Komunikace v síti MODBUS RTU (Gen2)

Abstrakt

Aplikační poznámka popisuje použití protokolu MODBUS RTU v řídicích systémech Gen2.

Autor: Petr Latina, Zbyněk Říha
Dokument: ap0056_ap_cz_002.pdf

Příloha

Obsah souboru: ap0056_ap_cz_002.zip

modbs_p1_cz_100.dsox	Příklad parametrizace řídicího systému jako Master stanice.
modbs_p2_cz_100.dsox	Příklad parametrizace řídicího systému jako Slave stanice.
modbs_p3_cz_100.dsox	Programová obsluha **AMR-OP7xRH / AMR-OP60RH / AMR-OP4x / AMR-OP3xARH**.
modbs_p4_cz_100.dsox	Programová obsluha **AMR-OP7xRHC**.
modbs_p5_cz_100.dsox	Programová obsluha **AMR-OP4xRHC**.

Copyright (c) 2022, AMiT®, spol. s r.o.
amitomation.cz

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
4.2 Priorita komunikace 8
Automatická priorita komunikace 8
Manuální priorita komunikace 9
4.3 Komunikace při rozpadu spojení 9
4.4 Detekce ztráty spojení u modulů AMRIO-xxx a DMM-xxx 9
5 Řídicí systém jako Master stanice 11
5.1 Komunikační definice 11
5.2 Definice Slave stanice a datových bodů pro komunikaci 12
5.2.1 Objekty pro obecnou komunikaci se Slave stanicí 14
5.2.2 Objekty pro konkrétní modul z produkce firmy AMiT 18
5.3 Stavy komunikace 18
5.4 Příklad parametrizace řídicího systému jako Master stanice 19
6 Řídicí systém jako Slave stanice 21
6.1 Komunikační definice 21
6.2 Definice datových bodů pro komunikaci 22
6.3 Stavy komunikace 22
6.4 Příklad parametrizace řídicího systému jako Slave stanice 22
7 Dodatek A 25
7.1 Programová obsluha AMR-OPxx 25
7.1.1 AMR-OP7x(RH) / AMR-OP60RH / AMR-OP4x / AMR-OP3xA(RH) 25
Zpracování stavu po restartu ovladače nebo rozpadu komunikace 25
Načtení nových hodnot z ovladače 25
Zápis vlastních hodnot do ovladače 26
7.1.2 AMR-OP7xRHC 26
7.1.3 AMR-OP40RHC 27
8 Technická podpora 28
9 Upozornění 29

Historie revizí

Verze	Datum	Autor změny	Změny
001	22. 08. 2016	Latina P., Říha Z.	Nový dokument.
002	18. 02. 2022	Novotný M.	Celková revize obsahu dokumentu.

Související dokumentace

1. Nápověda k části EsiDet vývojového prostředí DetStudio
   soubor: EsiDet_cs.chm
2. Návod na obsluhu k **AMRIO-xxx**
   soubor: amrio-xxx_g_cz_xxx.pdf
3. Návod na obsluhu k **AMR-OPxx**
   soubor: amr-opxx_g_cz_xxx.pdf
4. Aplikační poznámka AP0016 – Zásady používání RS485
   soubor: ap0016_cz_xx.pdf

Určení aplikační poznámky

Aplikační poznámka je určena řídicím systémům Generace 2 – řídicím systémům s operačním systémem FreeRTOS programovaných v DetStudiu v editoru EsiDet. Více informací o generacích systémů lze nalézt na webu amitomation.cz.

1 Definice použitých pojmů

Řídicí systém
Jedná se o řídicí systémy a terminály Generace 2, programovatelné regulátory a komunikační jednotky firmy AMiT, u kterých se algoritmy procesů programují v tzv. EsiDet části prostředí DetStudio. Např. **AMR-CP40/DM**, **AMR-OP87 RevA** nebo **AMiNi5D**.

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

Slave stanice
Jedná se o stanici s unikátní adresou, která pasivně naslouchá na komunikačním rozhraní a po příjmu rámce od Master stanice na něj odpoví. Na jednom komunikačním rozhraní může být pouze jediná Slave stanice. 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 uchovávacího) nebo bináru (vstupního či výstupního), který typicky reprezentuje vstup nebo výstup na Slave stanici. Každému datovému bodu se může přiřadit (maticová) proměnná či bit, do kterého se budou zapisovat načtené hodnoty, nebo ze kterých se budou brát hodnoty pro zápis do Slave stanice.

Moduly AMRIO-xxx a DMM-xxx
Moduly **AMRIO-xxx** s aplikací z výroby a moduly **DMM-xxx** umožňují prostřednictvím protokolu MODBUS RTU rozšířit počet vstupů a výstupů u zařízení, které je naprogramováno jako MODBUS Master stanice. Do jedné sítě MODBUS lze připojit až 63 modulů.

Nástěnné ovladače AMR-OPxx
Nástěnné ovladače mohou na rozhraní RS485 naslouchat jako MODBUS 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 protokolem MODBUS 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 RTU v rozšiřujících modulech a komunikačních převodnících řady **AMRIO-xxx** a **DMM-xxx** a ve všech svých řídicích systémech, řídicích terminálech a programovatelných regulátorech a ovladačích vybavených rozhraním RS485 nebo RS232. Určení Master stanice/Slave stanice závisí na konkrétní implementaci.

Poznámka
Komunikační rozhraní řídicího systému, kde probíhá komunikace ptotokolem MODBUS RTU, 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 „EsiDet – tvorba regulačních procesů“ v kapitole „Obsah/Prvky programové obsluhy/Komunikace“ v části „Mapování Modbus proměnných“.

2.1 Podporované funkce MODBUS

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

Funkce	Název	Popis
1	Read Coils	Čtení jednoho/více binárních výstupů.
2	Read Discrete Inputs	Čtení jednoho/více binárních vstupů.
3	Read Holding Registers	Čtení jednoho/více uchovávacích registrů.
4	Read Input Registers	Čtení jednoho/více vstupních registrů.
5	Write Single Coil	Zápis jednoho binárního výstupu.
6	Write Single Register	Zápis jednoho uchovávacího registru.
15	Write Multiple Coils	Zápis více binárních výstupů.
16	Write Multiple Registers	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 o tomto se dočtete v nápovědě DetStudia „EsiDet – tvorba regulačních procesů“ v kapitole „Obsah/Prvky programové obsluhy/Komunikace“ v části „Mapování Modbus proměnných“.

3 Zapojení komunikační sítě

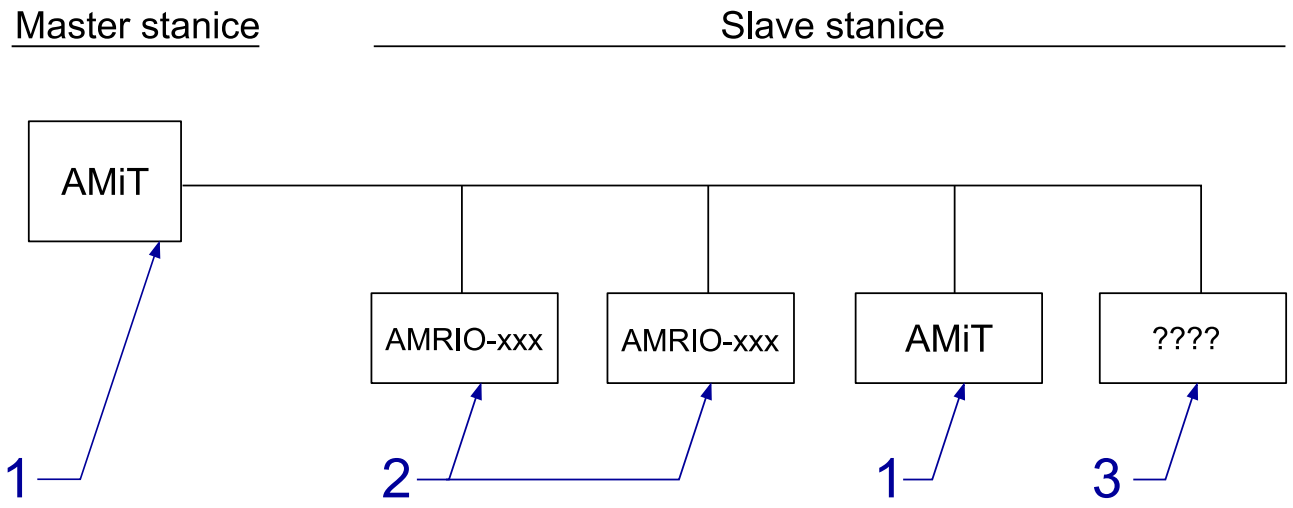
Pro správné plnění požadované funkce celé sítě MODBUS je nutno správně navrhnout, zapojit, nakonfigurovat jednotlivé moduly sítě a naprogramovat komunikaci v řídicích systémech.

