,0]	--manual--	Auto	normal Modbus	3

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Návěští
0 (30001)	5 (30006)	6	M_PDONi_Ni[0,0]	Normal	1

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Priorita zápisu	Funkce zápisu	Návěští
0 (1)	5 (6)	6	M_PDONi_DO.0	--manual--	Auto	normal Modbus	2

Obr. 17 – Příklad definice komunikace s DMM-PDO6NI6

Dle dané definice se parametry PWM výstupů budou měnit dle buněk proměnné **`M_PDONi_PWM`** typu Matice Int (např. dimenze [1×7]) po každém zápisu do této proměnné. Stavy výstupů se budou měnit dle bitů 0 až 5 proměnné **`M_PDONi_DO`** typu Int po každém zápisu do této proměnné. Načítané hodnoty RTD vstupů se budou ukládat do buněk 0 až 5 proměnné **`M_PDONi_Ni`** typu Matice Int (např. dimenze [1×6]) každých 1000 ms.

Pro zjišťování stavu komunikace s modulem se využije modul **`MdbmReqSt`** navázaný na návěští řádku čtení hodnot RTD vstupů. Pro použití v příkladu zvolíme hodnotu vlastnosti **`ClientLabel`** 60.

```
MdbmReqSt 60, 1, M_PDONi_Stat, NONE
```

Práce s RTD vstupy

Hodnoty vstupů v Input registrech na adresách 0 až 5 jsou interpretovány tak, že **0 až 32767** odpovídá **0 % až 100 %** rozsahu vstupu.

Pro převod hodnot na rozsah 0 V až 5 V a následně na hodnotu měřené teploty lze použít následující algoritmus s využitím modulu **`Interpol`**.

```
For i, 0.000, 5.000, 1.000
     Let tmpF1 = M_PDONi_Ni[0,i]
     Interpol tmpF1, tmpF2, Range0_5V
     Let PDONi_AI[0,i] = tmpF2
EndFor

Ni1000U2T PDONi_AI[0,0], PDONi_T[0,0], 6180, 15.000, 3920.000
Ni1000U2T PDONi_AI[0,1], PDONi_T[0,1], 6180, 15.000, 3920.000
…
```

Práce s klasickými digitálními výstupy

V algoritmu aplikace lze rovnou využívat komunikační proměnnou **`M_PDONi_DO`** pracující s Coils na adresách 0 až 5.

Aby nedocházelo k nadbytečným komunikacím stále stejných hodnot, doporučuje se využít kód zmíněný v kapitole 5.3 „Automatická komunikace“, část „Doporučení“.

Práce s výstupy v režimu PWM

V režimu PWM lze měnit jednotlivým výstupům střídu. Tato je dostupná v Holding registrech na adresách 0 až 5. Perioda PWM se zadává na adresu 6 a je společná pro všechny digitální výstupy.

Hodnoty pro střídu jsou interpretovány tak, že **0 až 32767** odpovídá **0 % až 100 %** rozsahu střídy. Hodnota periody PWM je interpretována tak, že **0 až 32767** odpovídá **0 s až 100 s**.

Pro převod hodnot lze použít následující algoritmus s využitím modulů **`VarWStat`** a **`Interpol`**.

```
VarWStat PDONi_P_per, @PDONi_P_p_w, 0
If @PDONi_P_p_w
     Interpol PDONi_P_per, tmpF2, Params100_32
     Let M_PDONi_PWM[0,6] = Int(tmpF2)
EndIf
```

Poznámka
*Reálný výstup na DO modulu **DMM-PDO6NI6** je logickým součtem hodnoty DO a PWM.*

Zpětné čtení výstupů a parametrů PWM

Zatímco zápis parametrů je díky prioritě **`Auto`** automatický po zápisu hodnoty do definované proměnné, čtení má definovánu prioritu **`--manual--`**. To znamená, že pro neperiodické vyčtení hodnot musí být využito modulů **`MdbmRead`** nebo **`MdbmMark`**.