Komunikace prostřednictvím protokolu MODBUS RTU probíhá typicky po síti RS485. K řídicímu systému lze jiné zařízení (např. moduly **AMRIO-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 síti MODBUS vystupovat jako Master stanice nebo jako Slave stanice. V roli Master stanice typicky v kombinaci s **AMRIO-xxx** moduly, nástěnnými ovladači **AMR-OPxx** a technologickými zařízeními třetích firem (např. akční členy) nebo v roli Slave stanice jako součást rozsáhlejších sítí.

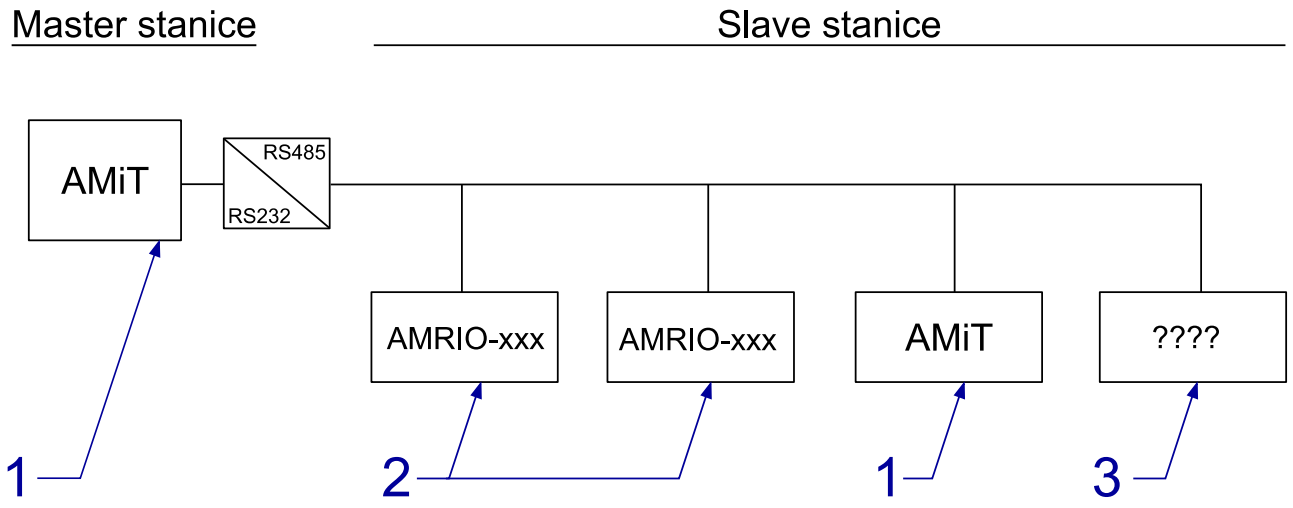
Rozšiřující moduly **AMRIO-xxx**, komunikační převodníky s protokolem MODBUS RTU a nástěnné ovladače označované **AMR-OPxx** jsou v síti typicky v roli Slave stanice.



Obr. 1 – Komunikace pomocí protokolu MODBUS RTU

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 RTU 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

4 Časové poměry na síti

Komunikace probíhá s určitou periodou, na jejímž začátku Master stanice aktivuje rozhraní a následně se vykomunikují požadavky pro každý vzdálený bod. Aktivita rozhraní končí vykomunikováním posledního rámce.

4.1 Doba komunikace

Master stanice 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
115200	8	0,6× registr, 0,3× každá započatá osmice binárů

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

4.2 Priorita komunikace

Komunikace probíhá s určitou periodou, která je dána nastavenou prioritou komunikace. Prioritu komunikace je možné nastavit jako vlastnost celého komunikačního objektu (např. objekty **`RoomUnit`**, **`AMRIO-xxx`** nebo **`DMM-xxx`**) nebo je možné každé komunikační proměnné nastavit prioritu individuálně (v objektech **`ModbusDevice`** a **`ModbusAmr`**). Priority komunikace slouží k optimalizaci zatíženosti komunikační linky. Priorita se dá nastavit:

- pro celý komunikační objekt v okně Vlastnosti – vlastnost **`Priority`**,
- pro komunikační proměnnou ve správci proměnných – sloupec **`Priorita`**.

Každá komunikovaná proměnná/vlastnost může mít nastavenou jednu ze čtyř priorit:

- None (žádná),
- Low (nízká),
- Normal (normální),
- High (vysoká).

Automatická priorita komunikace

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

Priorita čtení	Základní perioda komunikace [ms]
Low	5000
Normal	900
High	200

Priorita **`Low`** má základní periodu vyčítání 5000 ms. Je-li v aplikaci definováno více procesů a hodnota periody vykonávání některého z procesů je větší než 5000 ms, převezme se tato perioda i pro četnost vyčítání hodnot proměnných/vlastností s prioritou **`Low`**.

Priorita **`Normal`** má fixní periodu vyčítání 900 ms.

Priorita **High** má základní periodu vyčítání 200 ms. Je-li v aplikaci definováno více procesů a hodnota periody vykonávání některého z procesů je menší než 200 ms, převezme se tato perioda i pro četnost vyčítání hodnot proměnných/vlastností s prioritou **High**.

V případě priority **Low**, **Normal** a **High** se zápis do komunikační proměnné provede co nejdříve po vykonání kódu daného procesu/obrazovky. Pro komunikační vlastnosti objektů (např. objekty **RoomUnit**, **AMRIO-xxx** nebo **DMM-xxx**) se komunikují čtecí i zápisové funkce.

Volbou priority programátor volí, s jakou periodou se má daná proměnná/vlastnost komunikovat. Je tedy zřejmé, že není vhodné všude využít komunikaci s prioritou **High**, neboť by rozhraní mohlo být zcela přetíženo komunikačními požadavky.

Pozor!
*Komunikační požadavky se vyřizují postupně, dokud není fronta komunikačních požadavků vyřízená do konce. To znamená, že pokud se v jednom okamžiku označí např. dvě proměnné s prioritou **High** a dalších 20 s prioritou **Low** a celá tato komunikace trvá např. 600 ms, tak zmíněné dvě proměnné s prioritou **High** budou opětovně komunikovány **až po** dokončení komunikace všech označených proměnných/vlastností. 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.*

Manuální priorita komunikace

V případě, že automatická priorita komunikace proměnných/vlastností nedovoluje korektní požadovaný způsob komunikace se Slave stanicí, je nutné využít manuální priority komunikace. Tato se nastaví volbou **None** v požadovaném sloupci/vlastnosti priority komunikace.

Pro spuštění manuálního čtení proměnných se využívá v kódu procesu metoda **Refresh()**. Proměnné, které mají nastavenou prioritu **None** a jsou přiřazeny v obrazovkových prvcích se vyčítají s periodou refreshe obrazovky při jejím zobrazení na terminálu.

Pozor!
*V případě komunikace vlastností komunikačních objektů (např. objekty **RoomUnit**, **AMRIO-xxx** nebo **DMM-xxx**) se hodnoty vlastností nekomunikují s refreshem obrazovky. Proto je nutné mít v periodickém procesu vždy vhodně umístěnou metodu **Refresh()**.*

Zápis do komunikační proměnné/vlastnosti se provede co nejdříve po vykonání kódu daného procesu.