Pozor
*Pokud modul **DMM-PDO6NI6** nedetekuje na sběrnici žádný rámec (s jakoukoli adresou) po dobu 10 s, vyhodnotí rozpad komunikace a přejde do bezpečného stavu. V případě, že je na síti se vstupními moduly, doporučujeme volit periodu komunikace se vstupními moduly max. 5 s (automatická priorita **`Low`**). Je-li na síti pouze s výstupními moduly, je nutno zajistit označení proměnné, zapisované na výstupy, pro komunikaci alespoň jedenkrát za 5 s. Toto lze provést typicky zapsáním libovolné hodnoty (i stejné) do dané proměnné.*

Uvedené algoritmy jsou součástí přílohy ap0008_cz_xx.zip. Jedná se o ukázkový projekt s názvem modbus_p8_cz_xx.dso vytvořený ve vývojovém prostředí DetStudio. Projekt je vytvořen pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

8.1.7 DMM-UI8DO8

Modul **DMM-UI8DO8** umožňuje čtení jednoho nebo více analogových vstupů ve formě analogových hodnot a binárních stavů. Digitální výstupy je možno číst a zapisovat po jednom nebo více výstupech.

Definiční tabulka plné komunikace s modulem může vypadat dle následujícího obrázku.

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Návěští
0 (30001)	7 (30008)	8	M_UIDO_UI[0,0]	Normal	1

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	M_UIDO_DO.0	--manual--	Auto	normal Modbus	2

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Návěští
0 (10001)	7 (10008)	8	M_UIDO_DI.0	Low	

Obr. 18 – Příklad definice komunikace s DMM-UI8DO8

Dle dané definice se načítané hodnoty univerzálních vstupů budou ukládat do buněk 0 až 7 proměnné **M_UIDO_UI** typu Matice Int (např. dimenze [1×8]) ve formě analogových hodnot každých 1000 ms. Stavy univerzálních vstupů se budou ukládat do bitů 0 až 7 proměnné **M_UIDO_DI** typu Int ve formě binárních hodnot každých 5000 ms. Stavy výstupů se budou měnit dle bitů 0 až 7 proměnné **M_UIDO_DO** typu Int po každém zápisu do této proměnné.

Pro zjišťování stavu komunikace s modulem se využije modul **MdbmReqSt** navázaný na návěští řádku čtení hodnot analogových vstupů. Pro použití v příkladu zvolíme hodnotu vlastnosti **ClientLabel** 70.

```
MdbmReqSt 70, 1, M_UIDO_Stat, NONE
```

Práce s analogovými vstupy

Hodnoty vstupů v Input registrech na adresách 0 až 7 jsou interpretovány tak, že **0 až 32767** odpovídá **0 % až 100 %** rozsahu vstupu.

Pro převod hodnoty na rozsah 0 V až 5 V lze použít následující algoritmus s využitím modulu **Interpol**.

```
Let UIDO_f[0,0] = M_UIDO_AI[0,0]
Interpol UIDO_f[0,0], UIDO_AI[0,0], Range0_5V
```

Práce s digitálními vstupy

V algoritmu aplikace lze rovnou využívat komunikační proměnnou **M_UIDO_DI** pracující s Discrete inputs na adresách 0 až 7.

Práce s digitálními výstupy

V algoritmu aplikace lze rovnou využívat komunikační proměnnou **M_UIDO_DO** pracující s Coils na adresách 0 až 7.

Aby nedocházelo k nadbytečným komunikacím stále stejných hodnot, doporučuje se využít kód zmíněný v kapitole 5.3 „Automatická komunikace“, část „Doporučení“.

Zpětné čtení výstupů

Zatímco zápis stavů je díky prioritě **`Auto`** automatický po zápisu hodnoty do definované proměnné, čtení má definovánu prioritu **`--manual--`**. To znamená, že pro neperiodické vyčtení hodnot musí být využito modulů **`MdbmRead`** nebo **`MdbmMark`**.

Pozor
*Pokud modul **DMM-UI8DO8** nedetekuje na sběrnici žádný rámec (s jakoukoli adresou) po dobu 10 s, vyhodnotí rozpad komunikace a přejde do bezpečného stavu. V případě, že je na síti se vstupními moduly, doporučujeme volit periodu komunikace se vstupními moduly max. 5 s (automatická priorita **`Low`**). Je-li na síti pouze s výstupními moduly, je nutno zajistit označení proměnné, zapisované na výstupy, pro komunikaci alespoň jedenkrát za 5 s. Toto lze provést typicky zapsáním libovolné hodnoty (i stejné) do dané proměnné.*

Uvedené algoritmy jsou součástí přílohy ap0008_cz_xx.zip. Jedná se o ukázkový projekt s názvem modbus_p9_cz_xx.dso vytvořený ve vývojovém prostředí DetStudio. Projekt je vytvořen pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

8.1.8 DMM-UI8RDO8

Modul **DMM-UI8RDO8** umožňuje čtení jednoho nebo více analogových vstupů ve formě analogových hodnot a binárních stavů. Digitální výstupy je možno číst a zapisovat po jednom nebo více výstupech.

Definiční tabulka plné komunikace s modulem může vypadat dle následujícího obrázku.

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Návěští
0 (30001)	7 (30008)	8	M_UIRDO_UI[0,0]	Normal	1

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	M_UIRDO_DO.0	--manual--	Auto	normal Modbus	2

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Návěští
0 (10001)	7 (10008)	8	M_UIRDO_DI.0	Low	

Obr. 19 – Příklad definice komunikace s DMM-UI8RDO8

Dle dané definice se načítané hodnoty univerzálních vstupů budou ukládat do buněk 0 až 7 proměnné **`M_UIRDO_UI`** typu Matice Int (např. dimenze [1×8]) ve formě analogových hodnot každých 1000 ms. Stavy univerzálních vstupů se budou ukládat do bitů 0 až 7 proměnné **`M_UIRDO_DI`** typu Int ve formě binárních hodnot každých 5000 ms. Stavy výstupů se budou měnit dle bitů 0 až 7 proměnné **`M_UIRDO_DO`** typu Int po každém zápisu do této proměnné.

Pro zjišťování stavu komunikace s modulem se využije modul **`MdbmReqSt`** navázaný na návěští řádku čtení hodnot analogových vstupů. Pro použití v příkladu zvolíme hodnotu vlastnosti **`ClientLabel`** 80.

```
MdbmReqSt 80, 1, M_UIRDO_Stat, NONE
```

Práce s analogovými vstupy

Hodnoty vstupů v Input registrech na adresách 0 až 7 jsou interpretovány tak, že **0 až 32767** odpovídá **0 % až 100 %** rozsahu vstupu.

Pro převod hodnoty na rozsah 0 V až 5 V lze použít následující algoritmus s využitím modulu **`Interpol`**.

```
Let UIRDO_f[0,0] = M_UIRDO_AI[0,0]
Interpol UIRDO_f[0,0], UIRDO_AI[0,0], Range0_5V
```

Práce s digitálními vstupy

V algoritmu aplikace lze rovnou využívat komunikační proměnnou **`M_UIRDO_DI`** pracující s Discrete inputs na adresách 0 až 7.

Práce s digitálními výstupy

V algoritmu aplikace lze rovnou využívat komunikační proměnnou **`M_UIRDO_DO`** pracující s Coils na adresách 0 až 7.

Aby nedocházelo k nadbytečným komunikacím stále stejných hodnot, doporučuje se využít kód zmíněný v kapitole 5.3 „Automatická komunikace“, část „Doporučení“.

Zpětné čtení výstupů

Zatímco zápis stavů je díky prioritě **`Auto`** automatický po zápisu hodnoty do definované proměnné, čtení má definovánu prioritu **`--manual--`**. To znamená, že pro neperiodické vyčtení hodnot musí být využito modulů **`MdbmRead`** nebo **`MdbmMark`**.

Pozor
*Pokud modul **DMM-UI8RDO8** nedetekuje na sběrnici žádný rámec (s jakoukoli adresou) po dobu 10 s, vyhodnotí rozpad komunikace a přejde do bezpečného stavu. V případě, že je na síti se vstupními moduly, doporučujeme volit periodu komunikace se vstupními moduly max. 5 s (automatická priorita **`Low`**). Je-li na síti pouze s výstupními moduly, je nutno zajistit označení proměnné, zapisované na výstupy, pro komunikaci alespoň jedenkrát za 5 s. Toto lze provést typicky zapsáním libovolné hodnoty (i stejné) do dané proměnné.*

Uvedené algoritmy jsou součástí přílohy ap0008_cz_xx.zip. Jedná se o ukázkový projekt s názvem modbus_p10_cz_xx.dso vytvořený ve vývojovém prostředí DetStudio. Projekt je vytvořen pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

8.1.9 DMM-UI8AO8

Modul **DMM-UI8AO8** umožňuje čtení jednoho nebo více analogových vstupů ve formě analogových hodnot a binárních stavů. Analogové výstupy je možno číst a zapisovat po jednom nebo více výstupech.