4.3 Komunikace při rozpadu spojení

V případě, že není možná komunikace se Slave stanicí, resp. na 10 dotazů nepřišla odpověď od Slave stanice, je signalizován stav Disconnected. Jakmile nastane tento stav, ignoruje se následujících 10 požadavků na komunikaci. Následný požadavek bude vykomunikován. Pokud na něj nepřijde odpověď, bude opět následujících 10 požadavků ignorováno. Tento algoritmus se opakuje do doby, dokud není úspěšně navázaná komunikace se Slave stanicí.

4.4 Detekce ztráty spojení u modulů AMRIO-xxx a DMM-xxx

Pro případ fyzického přerušení sítě mají všechny výstupní moduly **AMRIO-xxx** s aplikací z výroby a **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.

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 stanice

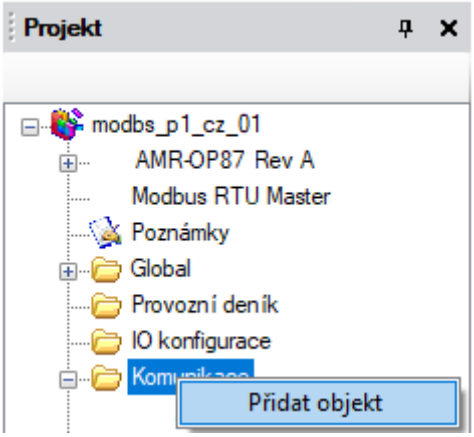
Definice komunikace protokolem MODBUS RTU v roli Master stanice probíhá pomocí trojí definice:

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

Poznámka
*K datu vytvoření této aplikační poznámky nelze jako Master stanici v síti MODBUS definovat regulátor **AMR-DI2RDO2**, **AMR-UI2RDO2/(DM)** a programovatelné nástěnné ovladače řady **AMR-OP3x(RH)** a **AMR-OP4x(RHC)**.*

5.1 Komunikační definice

Vytvoření komunikační definice protokolu MODBUS RTU v roli Master stanice reprezentuje vložení komunikačního objektu **`ModbusMaster`** do aplikace. Toto se provádí v DetStudiu v okně Projekt v uzlu „Projekt/Komunikace“. Po vyvolání kontextového menu nad touto položkou se vybere položka **`Přidat objekt`**. Ze zobrazeného okna bude proveden výběr objektu **`ModbusMaster`**.



Obr. 3 – Položka „Přidat objekt“ v definici uzlu „Komunikace“

Po vložení objektu **`ModbusMaster`** bude vytvořen komunikační uzel **`ModbusMaster1`** s výchozími hodnotami vlastností:

- **`BaudRate`**: 38400
- **`Parity`**: Even
- **`Routing`**: True
- **`SerialPort`**: COM0
- **`StopBits`**: One
- **`UndefinedSlaveRouting`**: False
- **`UndefinedSlaveTimeout`**: 10

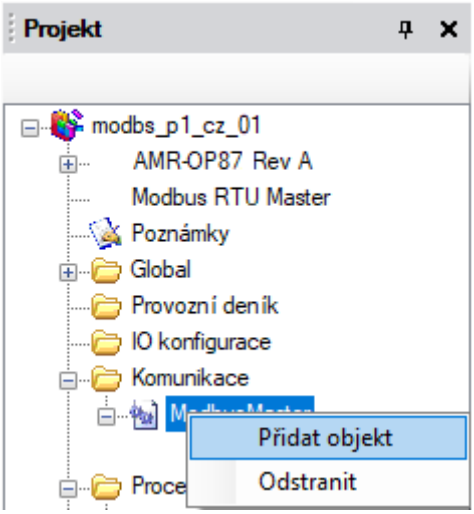
Popis vlastností je k dispozici v nápovědě DetStudia „EsiDet – tvorba regulačních procesů“ v kapitole „Obsah/Prvky programové obsluhy/Komunikace/ModbusMaster“.

Některým zařízením firmy AMiT lze adresu, komunikační rychlost a paritu v síti MODBUS nastavit DIP přepínači. Pokud je zařízení osazeno více COM rozhraními, definují DIP přepínače komunikační parametry rozhraní COM1. Komunikační parametry rozhraní COM0 se v takovém případě nastavují programově (typicky v Init procesu). Pokud je zařízení osazeno DIP přepínači, jsou vlastnosti Address, BaudRate a Parity pro dané rozhraní určené pouze pro čtení.

5.2 Definice Slave stanice a datových bodů pro komunikaci

Pro každou Slave stanici, se kterou má Master stanice komunikovat je nutné do aplikace vložit jeden komunikační objekt. Pro komunikaci s jednotlivými Slave stanicemi se pak musí nadefinovat její adresa v síti MODBUS a případně seznam komunikačních bodů.

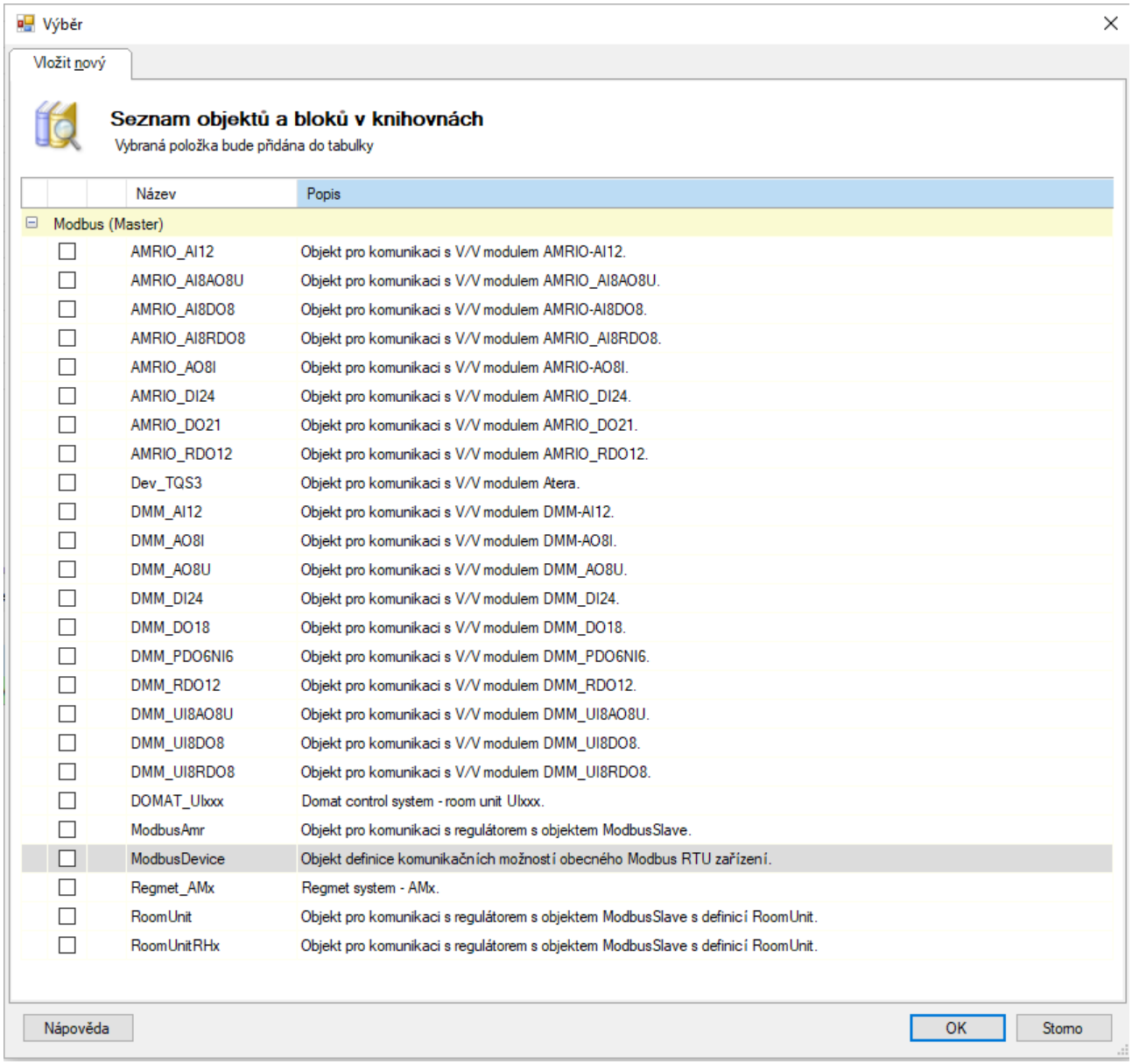
Jednotlivé Slave stanice se v okně Projekt definují v uzlu „Projekt/Komunikace/ModbusMaster“ konkrétní ModbusMasterX objekt. Po vyvolání kontextového menu nad touto položkou se vybere položka **`Přidat objekt`**.