Definiční tabulka plné komunikace s modulem může vypadat 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)	7 (40008)	8	M_UIAO_AO[0,0]	--manual--	Auto	normal Modbus	2
8 (40009)	9 (40010)	2	M_UIAO_Leds[0,0]	--manual--	Auto	normal Modbus	

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Návěští
0 (30001)	7 (30008)	8	M_UIAO_UI[0,0]	Normal	1

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Návěští
0 (10001)	7 (10008)	8	M_UIAO_DI.0	Low	

Obr. 20 – Příklad definice komunikace s DMM-UI8AO8

Dle dané definice se načítané hodnoty univerzálních vstupů budou ukládat do buněk 0 až 7 proměnné **`M_UIAO_UI`** typu Matice Int (např. dimenze [1×8]) ve formě analogových hodnot každých 1000 ms. Stavy univerzálních vstupů se budou ukládat do bitů 0 až 7 proměnné **`M_UIAO_DI`** typu Int ve formě binárních hodnot každých 5000 ms. Hodnoty výstupů se budou ukládat dle buněk 0 až 7 proměnné **`M_UIAO_AO`** typu Matice Int (např. dimenze [1×8]) po každém zápisu do této proměnné. Parametry chování LED se budou ukládat dle buněk proměnné **`M_UIAO_Leds`** typu Matice Int (např. dimenze [1×2]) po každém zápisu do této proměnné.

Pro zjišťování stavu komunikace s modulem se využije modul **`MdbmReqSt`** navázaný na návěští řádku čtení hodnot analogových vstupů. Pro použití v příkladu zvolíme hodnotu vlastnosti **`ClientLabel`** 90.

```
MdbmReqSt 90, 1, M_UIAO_Stat, NONE
```

Práce s analogovými vstupy

Hodnoty vstupů v Input registrech na adresách 0 až 7 jsou interpretovány tak, že **0 až 32767** odpovídá **0 % až 100 %** rozsahu vstupu.

Pro převod hodnoty na rozsah 0 V až 5 V lze použít např. následující algoritmus s využitím modulu **`Interpol`**.

```
Let UIAO_f[0,0] = M_UIAO_AI[0,0]
Interpol UIAO_f[0,0], UIAO_AI[0,0], Range0_5V
```

Práce s digitálními vstupy

V algoritmu aplikace lze rovnou využívat komunikační proměnnou **`M_UIAO_DI`** pracující s Discrete inputs na adresách 0 až 7.

Práce s analogovými výstupy

Hodnoty pro analogové výstupy jsou interpretovány tak, že **0 až 32767** odpovídá **0 % až 100 %** rozsahu analogového výstupu. Tyto jsou dostupné v Holding registrech na adresách 0 až 7.

Pro převod hodnot lze použít např. následující algoritmus s využitím modulů **VarWStat** a **Interpol**.

```
VarWStat UIAO_AO, @UIAO_AO_w, 0
If @UIAO_AO_w
     For i, 0.000, 7.000, 1.000
          Let tmpF1 = UIAO_AO[0,i]
          Interpol tmpF1, tmpF2, Rozsah_AO
          Let M_UIAO_AO[0,i] = Int(tmpF2)
     EndFor
EndIf
```

Práce s LED

U modulu **DMM-UI8AO8U** lze programově prostřednictvím MODBUS určit chování LED diod na modulu které odpovídají jednotlivým analogovým výstupům. Toto chování lze nastavit pomocí dvou Holding registrů na adresách 8 a 9. Chování LED pak bude následující:

4. Hodnota na výstupech bude větší jak hodnota registru na adrese 9 – LED bliká.
5. Hodnota na výstupech bude větší jak hodnota registru na adrese 8 a menší jak hodnota registru na adrese 9 – LED svítí.
6. Hodnota na výstupech bude menší jak hodnota registru na adrese 8 – LED je zhasnuta.

Druhá podmínka se zpracovává pouze při nesplnění prvé. Není-li splněna ani druhá, příslušná LEDx nesvítí.
Obě meze mohou být stejné a dokonce nulové. Tedy není nutné je zadávat. Pak pro nulový výstup LEDx nesvítí a pro nenulový svítí. Do režimu blikání se při rovnosti mezí LEDx nedostane.
Pokud se zvolí hodnota registru 9 větší než hodnota registru 8 a bude-li hodnota na výstupech menší jak hodnota registru na pozici 8, bude LEDx zhasnuta.
Bude-li hodnota na výstupech větší jak hodnota registru 8 a menší jak hodnota registru 9, LEDx svítí.
Bude-li hodnota na výstupech větší jak hodnota registru na pozici 9, LEDx bliká.

Opět platí, že **0 až 32767** odpovídá **0 % až 100 %** rozsahu analogového výstupu.

Zpětné čtení výstupů a parametrů chování LED

Zatímco zápis hodnot a parametrů je díky prioritě **Auto** automatický po zápisu hodnoty do definované proměnné, čtení má definovánu prioritu **--manual--**. To znamená, že pro neperiodické vyčtení hodnot musí být využito modulů **MdbmRead** nebo **MdbmMark**.

Pozor
*Pokud modul **DMM-UI8AO8U** nedetekuje na sběrnici žádný rámec (s jakoukoli adresou) po dobu 10 s, vyhodnotí rozpad komunikace a přejde do bezpečného stavu. V případě, že je na síti se vstupními moduly, doporučujeme volit periodu komunikace se vstupními moduly max. 5 s (automatická priorita **Low**). Je-li na síti pouze s výstupními moduly, je nutno zajistit označení proměnné, zapisované na výstupy, pro komunikaci alespoň jedenkrát za 5 s. Toto lze provést typicky zapsáním libovolné hodnoty (i stejné) do dané proměnné.*

Uvedené algoritmy jsou součástí přílohy ap0008_cz_xx.zip. Jedná se o ukázkový projekt s názvem modbus_p11_cz_xx.dso vytvořený ve vývojovém prostředí DetStudio. Projekt je vytvořen pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

9 Dodatek C

9.1 Programová obsluha AMR-OPxx

9.1.1 AMR-OP7x(RH) / AMR-OP6x / AMR-OP4x / AMR-OP3xA(RH)

Nástěnné ovladače **AMR-OP7x(RH)**, **AMR-OP6x**, **AMR-OP4x** a **AMR-OP3xA(RH)** umožňují z výroby nebo po nahrání příslušné typové aplikace čtení nebo zápis jednoho nebo více hodnot Holding registrů ve formě analogových hodnot a binárních stavů.

Definiční tabulka plné komunikace s nástěnným ovladačem může vypadat dle následujícího obrázku.

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Priorita zápisu	Funkce zápisu	Návěští
2 (40003)	3 (40004)	2	OP7x3x_Time	--manual--	Auto	normal Modbus	
100 (40101)	100 (40101)	1	OP7x3x_Set	--manual--	Auto	normal Modbus	
101 (40102)	101 (40102)	1	OP7x3x_Reset	--manual--	Auto	normal Modbus	
102 (40103)	103 (40104)	2	OP7x3x_Stats	Normal	--manual--	normal Modbus	1
104 (40105)	105 (40106)	2	OP7x3x_Corr	--manual--	--manual--	normal Modbus	2
106 (40107)	107 (40108)	2	OP7x3x_SetPt	--manual--	Auto	normal Modbus	
108 (40109)	111 (40112)	4	OP7x3x_Vals[0,0]	--manual--	--manual--	normal Modbus	3
112 (40113)	113 (40114)	2	OP7x3x_LED	--manual--	Auto	normal Modbus	

Obr. 21 – Příklad definice komunikace s AMR-OP7x(RH)/AMR-OP6x/ AMR-OP4x/AMR-OP3xA(RH)

Bližší popis jednotlivých registrů je možné nalézt v dokumentaci daných nástěnných ovladačů či popisu typové aplikace.

Pro zjišťování stavu komunikace s nástěnným ovladačem se využije modul **`MdbmReqSt`** navázaný na návěští řádku čtení dvojregistru 102-103. Pro použití v příkladu zvolíme hodnotu vlastnosti **`ClientLabel`**10.

```
MdbmReqSt 10, 1, OP7x3x_ReqSt, NONE
```

Zpracování stavu po restartu ovladače nebo rozpadu komunikace

V případě, že dojde k restartu ovladače nebo k rozpadu komunikace, je hodnota dvojregistru 102-103 nastavena na hodnotu **`0xFF`**. Očekává se zápis takové kombinace bitů do registrů 100 a 101, aby byl v ovladači platný režim místnosti a ventilátoru.

```
Let OP7x3x_Reset = 0xFF
Let OP7x3x_Set = (FanMode << 4) | (RoomMode << 1)
```