Obr. 4 – Položka „Přidat objekt“ v definici uzlu „ModbusMaster“

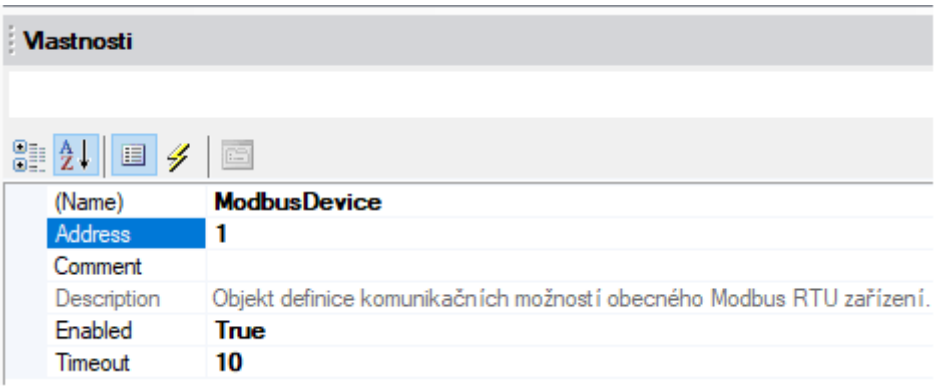
Ze zobrazeného okna bude proveden výběr objektu pro Slave stanici. Je možné volit Slave stanice jako objekt pro konkrétní zařízení z produkce firmy AMiT nebo jako objekt pro obecnou komunikaci se vzdáleným zařízením.

K datu vytvoření aplikační poznámky jsou k dispozici tyto objekty:



Obr. 5 – Okno se seznam komunikačních objektů pro obsluhu Slave stanic

Po vložení definice bude vytvořen komunikační uzel, u kterého v okně Vlastnosti je potřeba nastavit adresu Slave stanice.



Obr. 6 – Nastavení adresy Slave stanice

Popis vlastností komunikačního objektu je k dispozici v nápovědě DetStudia „EsiDet – tvorba regulačních procesů“ v kapitole „Obsah/Prvky programové obsluhy/Komunikace/ModbusDevice“.

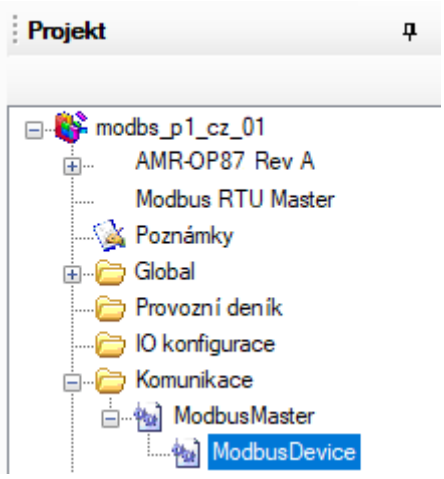
Do aplikace je možné vložit Slave stanice jako objekt pro konkrétní zařízení z produkce firmy AMiT nebo jako objekt pro obecnou komunikaci se Slave stanicí.

5.2.1 Objekty pro obecnou komunikaci se Slave stanicí

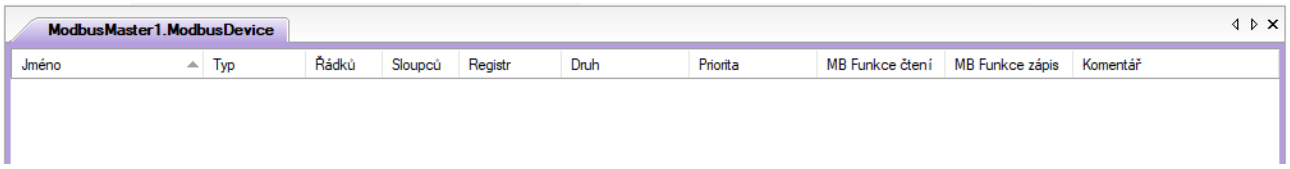
Objekty pro obecnou komunikaci se Slave stanicí jsou **`ModbusDevice`** nebo **`ModbusAmr`**. V objektu **`ModbusDevice`** lze definovat pozici jednotlivých datových bodů (registrů) od hodnoty 0. V objektu **`ModbusAmr`** lze definovat adresu uchovávacích (holding) registrů od hodnoty 100. Prvních 100 (0 až 99) uchovávacích (holding) registrů je určeno pro systémové účely.

V typových řešeních pro obsluhu vstupů a výstupů, která jsou dostupná pro programovatelné regulátory na webu amitomation.cz se využívá rozložení registrů od adresy 100. Z toho důvodu je pro tyto typová řešení vhodné použít objekt **`ModbusAmr`**.

Datové body, které se mají s jednotlivými Slave stanicemi komunikovat se definují v otevřené záložce objektu **`ModbusDevice`** nebo **`ModbusAmr`**. Záložka se otevře dvojklikem myší nad konkrétním objektem **`ModbusDevice`** nebo **`ModbusAmr`** v okně Projekt.



Obr. 7 – Vybraný objekt ModbusDevice



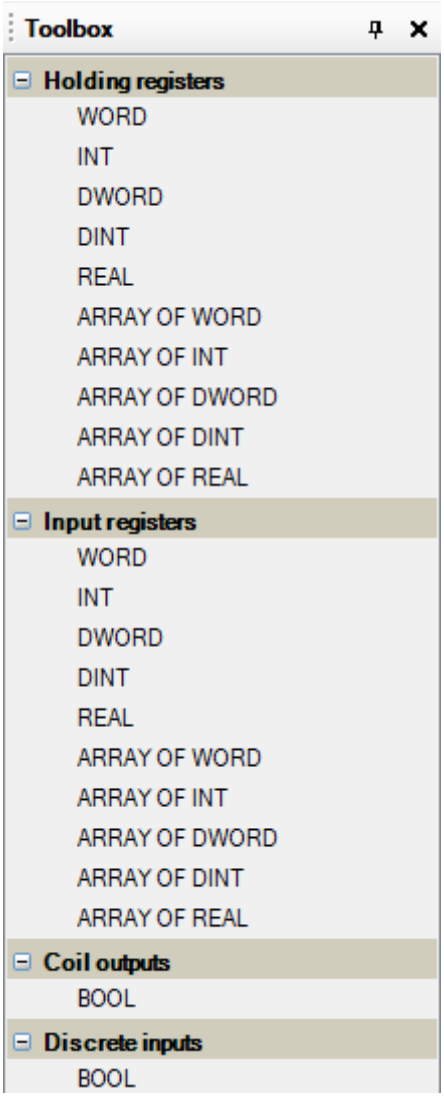
Obr. 8 – Otevřená záložka objektu ModbusDevice

K datu vytvoření aplikační poznámky lze definovat následující datové body:

- uchovávací registry (holding registers) typu WORD, INT, DWORD, DINT, REAL, a pole k těmto datovým typům,
- vstupní registry (input registers) typu WORD, INT, DWORD, DINT, REAL, a pole k těmto datovým typům,
- binární výstupy (coils) typu Bool,
- binární vstupy (discrete inputs) typu Bool.

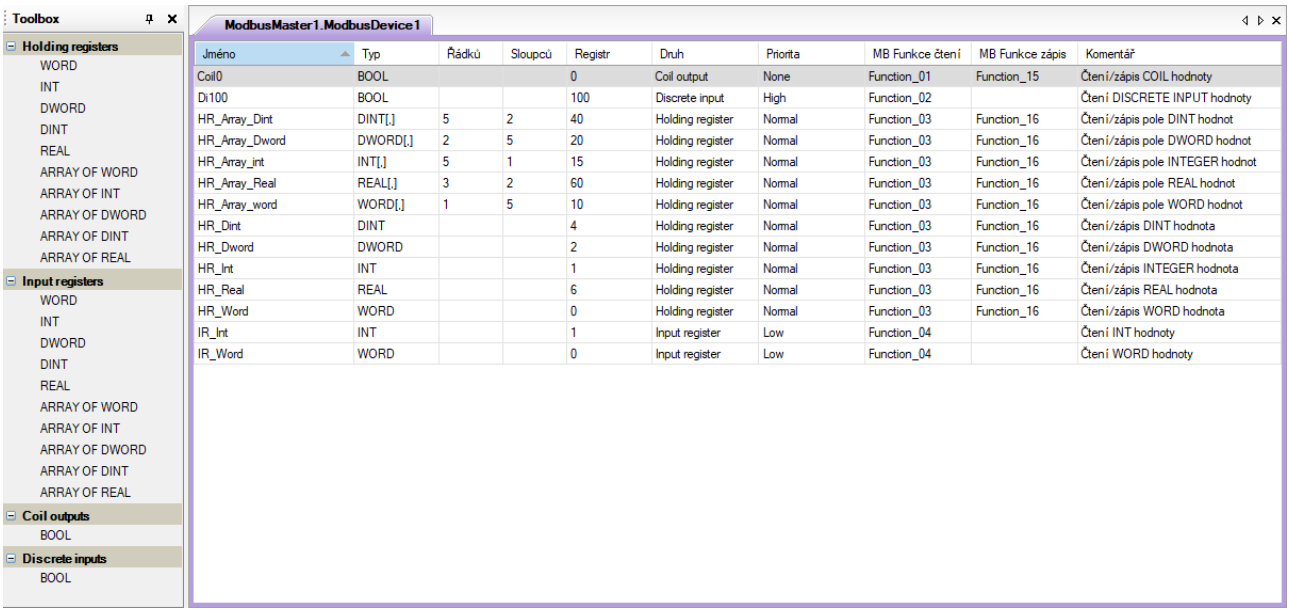
Poznámka
Typy DWORD, DINT a REAL jsou ukládány do dvojice po sobě jdoucích registrů.

Vzhledem k tomu, že 32 bitové typy (DWORD, DINT a REAL) nejsou v protokolu MODBUS RTU 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.



Obr. 9 – Okno Toolboxu pro vložení registrů

Jednotlivé datové body se definují přetažením požadovaného typu pomocí myši z okna Toolbox do otevřené záložky objektu **`ModbusDevice`** nebo **`ModbusAmr`**.



Jméno	Typ	Řádků	Sloupců	Registr	Druh	Priorita	MB Funkce čtení	MB Funkce zápis	Komentář
Coil0	BOOL			0	Coil output	None	Function_01	Function_15	Čtení/zápis COIL hodnoty
Di100	BOOL			100	Discrete input	High	Function_02		Čtení DISCRETE INPUT hodnoty
HR_Array_Dint	DINT[,]	5	2	40	Holding register	Normal	Function_03	Function_16	Čtení/zápis pole DINT hodnot
HR_Array_Dword	DWORD[,]	2	5	20	Holding register	Normal	Function_03	Function_16	Čtení/zápis pole DWORD hodnot
HR_Array_int	INT[,]	5	1	15	Holding register	Normal	Function_03	Function_16	Čtení/zápis pole INTEGER hodnot
HR_Array_Real	REAL[,]	3	2	60	Holding register	Normal	Function_03	Function_16	Čtení/zápis pole REAL hodnot
HR_Array_word	WORD[,]	1	5	10	Holding register	Normal	Function_03	Function_16	Čtení/zápis pole WORD hodnot
HR_Dint	DINT			4	Holding register	Normal	Function_03	Function_16	Čtení/zápis DINT hodnota
HR_Dword	DWORD			2	Holding register	Normal	Function_03	Function_16	Čtení/zápis DWORD hodnota
HR_Int	INT			1	Holding register	Normal	Function_03	Function_16	Čtení/zápis INTEGER hodnota
HR_Real	REAL			6	Holding register	Normal	Function_03	Function_16	Čtení/zápis REAL hodnota
HR_Word	WORD			0	Holding register	Normal	Function_03	Function_16	Čtení/zápis WORD hodnota
IR_Int	INT			1	Input register	Low	Function_04		Čtení INT hodnoty
IR_Word	WORD			0	Input register	Low	Function_04		Čtení WORD hodnoty

Obr. 10 – Záložka ModbusDevice s nadefinovanými registry

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

Jméno
Jméno datového bodu, pod kterým bude využíván v rámci aplikace.

Typ
Typ datového bodu (WORD, INT, …). Pro pole registrů (ARRAY OF WORD, ARRAY OF INT, …) lze nastavit počet řádků a sloupců.

Registr
Adresa datového bodu na Slave stanici. Datové body Bool (jeden binární vstup/výstup) a WORD, INT (jeden vstupní/uchovávací registr) zabírají vždy pouze jednu adresu. Datové body typu DWORD, DINT a REAL (dva vstupní/uchovávací registry) zabírají vždy dvě po sobě jdoucí adresy.

Druh
Druh datového bodu (binární vstup/výstup, …). Nelze editovat – předvyplněn po vložení datového bodu z okna Toolbox.

Priorita
Priorita komunikace. Jednotlivým datovým bodům lze nastavit prioritu komunikace. Význam jednotlivých úrovní priority komunikace je popsán v kapitole 4.2 Priorita komunikace.

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

MB Funkce zápis
Číslo MODBUS funkce, která bude využívána pro zápis do datového bodu. Pro 32 bitové typy a pole se nabízí pouze funkce 16.

Komentář
Poznámka k datovému bodu.

5.2.2 Objekty pro konkrétní modul z produkce firmy AMiT

Po vložení objektů pro konkrétní modul z produkce firmy AMiT (**`AMRIO-xxx`** s aplikací z výroby a **`DMM-xxx`**) dojde k přidání modulu do uzlu IO konfigurace. V rámci záložky IO konfigurace lze nastavit inicializační hodnoty vlastností pro vzdálené vstupy/výstupy. Více informací o nastavení parametrů IO konfigurace lze nalézt v nápovědě DetStudia „EsiDet – tvorba regulačních procesů“ v kapitole „Obsah/Prvky programové obsluhy/IO“.

IO

Poz.	Svorka	Název	Mód	Filtr	Parametry		Hodnota	Komentář
IO (2)								
		Vbatt						Napětí zálohovací baterie [V].
		LED						LED na panelu
ModbusMaster1.AMRIO_AI121 (12)								
7	AI0	AI0	V	0	0 V - 10 V -> 0 - 10	...		
8	AI1	AI1	V	0	0 V - 10 V -> 0 - 10	...		
9	AI2	AI2	V	0	0 V - 10 V -> 0 - 10	...		
10	AI3	AI3	V	0	0 V - 10 V -> 0 - 10	...		
12	AI4	AI4	V	0	0 V - 10 V -> 0 - 10	...		
13	AI5	AI5	V	0	0 V - 10 V -> 0 - 10	...		
14	AI6	AI6	V	0	0 V - 10 V -> 0 - 10	...		
15	AI7	AI7	V	0	0 V - 10 V -> 0 - 10	...		
17	AI8	AI8	V	0	0 V - 10 V -> 0 - 10	...		
18	AI9	AI9	V	0	0 V - 10 V -> 0 - 10	...		
19	AI10	AI10	V	0	0 V - 10 V -> 0 - 10	...		
20	AI11	AI11	V	0	0 V - 10 V -> 0 - 10	...		