Dále je doporučeno do ovladače zapsat přechozí hodnoty korekce, žádané teploty místnosti a jasu LED a nový čas pro šetřič obrazovky.

```
MdbmWrite 10, 2, NONE
Let OP7x3x_SetPt = OP7x3x_SetPt
Let OP7x3x_LED = OP7x3x_LED
GetTime OP7x3x_Time, NONE, NONE
```

Načtení nových hodnot z ovladače

Nástěnné ovladače využívají bit 0 dvojregistru 102-103 pro signalizaci změny hodnoty ze strany ovladače. Tento bit lze využít v podmínce a při jejím splnění tento bit vynulovat a při běhu procesu vyčíst nové hodnoty z ovladače.

```
If OP7x3x_Stats.0
     Let OP7x3x_Reset = 0b1
     Let OP7x3x_Set = 0
     Let @OP3x7x_read = true
Else
     If @OP3x7x_read
          Let @OP3x7x_read = false
          MdbmRead 10, 2, NONE
          MdbmRead 10, 3, NONE
          Let FanMode = Int((OP7x3x_Stats & 0b1110000) >> 4)
          Let RoomMode = Int((OP7x3x_Stats & 0b110) >> 1)
     EndIf
EndIf
```

Zápis vlastních hodnot do ovladače

V případě, že ovladač není ve stavu po restartu či výpadku komunikace nebo nedochází k čtení nových hodnot z ovladače, je možné do ovladače zapsat vlastní hodnoty.

Pro zápis režimu místnosti a ventilátoru lze využít porovnání poslední vyčtené hodnoty dvojregistru 102-103 s aktuálními hodnotami v proměnných.

```
If (FanMode != Int((OP7x3x_Stats & 0b1110000) >> 4)) or (RoomMode != Int((OP7x3x_Stats
& 0b110) >> 1))
     Let OP7x3x_Set = (FanMode << 4 ) | (RoomMode << 1)
     Let OP7x3x_Reset = 0b1110110 - (FanMode << 4) - (RoomMode << 1)
EndIf
```

Zápis vlastní hodnoty korekce je nutné provést pomocí modulů tzv. bezpečného zápisu.

```
If @OP3x7x_w_cr
     Let @OP3x7x_w_cr = false
     MdbmWrBeg 10, 2, NONE
     Let OP7x3x_Corr = NewCorr
     MdbmWrFin 10, 2, NONE
EndIf
```

Zápis žádané hodnoty teploty místnosti lze provést díky prioritě zápisu **Auto** kdykoliv.

Uvedené algoritmy jsou součástí přílohy ap0008_cz_xx.zip. Jedná se o ukázkový projekt s názvem modbus_p12_cz_xx.dso vytvořený ve vývojovém prostředí DetStudio. Projekt je vytvořen pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

9.1.2 AMR-OP7xC

Oproti nástěnným ovladačům **AMR-OP7x(RH)**, **AMR-OP6x**, **AMR-OP4x** a **AMR-OP3xA(RH)** se rozložení komunikačních Holding registrů liší v přítomnosti registrů pro práci s CO_2.

Definiční tabulka plné komunikace s nástěnným ovladačem může vypadat dle následujícího obrázku.

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Priorita zápisu	Funkce zápisu	Návěští
2 (40003)	3 (40004)	2	OP7xC_Time	--manual--	Auto	normal Modbus	
100 (40101)	100 (40101)	1	OP7xC_Set	--manual--	Auto	normal Modbus	
101 (40102)	101 (40102)	1	OP7xC_Reset	--manual--	Auto	normal Modbus	
102 (40103)	103 (40104)	2	OP7xC_Status	Normal	--manual--	normal Modbus	1
104 (40105)	105 (40106)	2	OP7xC_Corr	--manual--	--manual--	normal Modbus	2
106 (40107)	107 (40108)	2	OP7xC_SetPnt	--manual--	Auto	normal Modbus	
108 (40109)	111 (40112)	4	OP7xC_Values[0,0]	--manual--	--manual--	normal Modbus	3
114 (40115)	114 (40115)	1	OP7xC_LmtCO2	--manual--	Auto	normal Modbus	
115 (40116)	115 (40116)	1	OP7xC_MerCO2	--manual--	--manual--	normal Modbus	4

Obr. 22 – Příklad definice komunikace s AMR-OP7xC

Bližší popis jednotlivých registrů je možné nalézt v dokumentaci daného nástěnného ovladače.