Obr. 11 – Obsluha modulu **AMRIO-AI12** v IO konfiguraci

V rámci procesu aplikace nabízí objekty vnitřní běhové vlastnosti, které je možné číst (vstupní signály) nebo do nich zapisovat (výstupní signály). U každého objektu se počet a typ těchto vlastností liší. Seznam a popis vlastností k jednotlivým komunikačním objektům je k dispozici v nápovědě DetStudia „EsiDet – tvorba regulačních procesů“ v kapitole „Obsah/Prvky programové obsluhy/Komunikace/ModbusMaster“.

5.3 Stavy komunikace

Pro detekci rozpadu komunikace se Slave stanicí se využije vlastnosti **`Disconnected`** objektu komunikace Slave stanice.

Pro zjištění stavu komunikace datového bodu/objektu se využívá vlastnost **`ComState`**. Vlastnost **`ComState`** lze využít pro podrobnou detekci stavu komunikace.

Vlastnost **`LastError`** datového bodu/objektu 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.

Význam bitů vlastnosti ComState

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 stanice odpojená. Provádí se pouze občasný pokus o komunikaci pro testování, zda je zařízení opětovně připojené.
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.
10	Má hodnotu 1, pokud je právě komunikovaný požadavek zápis.
11	Má hodnotu 1, pokud poslední ukončená komunikace byl zápis.

Význam bitů vlastnosti LastError

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.

5.4 Příklad parametrizace řídicího systému jako Master stanice

Uvažujme aplikaci, ve které jeden řídicí systém **AMR-OP87 RevA** má sloužit jako Slave stanice pro druhý řídicí systém **AMR-OP87 RevA** (Master stanice). Rozložení registrů Slave stanice je uvedeno v kapitole 6.4 „Příklad parametrizace řídicího systému jako Slave stanice“.

Je známé rozložení uchovávacích (holding) registrů:
- adresa 0, typ WORD – HR_Word,
- adresy 1, typ INT – HR_Int,
- adresy 2 a 3, typ DWORD – HR_Dword,
- adresa 4 a 5, typ DINT – HR_Dint,
- adresa 6 a 7, typ REAL – HR_Real
- adresy 10 až 14, typ Array of WORD – HR_Array_word,
- adresy 15 až 19, typ Array of INT – HR_Array_int,
- adresy 20 až 39, typ Array of DWORD – HR_Array_Dword,
- adresy 40 až 59, typ Array of DINT – HR_Array_Dint,
- adresy 60 až 71, typ Array of REAL – HR_Array_Real.

Rozložení vstupních (input) registrů:
- adresa 0, typ WORD – IR_Word,
- adresy 1, typ INT – IR_Int.

Rozložení binárních vstupů (Discrete inputs):
- adresa 100, typ BOOL – Di100.

Rozložení binárních výstupů (Coil outputs):
- adresa 0, typ BOOL – Coil0.

Do aplikace je vložen komunikační objekt **`ModbusMaster1`** a v něm je vytvořena definice objektu **`ModbusDevice1`**.

V definici tabulky **`ModbusDevice1`** se nadefinuje deset řádků „Holding registers“, dva řádky „Input registers“, jeden řádek „Discrete Inputs“ a jeden řádek „Coil outputs“. V definičních řádcích zvolíme

priority například tak, že Holding registry budou s prioritou **`Normal`**, Input registr s prioritou **`Low`**, Discrete Input registr s prioritou **`High`** a Coil output registr s prioritou **`None`** (manuální zápis).

Výsledná tabulka vypadá dle následujícího obrázku.

ModbusMaster1.ModbusDevice1

Jméno	Typ	Řádků	Sloupců	Registr	Druh	Priorita	MB Funkce čtení	MB Funkce zápis	Komentář
Coil0	BOOL			0	Coil output	None	Function_01	Function_15	Čtení/zápis COIL hodnoty
Di100	BOOL			100	Discrete input	High	Function_02		Čtení DISCRETE INPUT hodnoty
HR_Array_Dint	DINT[,]	5	2	40	Holding register	Normal	Function_03	Function_16	Čtení/zápis pole DINT hodnot
HR_Array_Dword	DWORD[,]	2	5	20	Holding register	Normal	Function_03	Function_16	Čtení/zápis pole DWORD hodnot
HR_Array_int	INT[,]	5	1	15	Holding register	Normal	Function_03	Function_16	Čtení/zápis pole INTEGER hodnot
HR_Array_Real	REAL[,]	3	2	60	Holding register	Normal	Function_03	Function_16	Čtení/zápis pole REAL hodnot
HR_Array_word	WORD[,]	1	5	10	Holding register	Normal	Function_03	Function_16	Čtení/zápis pole WORD hodnot
HR_Dint	DINT			4	Holding register	Normal	Function_03	Function_16	Čtení/zápis DINT hodnota
HR_Dword	DWORD			2	Holding register	Normal	Function_03	Function_16	Čtení/zápis DWORD hodnota
HR_Int	INT			1	Holding register	Normal	Function_03	Function_16	Čtení/zápis INTEGER hodnota
HR_Real	REAL			6	Holding register	Normal	Function_03	Function_16	Čtení/zápis REAL hodnota
HR_Word	WORD			0	Holding register	Normal	Function_03	Function_16	Čtení/zápis WORD hodnota
IR_Int	INT			1	Input register	Low	Function_04		Čtení INT hodnoty
IR_Word	WORD			0	Input register	Low	Function_04		Čtení WORD hodnoty

Obr. 12 – Základní definice datových bodů

Posledním krokem je vytvoření manuální komunikace binárního výstupu. Aby se zamezilo nadbytečným komunikacím bináru, využije se algoritmus, ve kterém k zápisu hodnoty do MODBUS sítě dojde pouze při změně hodnoty proměnné **`MB_Write`**. Proměnná **`MB_Write`** je typu BOOL. V kódu se hlídá stav komunikace pomocí vlastnosti **`Disconnected`**.

Výsledný kód v periodickém procesu bude znít:

```
If not ModbusMaster1.ModbusDevice1.Disconnected then
   If ModbusMaster1.ModbusDevice1.Coil0 != MB_Write then
      ModbusMaster1.ModbusDevice1.Coil0 = MB_Write;
   EndIf;
EndIf;
```

Uvedený algoritmus je součástí přílohy aplikační poznámky. Jedná se o ukázkovou aplikaci s názvem „modbs_p1_cz_xxx.dsox“ vytvořenou ve vývojovém prostředí DetStudio. Aplikace je vytvořena pro řídicí systém **AMR-OP87 RevA**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

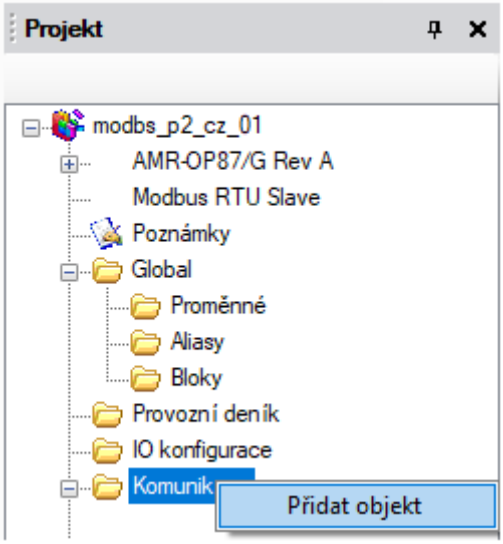
6 Řídicí systém jako Slave stanice

Aby komunikace v síti MODBUS fungovala správně, musí mít každá Slave stanice v síti nastavenu stejnou komunikační rychlost, stejnou paritu a každá Slave stanice musí mít nastavenu jedinečnou adresu. Definice komunikace protokolem MODBUS RTU v roli Slave stanice probíhá pomocí dvojí definice:

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

6.1 Komunikační definice

Vytvoření komunikační definice protokolu MODBUS RTU v roli Slave stanice reprezentuje vložení komunikačního objektu **`ModbusSlave`** do aplikace. Toto se provádí v DetStudiu v okně Projekt v uzlu „Projekt/Komunikace“. Po vyvolání kontextového menu nad touto položkou se vybere položka **`Přidat objekt`**. Ze zobrazeného okna bude proveden výběr objektu **`ModbusSlave`**.