Pro zjišťování stavu komunikace s nástěnným ovladačem se využije modul **`MdbmReqSt`** navázaný na návěští řádku čtení dvojregistru 102-103. Pro použití v příkladu zvolíme hodnotu vlastnosti **`ClientLabel`** 20.

```
MdbmReqSt 20, 1, OP7xC_ReqSt, NONE
```

Zpracování stavu po restartu ovladače nebo rozpadu komunikace

Způsob zpracování odpovídá přechozímu případu z kapitoly 9.1.1 „AMR-OP7x(RH) / AMR-OP6x / AMR-OP4x / AMR-OP3xA(RH)“, část „Zpracování stavu po restartu ovladače nebo rozpadu komunikace“.

Jediným rozdílem je zápis limitní hodnoty koncentrace CO_2 namísto hodnoty jasu LED.

```
Let OP7xC_LmtCO2 = OP7xC_LmtCO2
```

Načtení nových hodnot z ovladače

Způsob zpracování odpovídá přechozímu případu z kapitoly 9.1.1 „AMR-OP7x(RH) / AMR-OP6x / AMR-OP4x / AMR-OP3xA(RH)“, část „Načtení nových hodnot z ovladače“.

Jediným rozdílem je doplnění vyčtení měřené hodnoty CO_2.

```
MdbmRead 20, 4, NONE
```

Zápis vlastních hodnot do ovladače

Způsob zpracování odpovídá přechozímu případu z kapitoly 9.1.1 „AMR-OP7x(RH) / AMR-OP6x / AMR-OP4x / AMR-OP3xA(RH)“, část „Zápis vlastních hodnot do ovladače“.

Jediným rozdílem je možnost zápisu limitní hodnoty koncentrace CO_2 v libovolný okamžik díky prioritě zápisu **`Auto`** u tohoto registru.

Uvedené algoritmy jsou součástí přílohy ap0008_cz_xx.zip. Jedná se o ukázkový projekt s názvem modbus_p13_cz_xx.dso vytvořený ve vývojovém prostředí DetStudio. Projekt je vytvořen pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

9.1.3 AMR-OP7xRHC

Oproti nástěnnému ovladači **AMR-OP7xC** se rozložení komunikačních Holding registrů liší v přeskládání pořadí registrů pro práci s CO_2.

Definiční tabulka plné komunikace s nástěnným ovladačem může vypadat dle následujícího obrázku.

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Priorita zápisu	Funkce zápisu	Návěští
2 (40003)	3 (40004)	2	OP7RHC_Time	--manual--	Auto	normal Modbus	
100 (40101)	100 (40101)	1	OP7RHC_Set	--manual--	Auto	normal Modbus	
101 (40102)	101 (40102)	1	OP7RHC_Reset	--manual--	Auto	normal Modbus	
102 (40103)	103 (40104)	2	OP7RHC_Stat	Normal	--manual--	normal Modbus	1
104 (40105)	105 (40106)	2	OP7RHC_Corr	--manual--	--manual--	normal Modbus	2
106 (40107)	107 (40108)	2	OP7RHC_SetPt	--manual--	Auto	normal Modbus	
108 (40109)	113 (40114)	6	OP7RHC_Vals[0,0]	--manual--	--manual--	normal Modbus	3
116 (40117)	116 (40117)	1	OP7RHC_LmtH2	--manual--	Auto	normal Modbus	

Obr. 23 – Příklad definice komunikace s AMR-OP7xRHC

Bližší popis jednotlivých registrů je možné nalézt v dokumentaci daného nástěnného ovladače.

Pro zjišťování stavu komunikace s nástěnným ovladačem se využije modul **`MdbmReqSt`** navázaný na návěští řádku čtení dvojregistru 102-103. Pro použití v příkladu zvolíme hodnotu vlastnosti **`ClientLabel`** 30.

```
MdbmReqSt 30, 1, OP7RHC_ReqSt, NONE
```

Zpracování stavu po restartu ovladače nebo rozpadu komunikace

Způsob zpracování odpovídá přechozímu případu z kapitoly 9.1.1 „AMR-OP7x(RH) / AMR-OP6x / AMR-OP4x / AMR-OP3xA(RH)“, část „Zpracování stavu po restartu ovladače nebo rozpadu komunikace“.