Obr. 13 – Položka „Přidat objekt“ v definici uzlu „Komunikace“

Po vložení komunikačního objektu bude vytvořen komunikační uzel **`ModbusSlave1`** s výchozími hodnotami vlastností:

- **`Address`**: 1
- **`BaudRate`**: 38400
- **`Guard Time`**: 30000
- **`HoldingRegistryOffset`**: AMiT (from 100)
- **`Parity`**: Even
- **`SerialPort`**: COM0 RS485

Popis vlastností je k dispozici v nápovědě DetStudia „EsiDet – tvorba regulačních procesů“ v kapitole „Obsah/Prvky programové obsluhy/Komunikace/ModbusSlave“.

Některým zařízením firmy AMiT lze adresu, komunikační rychlost a paritu v síti MODBUS RTU nastavit DIP přepínači. Pokud je zařízení osazeno více COM rozhraními, definují DIP přepínače komunikační parametry rozhraní COM1. Komunikační parametry rozhraní COM0 se v takovém případě nastavují programově (typicky v Init procesu). Pokud je zařízení osazeno DIP přepínači, jsou vlastnosti Address, BaudRate a Parity pro dané rozhraní určené pouze pro čtení.

6.2 Definice datových bodů pro komunikaci

Po nadefinování komunikačního objektu **ModbusSlave** 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 register, Input register, Coil a Discrete input) proměnné řídicího systému, které mají být dostupné pod zvolenými adresami.

K datu vytvoření aplikační poznámky lze definovat následující datové body:

- uchovávací registry (holding register) typu WORD, INT, DWORD, DINT, REAL, a pole k těmto datovým typům,
- vstupní registry (input register) typu WORD a INT, a pole k těmto datovým typům,
- binární výstupy (coil) typu Bool,
- binární vstupy (discrete input) typu Bool.

Pro uchovávací registry (holding register) lze využít šablonu předdefinovaných registrů **RoomUnit** a **RoomUnitRHx**. Šablona se využije v případě, kdy bude použito propojení mezi dvěma zařízeními (např. **AMR-FCT20/DM** (Master stanice) a **AMR-OP7xRH** (Slave stanice)). V aplikaci pro Slave stanici bude vložena jedna ze šablon **RoomUnit** a **RoomUnitRHx**. V aplikaci pro Master stanici bude vložen komunikační objekt **ModbusMaster** a pod ním bude umístěn odpovídající objekt **RoomUnit** a **RoomUnitRHx**.

Definici jednotlivých datových bodů a šablon lze provést např. přetažením z okna ToolBox. Více informací o definici datových bodů lze nalézt v nápovědě DetStudia „EsiDet – tvorba regulačních procesů“ v kapitole „Obsah/Práce s proměnnými/Správce proměnných, aliasů“.

6.3 Stavy komunikace

Signalizace detekce komunikace ze strany Master stanice je k dispozici ve vlastnosti **Connect** komunikačního objektu **ModbusSlave**. Dále lze sledovat vlastnosti **Incoming** a **Outgoing** komunikačního objektu **ModbusSlave**. Tyto vlastnosti slouží jako počítadlo úspěšně přijatých zápisových funkcí a počítadlo odeslaných odpovědí na úspěšně přijaté čtecí funkce.

6.4 Příklad parametrizace řídicího systému jako Slave stanice

Je požadavek na vytvoření aplikace, po jejímž nahrání bude řídicí systém **AMR-OP87 RevA** se chovat jako Slave stanice pro Master stanici s aplikací modbs_p1_cz_xxx.dsox, která je součástí přílohy.

Do aplikace je vložen komunikační objekt **ModbusSlave1**. Po vložení komunikačního objektu budou nastaveny hodnoty vlastností takto:

- **Address**: 1
- **BaudRate**: 38400
- **Guard Time**: 30000
- **HoldingRegistryOffset**: General (from 0)
- **Parity**: Even
- **SerialPort**: COM0 RS485 galvanic

V dalším kroku je provedeno dvojí kliknutí na komunikační objekt **ModbusSlave1** v okně Projektu, čímž dojde k vyvolání definiční tabulky komunikačních datových bodů. Jednotlivé datové body se definují přetažením požadovaného typu pomocí myši z okna Toolbox do otevřené záložky objektu **ModbuSlave1**. Rozložení jednotlivých datových bodů je uvedeno na následujících obrázcích.

ModbusSlave1

Holding registr | Input register | Coil | Discrete input

Jméno	Typ	Adresa	Adresa koncová	Reference	Řádků	Sloupců	Komentář
HR_Word	WORD	0	0	NONE			WORD hodnota
HR_Int	INT	1	1	NONE			INTEGER hodnota
HR_Dword	DWORD	2	3	NONE			DWORD hodnota
HR_Dint	DINT	4	5	NONE			DINT hodnota
HR_Real	REAL	6	7	RandomREAL.RVal			REAL hodnota jako reference
HR_Array_Word	ARRAY OF WORD	10	14	NONE	1	5	Pole WORD hodnot
HR_Array_Int	ARRAY OF INT	15	19	NONE	5	1	Pole INT hodnot
HR_Array_Dword	ARRAY OF DWORD	20	39	NONE	2	5	Pole DWORD hodnot
HR_Array_Dint	ARRAY OF DINT	40	59	NONE	5	2	Pole DINT hodnot
HR_Array_Real	ARRAY OF REAL	60	71	NONE	3	2	Pole REAL hodnot

Obr. 14 – Rozložení uchovávacích (holding) registrů

ModbusSlave1

Holding registr | Input register | Coil | Discrete input

Jméno	Typ	Adresa	Adresa koncová	Reference	Řádků	Sloupců	Komentář
IR_Word	WORD	0	0	NONE			WORD hodnota
IR_Int	INT	1	1	NONE			INTEGER hodnota

Obr. 15 – Rozložení vstupních (input) registrů

ModbusSlave1

Holding registr | Input register | Coil | Discrete input

Jméno	Typ	Adresa	Adresa koncová	Reference	Řádků	Sloupců	Komentář
Coil1	BOOL	0	0	NONE			Binární výstup

Obr. 16 – Rozložení binárních výstupů (coil)

ModbusSlave1

Holding registr | Input register | Coil | Discrete input

Jméno	Typ	Adresa	Adresa koncová	Reference	Řádků	Sloupců	Komentář
Di1	BOOL	100	100	NONE			Binární vstup

Obr. 17 – Rozložení binárních vstupů (discrete input)

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

Jméno
Jméno datového bodu, pod kterým bude využíván v rámci aplikace.

Typ
Typ datového bodu (BOOL, WORD, INT, …). Pro pole registrů (ARRAY OF WORD, ARRAY OF INT, …) lze nastavit počet řádků a sloupců.

Adresa
Adresa datového bodu na Slave stanici. Pořadové číslo uchovávacích (holding) registrů je ovlivněno položkou **`HoldingRegistryOffset`** objektu **`ModbusSlave`**. Pokud je v položce číslování

uchovávacích (holding) registrů od hodnoty 0 (General from 0), lze je definovat od libovolné adresy (0, 1, 3, 4 atd.) Pokud je v položce číslování uchovávacích (holding) registrů od hodnoty 100 (AMiT from 100) lze definovat adresy od hodnoty 100.

Koncová adresa
Needitovatelná informace o poslední použité adrese.

Reference
Reference na vlastnost bloku nebo objektu. U datových bodů lze definovat komunikační registr jako referenci na vlastnost bloku nebo objektu. Typ komunikační proměnné se automaticky přizpůsobí typu referencované vlastnosti. Čtením daného registru se přímo čte hodnota referencované vlastnosti a zápisem do daného registru se hodnota přímo zapíše do referencované vlastnosti.