Jediným rozdílem je zápis limitní hodnoty koncentrace CO_2 namísto hodnoty jasu LED.

```
Let OP7RHC_LmtH2 = OP7RHC_LmtH2
```

Načtení nových hodnot z ovladače

Způsob zpracování odpovídá přechozímu případu z kapitoly 9.1.1 „AMR-OP7x(RH) / AMR-OP6x / AMR-OP4x / AMR-OP3xA(RH)“, část „Načtení nových hodnot z ovladače“.

Jediným rozdílem je, že maticová proměnná **`OP7RHC_Vals`** obsahuje i vyčtenou hodnotu měřené koncentrace CO_2.

Zápis vlastních hodnot do ovladače

Způsob zpracování odpovídá přechozímu případu z kapitoly 9.1.1 „AMR-OP7x(RH) / AMR-OP6x / AMR-OP4x / AMR-OP3xA(RH)“, část „Zápis vlastních hodnot do ovladače“.

Jediným rozdílem je možnost zápisu limitní hodnoty koncentrace CO_2 v libovolný okamžik díky prioritě zápisu **`Auto`** u tohoto registru.

Uvedené algoritmy jsou součástí přílohy ap0008_cz_xx.zip. Jedná se o ukázkový projekt s názvem modbus_p14_cz_xx.dso vytvořený ve vývojovém prostředí DetStudio. Projekt je vytvořen pro řídicí

systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

9.1.4 AMR-OP40(RH)C

Oproti dříve zmíněným nástěnným ovladačům se rozložení komunikačních Holding registrů liší v nevyužívání stavových registrů.

Definiční tabulka plné komunikace s nástěnným ovladačem může vypadat dle následujícího obrázku.

Holding registers | Input registers | Coils | Discrete inputs

Adresa (Modicon)	Adresa koncová (Modicon)	Počet	Proměnná	Priorita čtení	Priorita zápisu	Funkce zápisu	Návěští
108 (40109)	113 (40114)	6	OP40C_Values[0,0]	Low	--manual--	normal Modbus	1
115 (40116)	115 (40116)	1	OP40C_LED	--manual--	Auto	normal Modbus	
116 (40117)	116 (40117)	1	OP40C_LmtH2	--manual--	Auto	normal Modbus	
117 (40118)	117 (40118)	1	OP40C_LmtH1	--manual--	Auto	normal Modbus	

Obr. 24 – Příklad definice komunikace s AMR-OP40(RH)C

Bližší popis jednotlivých registrů je možné nalézt v dokumentaci daného nástěnného ovladače.

Pro zjišťování stavu komunikace s nástěnným ovladačem se využije modul **`MdbmReqSt`** navázaný na návěští řádku čtení registrů 108 až 113. Pro použití v příkladu zvolíme hodnotu vlastnosti **`ClientLabel`** 40.

```
MdbmReqSt 40, 1, OP40C_ReqSt, NONE
```

Zpracování stavu po restartu ovladače nebo rozpadu komunikace

Nástěnný ovladač nesignalizuje restart nebo rozpad komunikace. K tomuto účelu lze tedy využít proměnnou se stavem komunikace. K zápisu jasu LED a limitních hodnot koncentrace CO_2 tedy dojde, když byl detekován rozpad komunikace a komunikace je již opět funkční.

```
If OP40C_ReqSt.4
     Let @OP40C_write = true
EndIf

If @OP40C_write and OP40C_ReqSt.1
     Let @OP40C_write = false
     Let OP40C_LED = OP40C_LED
     Let OP40C_LmtH1 = OP40C_LmtH1
     Let OP40C_LmtH2 = OP40C_LmtH2
EndIf
```

Načtení nových hodnot z ovladače

Hodnoty z ovladače se načítají periodicky s prioritou **`Low`**, tedy co 5000 ms.

Zápis vlastních hodnot do ovladače

Hodnoty lze zapisovat kdykoliv díky prioritě zápisu **`Auto`** u daných registrů.

Uvedené algoritmy jsou součástí přílohy ap0008_cz_xx.zip. Jedná se o ukázkový projekt s názvem modbus_p15_cz_xx.dso vytvořený ve vývojovém prostředí DetStudio. Projekt je vytvořen pro řídicí systém **AMiNi4DW2**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

10 Technická podpora

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

11 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.