Komentář
Poznámka k datovému bodu.

Uvedený algoritmus je součástí přílohy aplikační poznámky. Jedná se o ukázkovou aplikaci s názvem „modbs_p2_cz_xxx.dsox“ vytvořenou ve vývojovém prostředí DetStudio. Aplikace je vytvořena pro řídicí systém **AMR-OP87 RevA**. 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 Programová obsluha AMR-OPxx

7.1.1 AMR-OP7x(RH)/AMR-OP60RH/AMR-OP4x/AMR-OP3xA(RH)

Nástěnné ovladače **AMR-OP7x(RH)**, **AMR-OP60RH**, **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.

Do aplikace je vložen komunikační objekt **`ModbusMaster1`** a v něm je vytvořena definice objektu **`RoomUnitRHx`**. Objekt **`RoomUnitRHx`** slouží pro zjednodušenou komunikaci s nástěnným ovladačem řady **AMR-OPxxRHx** s typovou aplikací.

Popis a význam jednotlivých registrů komunikačního objektu **`RoomUnitRHx`** je možné nalézt v nápovědě DetStudia „EsiDet – tvorba regulačních procesů“ v kapitole „Obsah/Prvky programové obsluhy/Komunikace/ModbusMaster/RoomUnitRHx“ nebo v dokumentaci daných nástěnných ovladačů.

V ukázkových aplikacích je objektu **`RoomUnitRHx`** nastaveno nové jméno **`RCU1`**.

Pro zjišťování stavu komunikace s nástěnným ovladačem se využije vlastnosti **`Disconnected`** objektu **`ModbusMaster1.RCU1`**.

Zpracování stavu po restartu ovladače nebo rozpadu komunikace

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

```
If (ModbusMaster1.RCU1.Status == 0x00FF) Then
     StatusReset = 0x00FF;
     StatusSet = StatusSet | ((RoomMode << 1) & 0x0006);
     StatusSet = StatusSet | ((FanMode << 4) & 0x0070);
     StatusSet.3 = Switch_ON_OFF;
     StatusWrite = true;
…
```

Načtení nových hodnot z ovladače

Nástěnné ovladače využívají registr **`ValueChanged`**, který reprezentuje bit č. 0 dvojregistru **`Status`** (102-103), pro signalizaci změny hodnoty ze strany ovladače. Vyhodnocení změny v ovladači se provádí se zpožděním (blok TimerOffRUchange) z důvodu nastavení požadovaného režimu místnosti a ventilátoru. Nastavení režimů může být provedeno vícenásobným rychlým stiskem tlačítka či ikony na displeji a díky zpoždění se vyhodnotí výsledný režim.

```
TimerOffRUchange (Input = not ModbusMaster1.RCU1.Disconnected and
ModbusMaster1.RCU1.ValueChanged);

If TimerOffRUchange.Out Then
     StatusReset.0 = ModbusMaster1.RCU1.ValueChanged;
     RoomMode = ModbusMaster1.RCU1.RoomMode;
     FanMode = ModbusMaster1.RCU1.FanMode;
     Switch_ON_OFF = ModbusMaster1.RCU1.SwitchOnOff;
     Trscor = ModbusMaster1.RCU1.TempSetpointCorr;
     Trm = ModbusMaster1.RCU1.TempMeasured;
     Hum = ModbusMaster1.RCU1.Humidity;
     StatusWrite = true;
…
```

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 **Status** (102-103) s aktuálními hodnotami v proměnných.

```
If (ModbusMaster1.RCU1.RoomMode != RoomMode) Then
     ix = (RoomMode << 1);
     StatusReset = StatusReset | (~ix & 0x0006);
     StatusSet = StatusSet | (ix & 0x0006);
     StatusWrite = true;
EndIf;
If (ModbusMaster1.RCU1.FanMode != FanMode) Then
     ix = (FanMode << 4);
     StatusReset = StatusReset | (~ix & 0x0070);
     StatusSet = StatusSet | (ix & 0x0070);
     StatusWrite = true;
EndIf;
If (ModbusMaster1.RCU1.SwitchOnOff != Switch_ON_OFF) Then
     StatusReset.3 = not Switch_ON_OFF;
     StatusSet.3 = Switch_ON_OFF;
     StatusWrite = true;
EndIf;
If (Abs(ModbusMaster1.RCU1.TempSetpointCorr - Trscor) > 0.5) Then
     ModbusMaster1.RCU1.TempSetpointCorr = Trscor;
EndIf;
If (StatusReset != ModbusMaster1.RCU1.StatusReset) or StatusWrite Then
     ModbusMaster1.RCU1.StatusReset = StatusReset;
EndIf;
If (StatusSet != ModbusMaster1.RCU1.StatusSet) or StatusWrite Then
     ModbusMaster1.RCU1.StatusSet = StatusSet
EndIf;
If (Abs(ModbusMaster1.RCU1.TempSetpoint - Trs) > 0.1) Then
     ModbusMaster1.RCU1.TempSetpoint = Trs;
EndIf;
```

Uvedené algoritmy jsou součástí přílohy aplikační poznámky. Jedná se o ukázkovou aplikaci s názvem modbus_p03_cz_xx.dsox vytvořenou ve vývojovém prostředí DetStudio. Aplikace je vytvořena pro řídicí systém **AMR-OP87 RevA**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

7.1.2 AMR-OP7xRHC

Oproti nástěnnému ovladači **AMR-OP7xRH** je doplněna obsluha čtení registru s CO_2.

```
CO2m = ModbusMaster1.RCU1.CO2;
```

Popis a význam jednotlivých registrů je možné nalézt v dokumentaci daného nástěnného ovladače.

Uvedené algoritmus je součástí přílohy aplikační poznámky. Jedná se o ukázkovou aplikaci s názvem modbus_p04_cz_xxx.dsox vytvořenou ve vývojovém prostředí DetStudio. Aplikace je vytvořena pro řídicí systém **AMR-OP87 RevA**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

7.1.3 AMR-OP40RHC

Oproti dříve zmíněným nástěnným ovladačům se u tohoto ovladače nevyužívají stavové registry pro nastavení režimu místnosti a ventilátoru. U tohoto ovladače je vytvořena obsluha čtení registrů pro měřené hodnoty teploty, vlhkosti, CO_2 a jasu LED. Dále je řešen zápis jasu LED. K zápisu jasu LED dojde, když byl detekován rozpad komunikace a komunikace je již opět funkční. Dále je možné při navázané komunikaci změnit jas LED zápisem do proměnné **LED**.

```
If ModbusMaster1.RCU1.Disconnected Then
     StatusWrite = true;
Else
     If StatusWrite then
          StatusWrite = false;
          ModbusMaster1.RCU1.LED_brightness = LED;
     EndIf;
     Trm = ModbusMaster1.RCU1.TempMeasured;
     Hum = ModbusMaster1.RCU1.Humidity;
     CO2m = ModbusMaster1.RCU1.CO2;
     If (Abs(ModbusMaster1.RCU1.LED_brightness - LED) > 0.5) Then
          ModbusMaster1.RCU1.LED_brightness = LED;
     Else
        LED = ModbusMaster1.RCU1.LED_brightness;
     EndIf;
EndIf;
```

Popis a význam jednotlivých registrů je možné nalézt v dokumentaci daného nástěnného ovladače.

Uvedené algoritmy jsou součástí přílohy aplikační poznámky. Jedná se o ukázkovou aplikaci s názvem modbus_p05_cz_xx.dsox vytvořenou ve vývojovém prostředí DetStudio. Aplikace je vytvořena pro řídicí systém **AMR-OP87 RevA**. Lze jej však změnit pro jakýkoliv jiný řídicí systém, osazený sériovým komunikačním rozhraním, pomocí menu DetStudia „Nástroje/Změnit typ stanice“.

8 Technická podpora

Veškeré informace ohledně možností komunikace protokolem MODBUS RTU v zařízeních firmy AMiT, Vám poskytne oddělení technické podpory firmy AMiT. Technickou podporu můžete kontaktovat nejlépe prostřednictvím e-mailu 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.